

# Informatique des expériences

Du détecteur à la mesure  
Roman Le Montagner - IJCLab



# Plan du cours

---

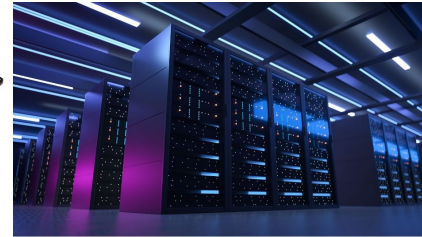
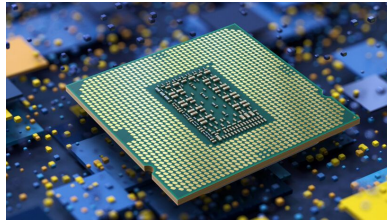
- I. Fink, un logiciel pour le traitement automatique d'alerte astronomique
- II. Qu'est ce qu'un logiciel ?
- III. Qualité logicielle
- IV. Architecture logicielle
- V. Tests
- VI. Documentations et Normes
- VII. Gestion de Version
- VIII. Integration Continue / Déploiement Continue
- IX. Packaging & Publication
- X. Reproductibilité

# Le numérique

---

Pourquoi numériser ?

- Stockage fiable (**reproductibilité**, code correcteur, transmission simple)
- Partage des données (**interopérabilité**, formats standard)
- Débruitage et correction (**calibration**, prétraitement, compression)
- Continuité et traçabilité des mesures (**version**, archivage)
- Traitement automatique (**algorithmes**, IA, filtrage, mesures statistiques)



# Un logiciel pour le traitement automatique d'alerte astro.

---







# Quelques chiffres ...

**Champ de vue:** 3.5 degrés

- 7 fois la Lune !

**Diamètre miroir primaire:** 8.4 m

- 5 fois la taille d'un humain moyen

**Nombre de pixel caméra:** 3.2 Gpixels

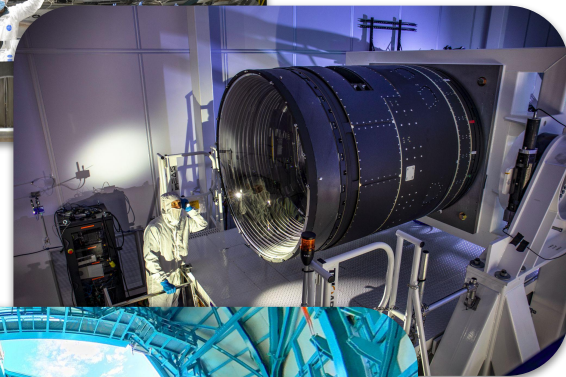
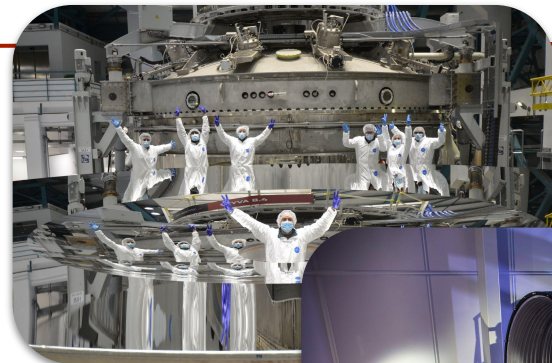
- 1000 fois un smartphone

**Volume de donnée nocturne:** 20TB/nuit

- 20 disque dur standard !

**Flux d'alerte nocturne:** 1TB/night

- 10,000,000 notifications sur vos smartphone





# Rubin/LSST Flux de donnée



## Observatoire Rubin (2025+)

- **20TB d'images / nuit**
- **1TB d'alerts / nuit:** x100-x1000 au dessus des flux actuelles
- *Everything matters a priori*

**Now**

### Raw Data

Sequential 30s image, 20TB/night

**60s**

### Prompt Data Product

Difference Image Analysis  
Alerts: up to 10 million per night

**24h**

### Prompt Products DataBase

Images, Object and Source catalogs from DIA  
Orbit catalog for ~6 million Solar System bodies

**Year**

### Annual Data Release

Accessible via the LSST Science Platform & LSST Data Access Centers.

**End**

### Final 10yr Data Release

Images: 5.5 million x 3.2 Gpx  
Catalog: 15PB, 37 billion objects

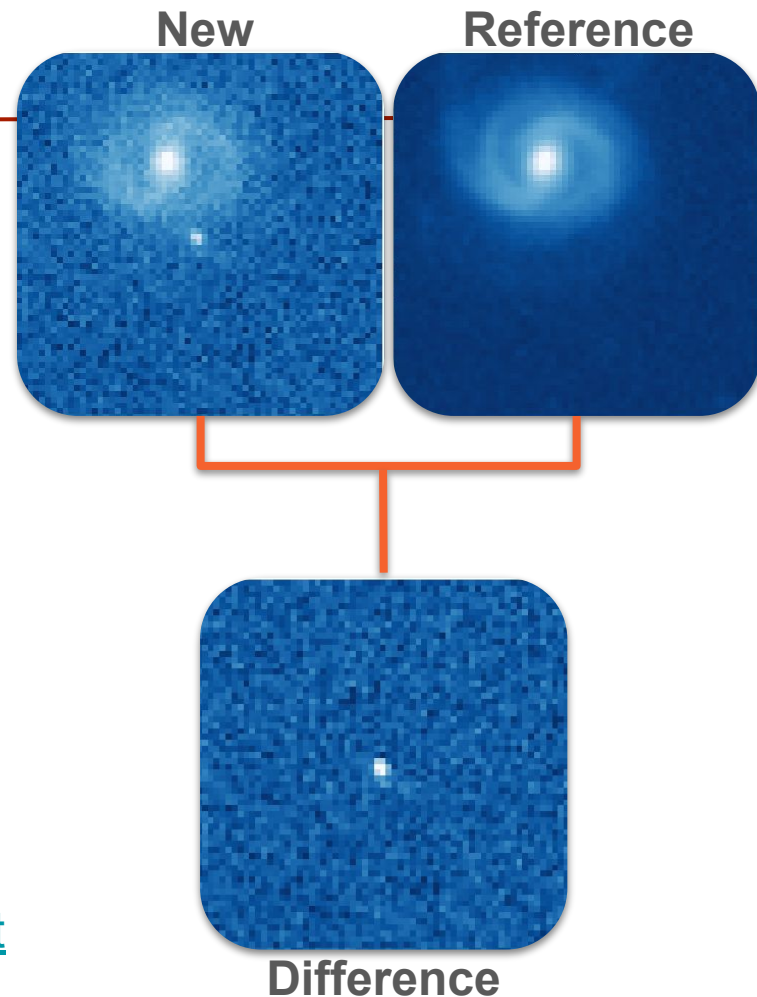
# Contenu des alertes

## Alertes basées sur l'analyse d'images différentielles (DIA)

Chaque alerte contient :

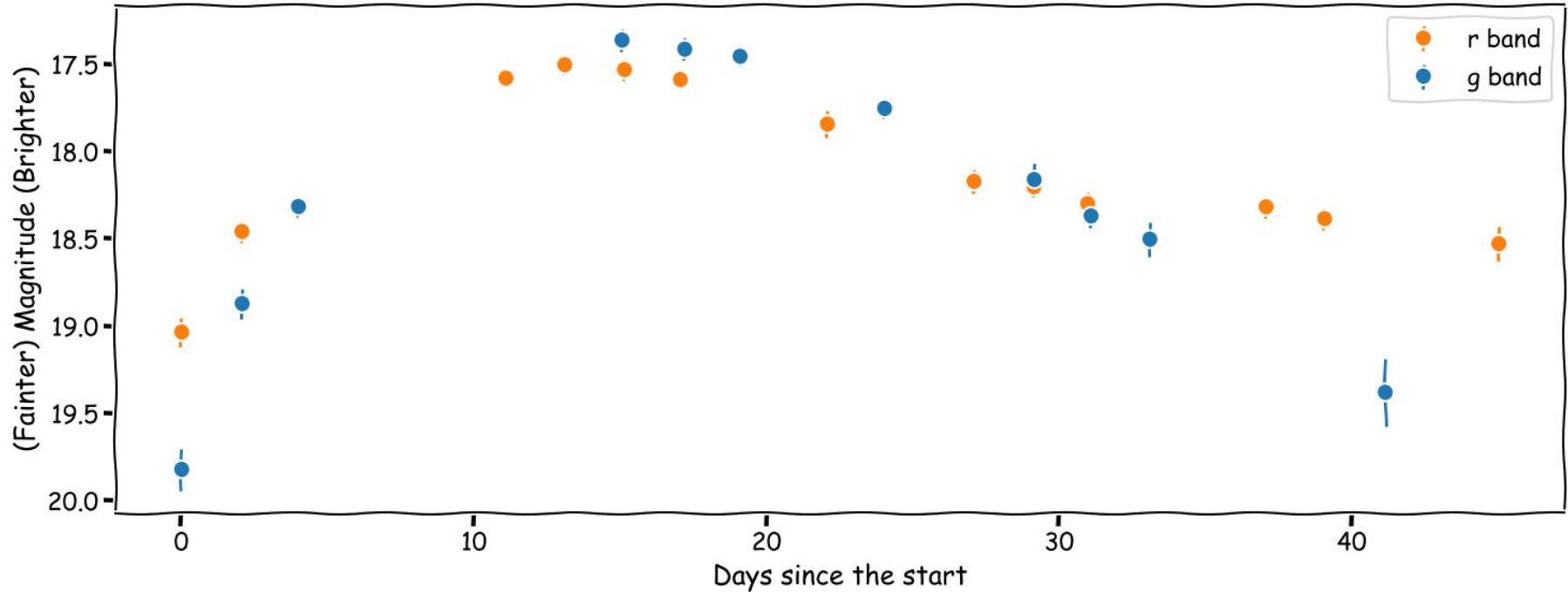
- Des informations sur la nouvelle détection (estimation de magnitude, position, ...)
- Des informations sur les sources voisines (Gaia, Pan-STARRS)
- L'historique des observations à cette position
- De petites images autour de la détection (60×60 pixels)

More information: [https://github.com/lsst/alert\\_packet](https://github.com/lsst/alert_packet)



# Courbe de lumière (Supernovae de type Ia)

Observations à intervalles irréguliers. Cadence variable d'une bande à l'autre.



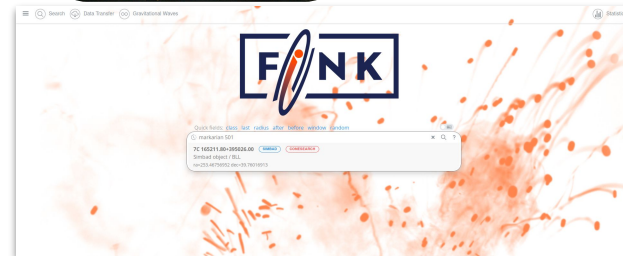
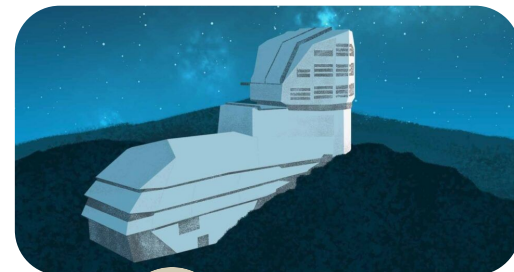
# Fink in a nutshell



<https://fink-broker.org>

Fink est un *broker* au service de la communauté scientifique, chargé d'ingérer, de classifier, de filtrer et de *redistribuer* les alertes à la communauté scientifique.

- À partir de 2025 : plus de **70 collaborateurs**, répartis dans **15 pays**.
- Services déployés sur de *grands clouds OpenStack* (UPSaclay & CC-IN2P3).
- *scalable* jusqu'à plusieurs millions d'alertes par nuit.
- En fonctionnement **24h/24** et **7j/7** depuis 2019, au service de plus de **100 utilisateurs uniques par jour** (*scientifiques, système de suivi et amateurs*).



# Fink : Infrastructure informatique

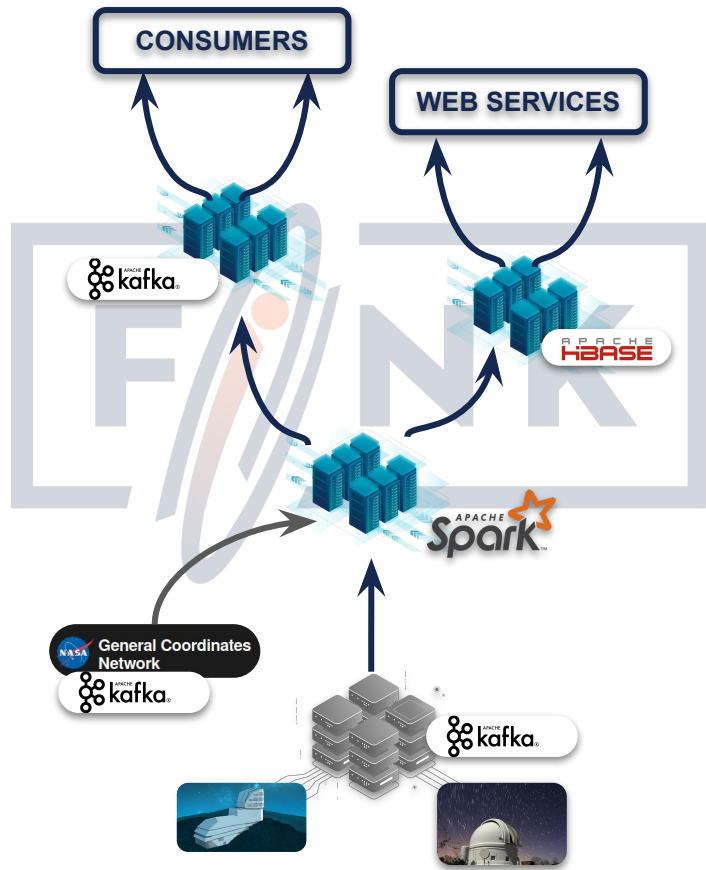
Déploiement de Fink sur des plateformes cloud (UPSaclay, CC-IN2P3).

Utilisation de système distribué :

- Calcul (Spark), Base de donnée (HBase), Streaming (Kafka), Stockage (Ceph et HDFS)
- Orchestration : Mesos et Kubernetes.
- Autoscaling en fonction de la charge.

ML : PyTorch, TensorFlow, scikit-learn (remarque : CPU uniquement !).

Versionnage du code, des modèles et des données.



# Le bloc de base: Le logiciel

---

Logiciel : Une suite d'instructions interprétable par une machine **et** un jeu de données nécessaire aux opérations.

Trois éléments clé:

- Processus : ce que le logiciel fait
- Données : ce que le logiciel manipule
- Interfaces : comment on interagit avec le logiciel



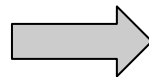
## Le bloc de base: Le logiciel

Logiciel : Une suite d'instructions interprétable par une machine **et** un jeu de données nécessaire aux opérations.

## Données

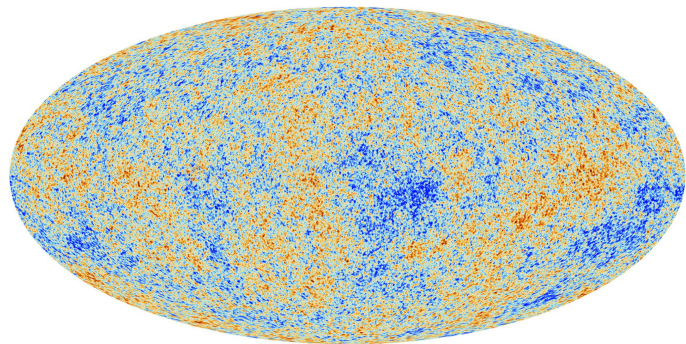
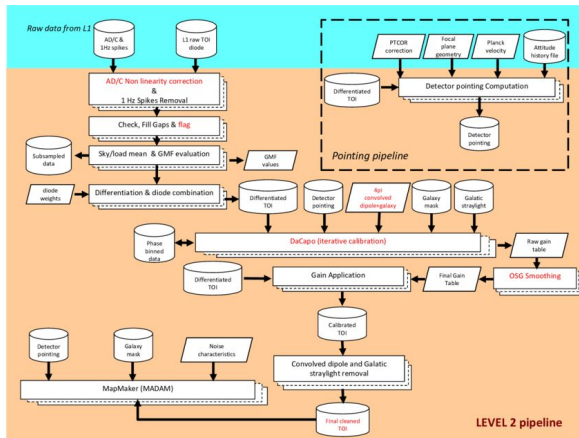
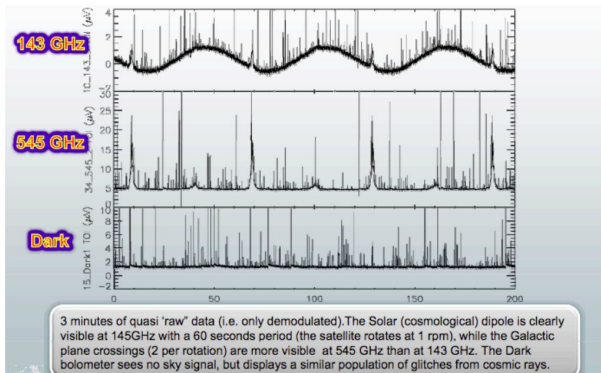


# Traitement



## Produit scientifique

**(nearly) raw data stream**



European Space Agency. *Planck CMB Map*. 2013. ESA,  
[https://www.esa.int/ESA\\_Multimedia/Images/2013/03/Planck\\_CMB](https://www.esa.int/ESA_Multimedia/Images/2013/03/Planck_CMB)

Natoli, P. (2013). *Planck data processing and map-making* [Présentation]. Vietnam School of Physics, IN2P3.  
<http://vietnam.in2p3.fr/2013/Cosmology/transparencies/Natoli1.pdf>

Planck Collaboration, et al. *Planck 2018 Results. II. Low Frequency Instrument Data Processing*. 2020.  
[https://www.researchgate.net/publication/345607185\\_Planck\\_2018\\_results\\_II\\_Low\\_Frequency\\_Instrument\\_data\\_processing](https://www.researchgate.net/publication/345607185_Planck_2018_results_II_Low_Frequency_Instrument_data_processing)

# Programme vs Logiciel

# Programme

## un fichier exécutable ou un script

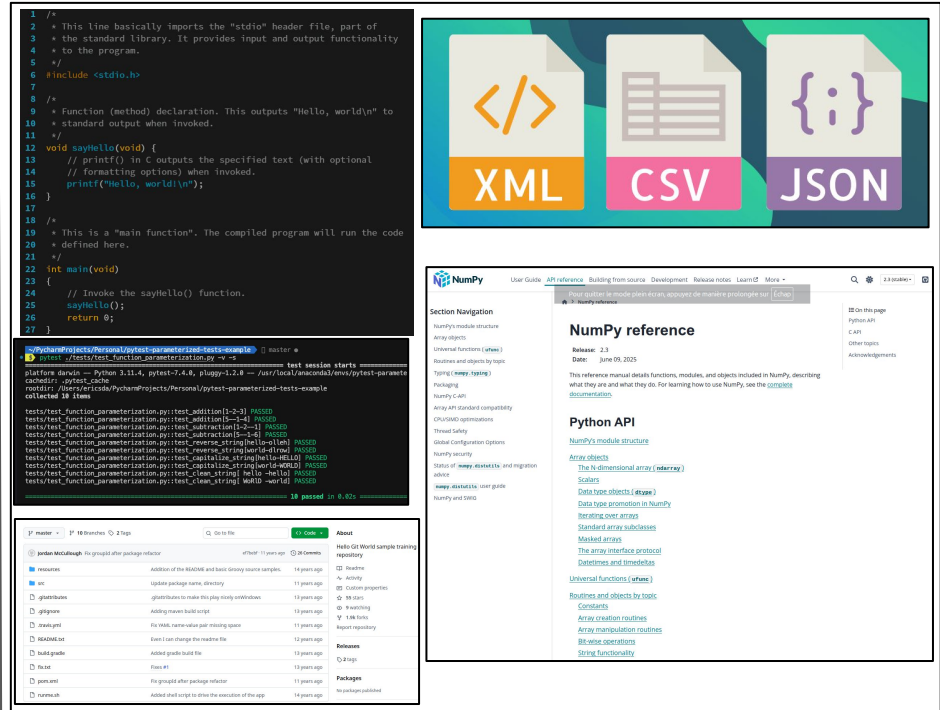
```

1  /*
2   * This line basically imports the "stdio" header file, part of
3   * the standard library. It provides input and output functionality
4   * to the program.
5   */
6  #include <stdio.h>
7
8  /*
9   * Function (method) declaration. This outputs "Hello, world\n" to
10  * standard output when invoked.
11  */
12  void sayHello(void) {
13      // printf() in C outputs the specified text (with optional
14      // formatting options) when invoked.
15      printf("Hello, world!\n");
16  }
17
18  /*
19   * This is a "main function". The compiled program will run the code
20   * defined here.
21   */
22  int main(void)
23  {
24      // Invoke the sayHello() function.
25      sayHello();
26      return 0;
27  }

```

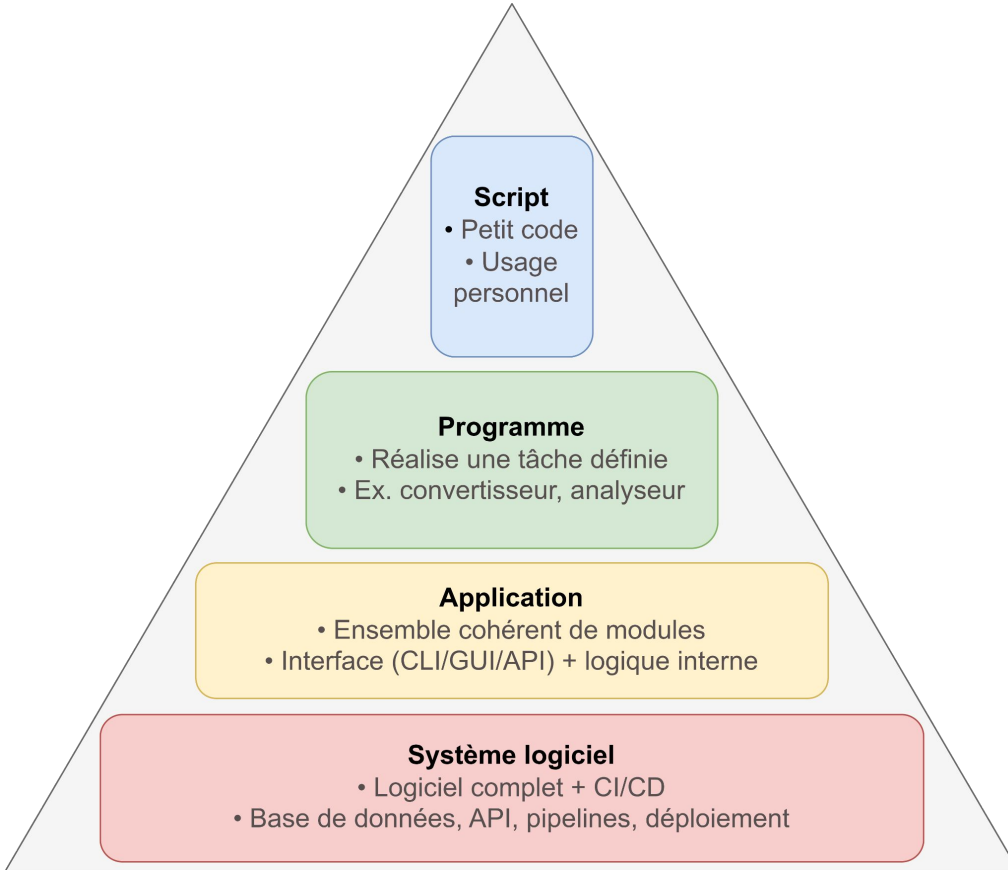
# Logiciel

un ensemble **organisé**



# Les différentes échelles

---



# Les différentes échelles

## Pourquoi distinguer script / programme / application ?

- *Ce que vous développez n'est pas un script quand d'autres doivent l'utiliser.*

Niveau	Robustesse	Tests & Doc	Scénario d'utilisation
Script	minimal	quasi nulle	Usage local, non partagé
Programme	moyenne	basique	Petites équipes
Application	élevée	obligatoire	Utilisateurs multiples
Système logiciel	critique	complet (CI/CD)	Production, pipeline durable

### Script

- Petit code
- Usage personnel

### Programme

- Réalise une tâche définie
- Ex. convertisseur, analyseur

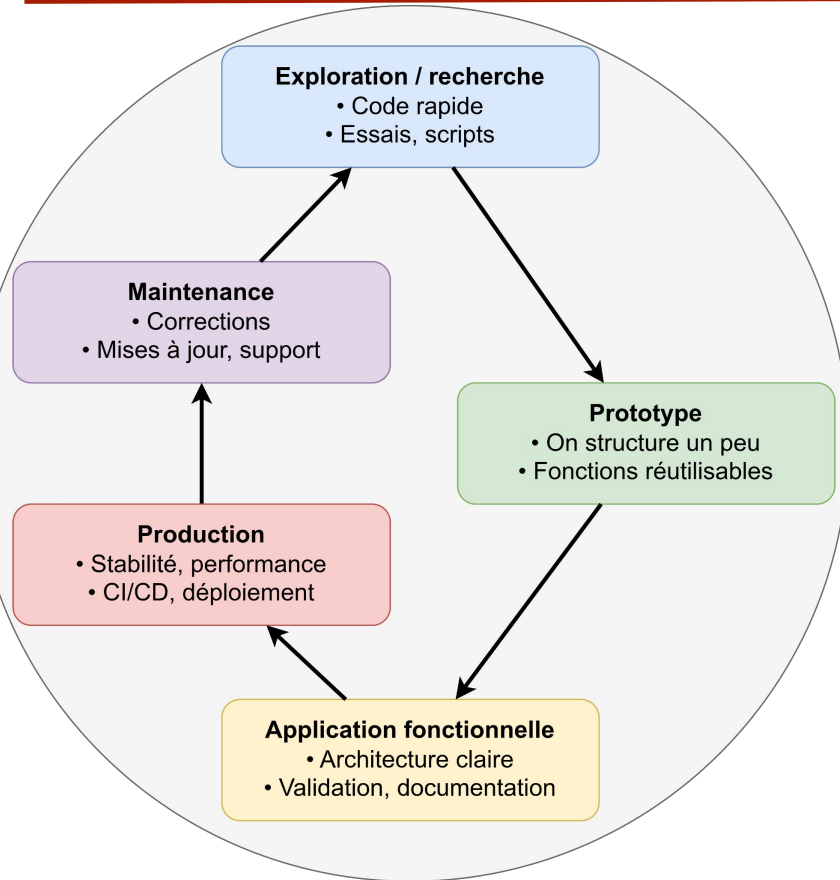
### Application

- Ensemble cohérent de modules
- Interface (CLI/GUI/API) + logique interne

### Système logiciel

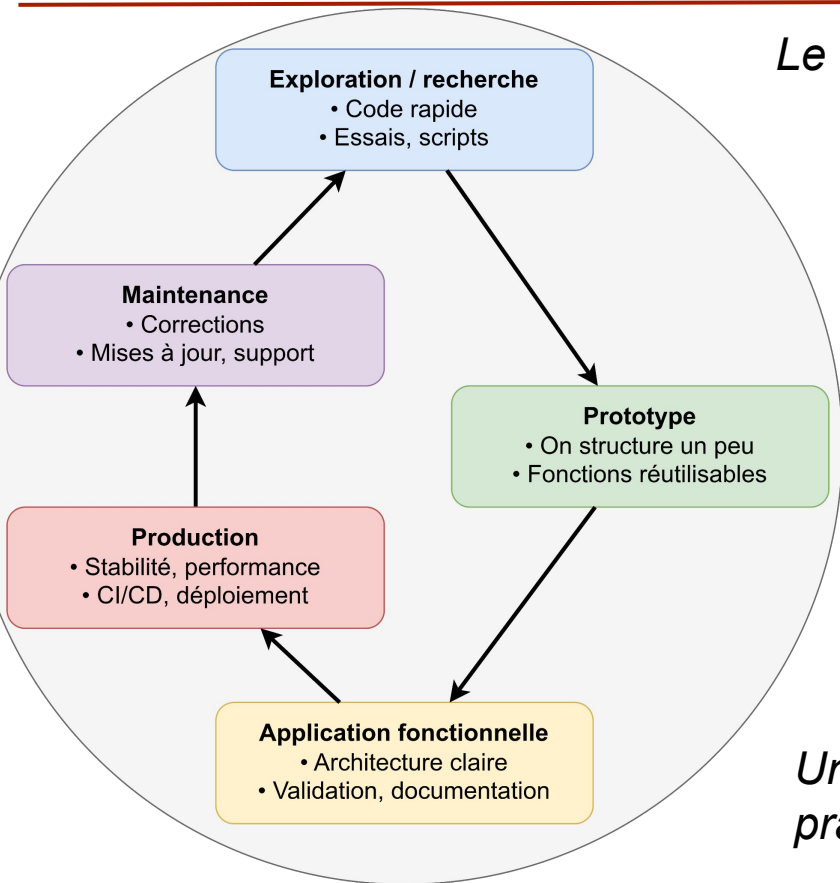
- Logiciel complet + CI/CD
- Base de données, API, pipelines, déploiement

# Cycle de vie d'un logiciel



# Cycle de vie d'un logiciel

*Le cycle de vie d'un logiciel dépend de son contexte.*



## - Cycle scientifique

- Rapidité
- flexibilité
- prototypage rapide
- Priorités: résultats, exploration

## - Cycle industriel

- Stabilité
- Qualité
- Sécurité
- Priorités: prédictibilité, longévité

*Un logiciel scientifique mûrit et finit par adopter les pratiques industrielles lorsqu'il devient critique.*

# La qualité logicielle

---

- **Qu'est ce qu'un logiciel de qualité ?**
  - Un logiciel de qualité est...
    - Fiable (ne casse pas à la moindre erreur)
    - Durable (évolue sans s'effondrer)
    - Compréhensible (lisible, clair)
    - Utilisable (adapté à ses utilisateurs)

# La qualité logicielle

---

- **Qu'est ce qu'un logiciel de qualité ?**
  - Un logiciel de qualité est...
    - Fiable (ne casse pas à la moindre erreur)
    - Durable (évolue sans s'effondrer)
    - Compréhensible (lisible, clair)
    - Utilisable (adapté à ses utilisateurs)
  - Dans un projet scientifique, un logiciel doit :
    - survivre à plusieurs campagnes d'observation
    - s'adapter à de nouveaux instruments ou formats
    - être compris par les futurs doctorants / chercheurs
    - produire des résultats fiables et traçables



# La qualité logicielle

---

- 6 aspect de qualité essentielles:
  1. Robustesse
  2. Maintenabilité
  3. Performance / Scalabilité
  4. Testabilité
  5. Portabilité

# Qualité logicielle - Robustesse

## Entrées imparfaites

- Bruit
- donnée manquante
- formats partiels

Exemples:

- fichier de donnée incomplet
- série temporelle avec des NaN

## Situations inattendues

- Exceptions
- erreurs I/O
- Situations imprévues

Exemples:

- division par zéro
- connexion réseau perdu
- mauvaise permission de fichier

## Valeurs limites

- limites mathématiques
- conditions extrêmes

Exemples:

- valeur proche de zéro
- flux saturé
- horizon temporel très long

Le logiciel ne doit jamais surprendre.  
Il doit résister aux erreurs.  
Il doit échouer proprement.

# Qualité logicielle - Maintenabilité

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

# Qualité logicielle - Maintenabilité

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ python bad.py
Traceback (most recent call last):
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 30, in <module>
    run(data)
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 16, in run
    m = s / len(r)
        ~^~~~~~
ZeroDivisionError: division by zero
$
```

# Qualité logicielle - Maintenabilité

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ python bad.py
Traceback (most recent call last):
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 30, in <module>
    run(data)
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 16, in run
    m = s / len(r)
        ~^~~~~~
```

ZeroDivisionError: division by zero

```
$
```

```
$ python bad.py abc
Traceback (most recent call last):
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 30, in <module>
    run(data)
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 8, in run
    v = float(v)
        ^^^^^^^
```

ValueError: could not convert string to float: 'abc'

```
$
```

# Qualité logicielle - Maintenabilité

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ python bad.py abc
Traceback (most recent call last):
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 30, in <module>
    run(data)
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/bad.py", line 8, in run
    v = float(v)
        ^^^^^^^
ValueError: could not convert string to float: 'abc'
$
```

```
$ python bad.py nan
OK nan nan
$
```

<https://www.destroyallsoftware.com/talks/wat>

# Qualité logicielle - Maintenabilité

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ python bad.py nan
OK nan nan
$
```

```
$ python bad.py 50 60 70
OK 0.4938 0.010042907945411034
$
```

```
$ python bad.py 50 60 300
ALERT 0.5881 0.14214647375154965
$
```



# Qualité logicielle - Maintenabilité

```
35 from __future__ import annotations
36
37 import math
38 from dataclasses import dataclass
39 from typing import Iterable, List
40
41
42 #==== Configuration & constantes ====
43
44 ADC_GAIN = 0.00123 ..... # V / ADU
45 ADC_OFFSET = 0.42 ..... # V
46 STD_ALERT_THRESHOLD = 0.10 ..... # V
47
48
49 #==== Modèle de donnée ====
50
51 @dataclass
52 class RawSample:
53     """Une mesure brute lue depuis la chaîne d'acquisition."""
54     value: float
55
56
57 @dataclass
58 class CalibratedSample:
59     """Une mesure convertie en unités physiques (volts)."""
60     volts: float
61
```



# Qualité logicielle - Maintenabilité

```
35 from __future__ import annotations
36
37 import math
38 from dataclasses import dataclass
39 from typing import Iterable, List
40
41
42 #==== Configuration & constantes ====
43
44 ADC_GAIN = 0.00123 ..... # V / ADU
45 ADC_OFFSET = 0.42 ..... # V
46 STD_ALERT_THRESHOLD = 0.10 ..... # V
47
48
49 #==== Modèle de donnée ====
50
51 @dataclass
52 class RawSample:
53     """Une mesure brute lue depuis la chaîne d'acquisition."""
54     value: float
55
56
57 @dataclass
58 class CalibratedSample:
59     """Une mesure convertie en unités physiques (volts)."""
60     volts: float
61
```

```
63 #==== Fonctions métier ====
64
65 def parse_raw_values(args: Iterable[str]) -> List[RawSample]:
66     """Transforme une liste de chaînes en valeurs brutes numériques.
67     ... Ignore les entrées vides ou invalides au lieu de planter.
68     ... """
69     samples: List[RawSample] = []
70     for arg in args:
71         if not arg:
72             continue
73         try:
74             samples.append(RawSample(float(arg)))
75         except ValueError:
76             # On loguerait un warning dans un vrai logiciel
77             continue
78     return samples
79
80
81
82 def calibrate(sample: RawSample) -> CalibratedSample:
83     """Convertit une mesure brute en tension calibrée.
84     ... La loi de calibration est :  $V = G \cdot \text{ADU} + \text{offset}$ .
85     ... """
86     volts = ADC_GAIN * sample.value + ADC_OFFSET
87     volts = max(volts, 0.0) ..... # on évite les valeurs négatives
88     return CalibratedSample(volts=volts)
89
90
91
92 def mean_and_std(values: Iterable[float]) -> tuple[float, float]:
93     """Calcule moyenne et écart-type d'une séquence de réels."""
94     data = list(values)
95     if not data:
96         raise ValueError("Impossible de calculer des statistiques sur une liste vide.")
97     mean = sum(data) / len(data)
98     var = sum((x - mean) ** 2 for x in data) / len(data)
99     std = math.sqrt(var)
100     return mean, std

```

# Qualité logicielle - Maintenabilité

---

```
102
103 def analyse_samples(raw_args: Iterable[str]) -> None:
104     """Pipeline complet: parsing, calibration, statistiques, décision."""
105     raw = parse_raw_values(raw_args)
106     calibrated = [calibrate(s).volts for s in raw]
107
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
112
113
114 #==== Point d'entrée ====
115
116 if __name__ == "__main__":
117     import sys
118
119     analyse_samples(sys.argv[1:])
120
```

# Qualité logicielle - Maintenabilité

```
102
103 def analyse_samples(raw_args: Iterable[str]) -> None:
104     """Pipeline complet : parsing, calibration, statistiques, décision."""
105     raw = parse_raw_values(raw_args)
106     calibrated = [calibrate(s).volts for s in raw]
107
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
112
113
114 #==== Point d'entrée ====
115
116 if __name__ == "__main__":
117     import sys
118
119     analyse_samples(sys.argv[1:])
120
```

```
$ python good.py
Traceback (most recent call last):
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/good.py", line 119, in <module>
    analyse_samples(sys.argv[1:])
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/good.py", line 108, in analyse_samples
    mean, std = mean_and_std(calibrated)
                ~~~~~^~~~~~
  File "/home/roman/Documents/Work/Mission/Roscoff_11_2025/good.py", line 96, in mean_and_std
    raise ValueError("Impossible de calculer des statistiques sur une liste vide.")
ValueError: Impossible de calculer des statistiques sur une liste vide.
$
```

# Qualité logicielle - Maintenabilité

```
102
103 def analyse_samples(raw_args: Iterable[str]) -> None:
104     """Pipeline complet: parsing, calibration, statistiques, décision."""
105     raw = parse_raw_values(raw_args)
106     calibrated = [calibrate(s).volts for s in raw]
107
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
112
113
114 #==== Point d'entrée ====
115
116 if __name__ == "__main__":
117     import sys
118
119     analyse_samples(sys.argv[1:])
120
```

```
$ python good.py 50 60 70
OK | mean=0.494 V | std=0.010 V
$
```

# Qualité logicielle - Maintenabilité

---

```
102
103 def analyse_samples(raw_args: Iterable[str]) -> None:
104     """Pipeline complet: parsing, calibration, statistiques, décision."""
105     raw = parse_raw_values(raw_args)
106     calibrated = [calibrate(s).volts for s in raw]
107
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
112
113
114 #==== Point d'entrée ====
115
116 if __name__ == "__main__":
117     import sys
118
119     analyse_samples(sys.argv[1:])
120
```

- code lisible
- Documentation
- Typage
- Modulaire
- Dépendances claires

*Un logiciel maintenable peut évoluer sans réécriture complète.*

# Qualité logicielle - Performance

---

**Performance** : Aller vite sur un dataset donné

- Temps d'exécution raisonnable
- Optimisations CPU/GPU, I/O, allocations
- Bons choix algorithmiques (éviter  $O(N^2)$  si possible)

Exemples: *Extraire les particules d'un évènement en quelques millisecondes*

# Qualité logicielle - Performance / Scalabilité

---

**Performance** : Aller vite sur un dataset donné

- Temps d'exécution raisonnable
- Optimisations CPU/GPU, I/O, allocations
- Bons choix algorithmiques (éviter  $O(N^2)$  si possible)

Exemples: *Extraire les particules d'un évènement en quelques millisecondes*

Exemples: *Gérer 100 kHz, 1 MHz, 100 MHz jusqu'à 1 GHz d'événements*

**Scalabilité** : Performance stable quand le volume croît

- Supporter un volume de données  $\times 1000$ .
- Gestion mémoire, streaming, parallélisation

# Qualité logicielle - Testabilité

---

*Un logiciel doit être conçu pour être testé dès le départ.*



# Qualité logicielle - Testabilité

---

*Un logiciel doit être conçu pour être testé dès le départ.*

Comment rendre un logiciel testable ?

- interfaces claires
- Composants indépendants
- Résultats déterministes
- Pas d'effets cachés (I/O, états globales)

```
81
82 def calibrate(sample: RawSample) -> CalibratedSample:
83     """Convertit une mesure brute en tension calibrée.
84
85     La loi de calibration est :  $V = G \cdot ADU + offset$ .
86     """
87     volts = ADC_GAIN * sample.value + ADC_OFFSET
88     volts = max(volts, 0.0) # on évite les valeurs négatives
89     return CalibratedSample(volts=volts)
90
```

# Qualité logicielle - Testabilité

*Un logiciel doit être conçu pour être testé dès le départ.*

Comment rendre un logiciel testable ?

- interfaces claires
- Composants indépendants
- Résultats déterministes
- Pas d'effets cachés (I/O, états globales)

Comment on teste ?

- Tests unitaire
- Tests d'intégration

```
81
82 def calibrate(sample: RawSample) -> CalibratedSample:
83     """Convertit une mesure brute en tension calibrée.
84
85     La loi de calibration est :  $V = G \cdot ADU + offset$ .
86     """
87     volts = ADC_GAIN * sample.value + ADC_OFFSET
88     volts = max(volts, 0.0) # on évite les valeurs négatives
89     return CalibratedSample(volts=volts)
90
```

```
103 def analyse_samples(raw_args: Iterable[str]) -> None:
104     """Pipeline complet : parsing, calibration, statistiques, décision."""
105     raw = parse_raw_values(raw_args)
106     calibrated = [calibrate(s).volts for s in raw]
107
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
```

# Qualité logicielle - Testabilité

*Un logiciel doit être conçu pour être testé dès le départ.*

Comment rendre un logiciel testable ?

- interfaces claires
- Composants indépendants
- Résultats déterministes
- Pas d'effets cachés (I/O, états globales)

Comment on teste ?

- Tests unitaire
- Tests d'intégration
- Jeu de données de référence
- CI automatique (forges)

```
81
82 def calibrate(sample: RawSample) -> CalibratedSample:
83     """Convertit une mesure brute en tension calibrée.
84
85     La loi de calibration est :  $V = G \cdot ADU + offset$ .
86     """
87     volts = ADC_GAIN * sample.value + ADC_OFFSET
88     volts = max(volts, 0.0) # on évite les valeurs négatives
89     return CalibratedSample(volts=volts)
90
```

```
103 def analyse_samples(raw_args: Iterable[str]) -> None:
104     """Pipeline complet : parsing, calibration, statistiques, décision."""
105     raw = parse_raw_values(raw_args)
106     calibrated = [calibrate(s).volts for s in raw]
107
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
```

# Qualité logicielle - Testabilité

*Un logiciel doit être conçu pour être testé dès le départ.*

Comment rendre un logiciel testable ?

- interfaces claires
- Composants indépendants
- Résultats déterministes
- Pas d'effets cachés (I/O, états globales)

Comment on teste ?

- Tests unitaire
- Tests d'intégration
- Jeu de données de référence
- CI automatique (forages)

```
81
82 def calibrate(sample: RawSample) -> CalibratedSample:
83     """Convertit une mesure brute en tension calibrée.
84
85     La loi de calibration est :  $V = G \cdot ADU + offset$ .
86     """
87     volts = ADC_GAIN * sample.value + ADC_OFFSET
88     volts = max(volts, 0.0) # on évite les valeurs négatives
89     return CalibratedSample(volts=volts)
90
```

```
103
104 def analyse_samples(raw_args: Iterable[str]) -> None:
105     """Pipeline complet : parsing, calibration, statistiques, décision."""
106     raw = parse_raw_values(raw_args)
107     calibrated = [calibrate(s).volts for s in raw]
108     mean, std = mean_and_std(calibrated)
109
110     status = "ALERT" if std > STD_ALERT_THRESHOLD else "OK"
111     print(f"{status} | mean={mean:.3f} V | std={std:.3f} V")
```

## Reproductibilité

# Qualité logicielle - Portabilité

---

Problématique : Architecture matérielle et logiciel hétérogène

- Linux, macOS, Windows
- environnement local, serveur, cluster HPC
- matériel CPU/GPU (AMD, NVIDIA, INTEL, ARM...)

# Qualité logicielle - Portabilité

---

Problématique : Architecture matérielle et logiciel hétérogène

- Linux, macOS, Windows
- environnement local, serveur, cluster HPC
- matériel CPU/GPU (AMD, NVIDIA, INTEL, ARM...)

---

Garantir les mêmes résultats

- dépendances identiques
  - python
  - système
- mêmes configurations
- Gestions d'environnement
  - venv
  - conda
  - pdm

Garantir le lancement du logiciel

- Documentation claire
- Virtualisation
  - Docker
  - Singularity
- portabilité CPU/GPU
  - NumPy ↔ CuPy
  - Pytorch / tensorflow
  - SYCL / oneAPI / Kokkos

# Qualité logicielle - Résumé

---

**Robustesse** — résister aux entrées imparfaites et aux situations imprévues

**Maintenabilité** — pouvoir évoluer sans tout casser

**Performance & Scalabilité** — rester efficace, même quand la charge augmente

**Testabilité** — garantir la vérification et la constance des résultats

**Portabilité** — fonctionner partout, du laptop au cluster HPC

# Penser l'architecture logicielle

---

Architecture = organisation intelligente du logiciel / code

Sans architecture :

- maintenabilité et testabilité compromise
- Ajout / mise à jour du code difficile
- fichier principal gigantesque (plusieurs milliers de lignes)
- dépendances entremêlées



# 3 principes importants en architecture logicielle

---

## 1. Séparation des responsabilités

Définir les briques du logiciel

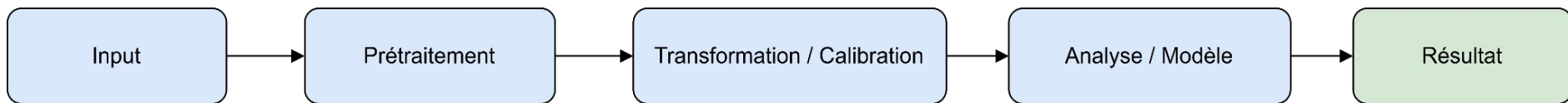
- I/O → Lecture / écriture des données
- Modèle → représentation des données
- Processing → transformer les données
- Analyse → algorithme, traitement sur les données
- Résultats → plots, visualisation des données

# 3 principes importants en architecture logicielle

---

1. Séparation des responsabilités
2. **Flux logique des données (dataflow)**

**logiciel scientifique** → Pipeline, une suite d'étape



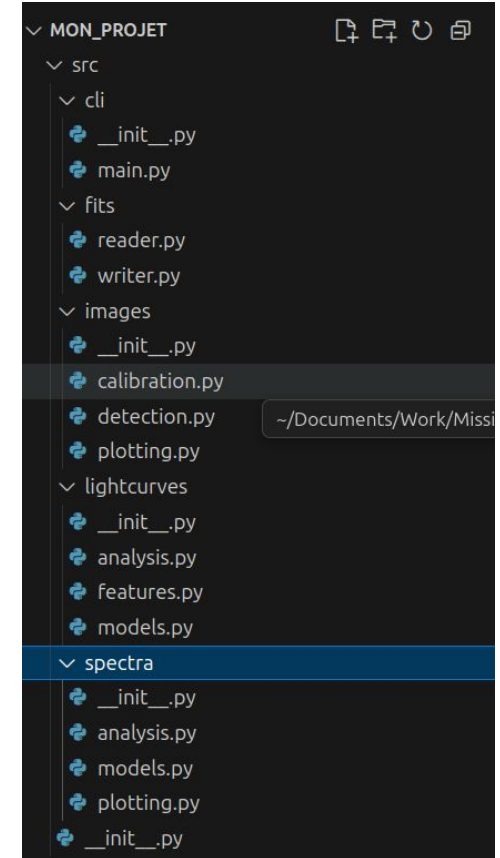
- facile à tester, étape par étape
- lisibilité immédiate

# 3 principes importants en architecture logicielle

1. Séparation des responsabilités
2. Flux logique des données (dataflow)

## 3. **Modularité** & encapsulation

**module = bloc de code cohérent, autonome  
servant un objectif via une interface claire**



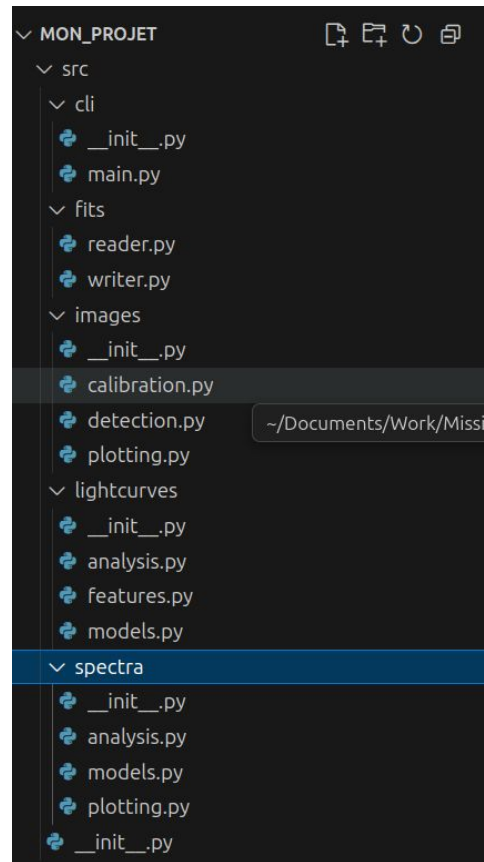
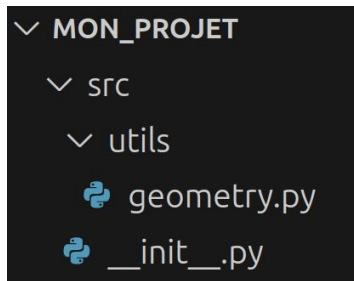
# 3 principes importants en architecture logicielle

1. Séparation des responsabilités
2. **Flux logique des données (dataflow)**
3. **Modularité & encapsulation**

**module = bloc de code cohérent, autonome  
servant un objectif via une interface claire**

Eviter les pièges: “utils/”, “helpers/”, “misc/”, “core/”

Exemple: Une fonction “Convertir RA/Dec en radians”



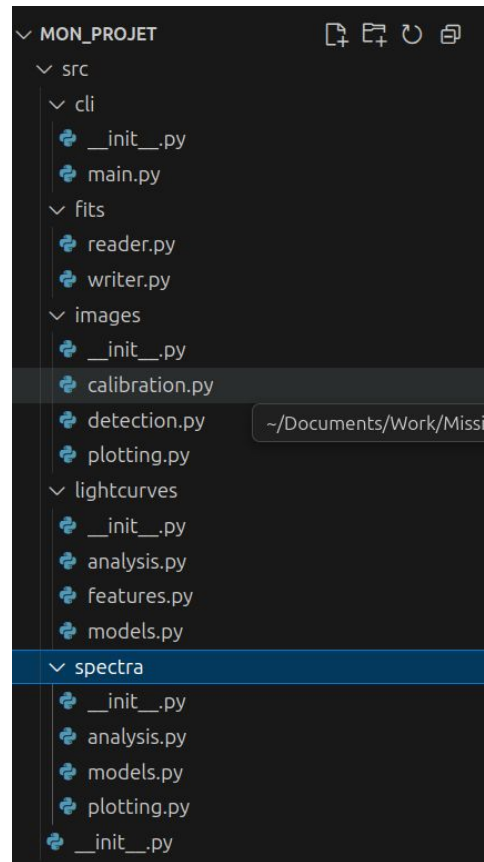
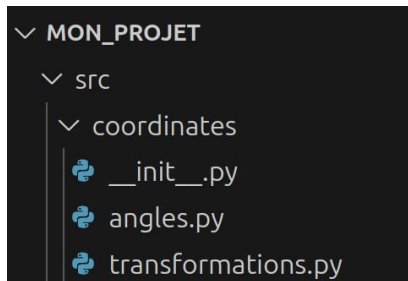
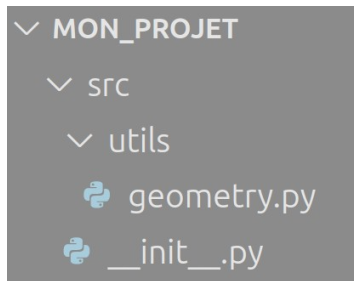
# 3 principes importants en architecture logicielle

1. Séparation des responsabilités
2. **Flux logique des données (dataflow)**
3. **Modularité & encapsulation**

**module = bloc de code cohérent, autonome  
servant un objectif via une interface claire**

Eviter les pièges: “utils/”, “helpers/”, “misc/”, “core/”

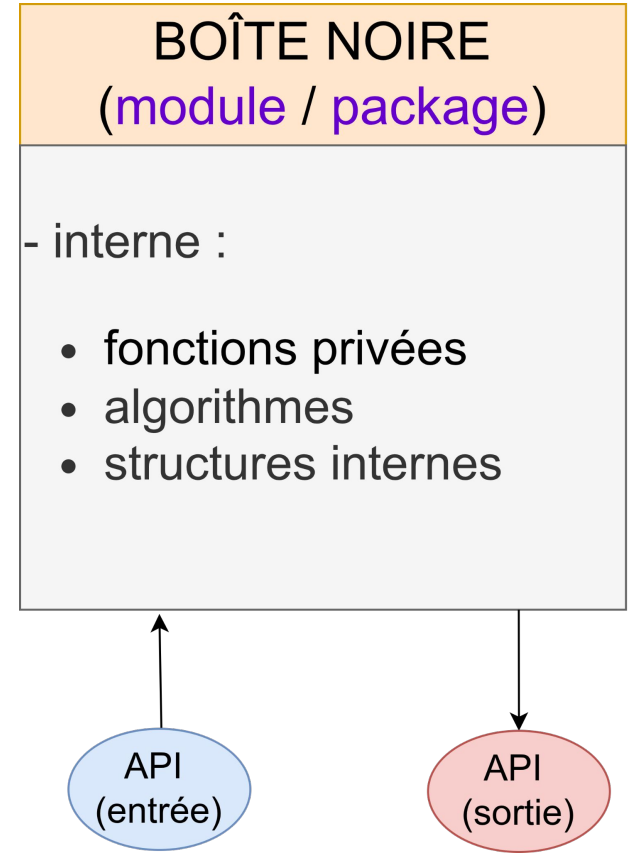
Exemple: Une fonction “Convertir RA/Dec en radians”



# 3 principes importants en architecture logicielle

1. Séparation des responsabilités
2. **Flux logique des données (dataflow)**
3. **Modularité & encapsulation**  
**encapsulation = cacher la complexité interne**  
**et exposer uniquement le nécessaire**

→ API : Application Programming Interface



# Tests : fondations

---

Pourquoi tester ?

- détections d'erreur → éviter effets cascades
- confiance dans le code → refactoring safe
- documentation exécutable → « exemples vérifiés »
- non-régression → reproductibilité scientifique

# Type de tests

---

## Tests unitaires

permet de tester des fonctions isolé → *rapide*

```
64
65 def parse_raw_values(args: Iterable[str]) -> List[RawSample]:
66     """Transforme une liste de chaînes en valeurs brutes numériques.
67
68     Ignore les entrées vides ou invalides au lieu de planter.
69     """
70     samples: List[RawSample] = []
71     for arg in args:
72         if not arg:
73             continue
74         try:
75             samples.append(RawSample(float(arg)))
76         except ValueError:
77             # On loguerait un warning dans un vrai logiciel
78             continue
79     return samples
80
```



# Type de tests

## Tests unitaires

permet de tester des fonctions isolé → *rapide*

```
65 def parse_raw_values(args: Iterable[str]) -> List[RawSample]:
66     """Transforme une liste de chaînes en valeurs brutes numériques.
67     Ignore les entrées vides ou invalides au lieu de planter.
68     """
69     samples: List[RawSample] = []
70     for arg in args:
71         if not arg:
72             continue
73         try:
74             samples.append(RawSample(float(arg)))
75         except ValueError:
76             # On loguerait un warning dans un vrai logiciel
77             continue
78     return samples
```

```
def test_parse_valid_strings(self):
    args = ["1.0", "2.5", "3"]
    samples = parse_raw_values(args)

    assert len(samples) == 3
    assert isinstance(samples[0], RawSample)
    assert [s.value for s in samples] == [1.0, 2.5, 3.0]
```

```
def test_parse_ignores_empty_and_invalid(self):
    args = ["", "- ", "abc", "42", "NaN", "3.14"]
    samples = parse_raw_values(args)

    values = [s.value for s in samples]

    assert len(samples) == 3
    assert values[0] == 42.0
    assert math.isnan(values[1])
    assert values[2] == 3.14
```

# Type de tests

## Tests intégration

permet de tester plusieurs modules → *API globale*

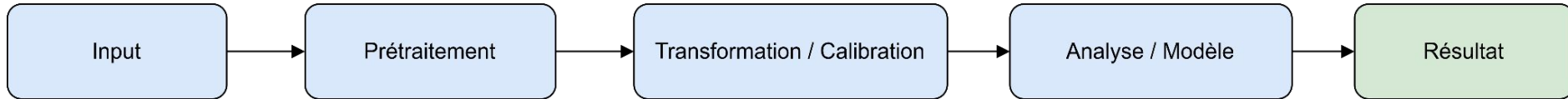
```
def parse_raw_values(args: Iterable[str]) -> List[RawSample]:  
def calibrate(sample: RawSample) -> CalibratedSample:  
def mean_and_std(values: Iterable[float]) -> tuple[float, float]:
```

```
class TestIntegrationPipeline:  
    def test_full_pipeline_nominal(self):  
        """  
        Test d'intégration "heureux":  
        - on part de chaînes de caractères  
        - on parse en RawSample  
        - on calibre en CalibratedSample  
        - on calcule moyenne et écart-type sur les volts  
        """  
  
        # 1) Entrée simulée (ex: valeurs ADC en argument de ligne de commande)  
        raw_args = ["100", "110", "120"]  
  
        # 2) Parsing → RawSample  
        raw_samples = parse_raw_values(raw_args)  
        assert [s.value for s in raw_samples] == [100.0, 110.0, 120.0]  
  
        # 3) Calibration → CalibratedSample  
        calibrated = [calibrate(s) for s in raw_samples]  
        volts = [c.volts for c in calibrated]  
  
        # 4) Statistiques sur les volts  
        mean, std = mean_and_std(volts)  
  
        # 5) Calcul attendu "à la main" en utilisant les mêmes constantes  
        adus = [100.0, 110.0, 120.0]  
        expected_volts = [max(ADC_GAIN * v + ADC_OFFSET, 0.0) for v in adus]  
        expected_mean = sum(expected_volts) / len(expected_volts)  
        expected_var = sum((v - expected_mean) ** 2 for v in expected_volts) / len(  
            expected_volts  
        )  
        expected_std = math.sqrt(expected_var)  
  
        assert volts == pytest.approx(expected_volts)  
        assert mean == pytest.approx(expected_mean)  
        assert std == pytest.approx(expected_std)
```

# Type de tests

## Tests End-to-End

variante du test d'intégration (scénario utilisateur)

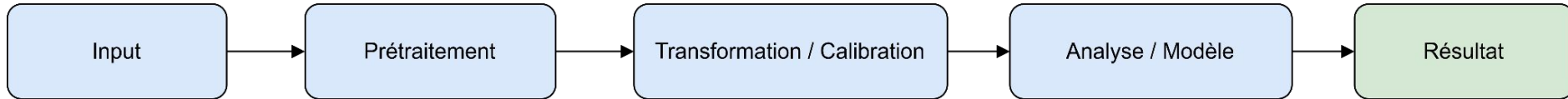


```
6  # On récupère le chemin absolu du fichier à tester (good.py)
7  SCRIPT_PATH = os.path.join(os.path.dirname(__file__), "..", "good.py")
8
9
10 class TestCLIEndToEnd:
11     def test_cli_ok_low_dispersion(self):
12         """
13         Reproduit :
14         $ python good.py 120 123 125
15         """
16         result = subprocess.run(
17             [sys.executable, SCRIPT_PATH, "120", "123", "125"],
18             capture_output=True,
19             text=True,
20             check=True,
21         )
22         assert result.stdout.strip() == "OK | mean=0.571 V | std=0.003 V"
23         assert result.stderr == ""
```

# Type de tests

## Tests End-to-End

variante du test d'intégration (scénario utilisateur)



```
6 # On récupère le chemin absolu du fichier à tester (good.py)
7 SCRIPT_PATH = os.path.join(os.path.dirname(__file__), "..", "good.py")
8
9
10 class TestCLIEndToEnd:
11     def test_cli_ok_low_dispersion(self):
12         """
13         Reproduit :
14         $ python good.py 120 123 125
15         """
16         result = subprocess.run(
17             [sys.executable, SCRIPT_PATH, "120", "123", "125"],
18             capture_output=True,
19             text=True,
20             check=True,
21         )
22         assert result.stdout.strip() == "OK | mean=0.571 V | std=0.003 V"
23         assert result.stderr == ""
```

```
$ python good.py 120 123 125
OK | mean=0.571 V | std=0.003 V
$
```

# Type de tests

## Property based testing (<https://hypothesis.readthedocs.io/en/latest/>)



Tester des *invariants* : on génère des dizaines d'entrées aléatoires pour s'assurer que les lois mathématiques restent vraies pour toutes les valeurs.

```
32 # =====
33 # Stratégies Hypothesis
34 # =====
35
36 finite_floats = st.floats(
37     min_value=-1e6,
38     max_value=1e6,
39     allow_nan=False,
40     allow_infinity=False,
41 )
42
43 non_empty_float_lists = st.lists(
44     finite_floats,
45     min_size=1,
46     max_size=50,
47 )
```

# Type de tests

## Property based testing (<https://hypothesis.readthedocs.io/en/latest/>)



Tester des *invariants* : on génère des dizaines d'entrées aléatoires pour s'assurer que les lois mathématiques restent vraies pour toutes les valeurs.

```
32 # =====
33 # Stratégies Hypothesis
34 # =====
35
36 finite_floats = st.floats(
37     min_value=-1e6,
38     max_value=1e6,
39     allow_nan=False,
40     allow_infinity=False,
41 )
42
43 non_empty_float_lists = st.lists(
44     finite_floats,
45     min_size=1,
46     max_size=50,
47 )
```

```
136 @given(non_empty_float_lists)
137 def test_mean_and_std_basic_invariants(values):
138     """
139     Propriétés de base :
140     - l'écart-type est toujours  $\geq 0$ 
141     - la moyenne est (à flot près) dans l'intervalle [min, max]
142     """
143     mean, std = mean_and_std(values)
144
145     assert std  $\geq$  0.0
146
147     mn = min(values)
148     mx = max(values)
149
150     # La moyenne d'une liste finie est dans [min, max], à une petite dérive près
151     assert mn - MEAN_BOUNDS_TOL  $\leq$  mean  $\leq$  mx + MEAN_BOUNDS_TOL
152
```



# Type de tests

## Property based testing (<https://hypothesis.readthedocs.io/en/latest/>)



*Tester des invariants : on génère des dizaines d'entrées aléatoires pour s'assurer que les lois mathématiques restent vraies pour toutes les valeurs.*

```
32 # =====
33 # Stratégies Hypothesis
34 # =====
35
36 finite_floats = st.floats(
37     ... min_value=-1e6,
38     ... max_value=1e6,
39     ... allow_nan=False,
40     ... allow_infinity=False,
41 )
42
43 non_empty_float_lists = st.lists(
44     ... finite_floats,
45     ... min_size=1,
46     ... max_size=50,
47 )
```

```
154 @given(non_empty_float_lists, finite_floats)
155 def test_mean_and_std_translation_invariance(values, shift):
156     """
157     ... Propriété : translation.
158     ...  $\text{mean}(x + c) = \text{mean}(x) + c$ 
159     ...  $\text{std}(x + c) = \text{std}(x)$ 
160
161     ... On exclut les séquences constantes pour ce test, car
162     ... le cas "std ~ 0" est déjà testé séparément.
163     ... """
164     ... # évite les listes constantes (tous les éléments identiques)
165     ... assume(len(set(values)) > 1)
166
167     ... shifted = [x + shift for x in values]
168
169     ... mean1, std1 = mean_and_std(values)
170     ... mean2, std2 = mean_and_std(shifted)
171
172     ... # moyenne : tolérance relative + absolue
173     ... assert mean2 == pytest.approx(
174     ...     mean1 + shift, rel=MEAN_REL_TOL, abs=MEAN_ABS_TOL
175     ... )
176
177     ... # std : doit rester ~ la même, à une petite erreur flottante près
178     ... assert std2 == pytest.approx(
179     ...     std1, rel=STD_REL_TOL, abs=STD_ABS_TOL
180     ... )
```

# Type de tests

## Property based testing (<https://hypothesis.readthedocs.io/en/latest/>)



*Tester des invariants : on génère des dizaines d'entrées aléatoires pour s'assurer que les lois mathématiques restent vraies pour toutes les valeurs.*

```
32 # =====
33 # Stratégies Hypothesis
34 # =====
35
36 finite_floats = st.floats(
37     min_value=-1e6,
38     max_value=1e6,
39     allow_nan=False,
40     allow_infinity=False,
41 )
42
43 non_empty_float_lists = st.lists(
44     finite_floats,
45     min_size=1,
46     max_size=50,
47 )
```

```
183 @given(non_empty_float_lists, finite_floats)
184 def test_mean_and_std_scaling(values, scale):
185     """
186     ... Propriété : homothétie.
187     ...  $\text{mean}(a \cdot x) = a \cdot \text{mean}(x)$ 
188     ...  $\text{std}(a \cdot x) = |a| \cdot \text{std}(x)$ 
189
190     ... On exclut les listes constantes pour cette propriété, car on a
191     ... un test spécifique pour ce cas ( $\text{std} \sim 0$ ).
192     """
193     ... # On évite les séquences constantes pour ce test
194     ... assume(len(set(values)) > 1)
195
196     ... scaled = [scale * x for x in values]
197
198     ... mean1, std1 = mean_and_std(values)
199     ... mean2, std2 = mean_and_std(scaled)
200
201     ... assert mean2 == pytest.approx(
202     ...     scale * mean1, rel=MEAN_REL_TOL, abs=MEAN_ABS_TOL
203     ... )
204     ... assert std2 == pytest.approx(
205     ...     abs(scale) * std1, rel=STD_REL_TOL, abs=STD_ABS_TOL
206     ... )
207
```



# Aparté : Tests et flottant

```
16 # =====
17
18 # Tolérance absolue pour les comparaisons sur la moyenne (et limites min/max)
19 MEAN_ABS_TOL = 1e-9
20 # Tolérance relative pour les comparaisons de moyenne
21 MEAN_REL_TOL = 1e-9
22
23 # Tolérance absolue pour les comparaisons sur l'écart-type
24 STD_ABS_TOL = 1e-9
25 # Tolérance relative pour les comparaisons sur l'écart-type
26 STD_REL_TOL = 1e-9
27
28 # Tolérance spécifique pour "mean dans [min, max]"
29 MEAN_BOUNDS_TOL = 1e-9
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

# Aparté : Tests et flottant

Nombre flottant : Représentation en base 2 (Norme IEEE 754)  $\mathbb{F} \subsetneq \mathbb{R}$

IEEE 754 Floating Point Standard



$$\text{number} = (-1)^s * (1.m) * 2^{e-127}$$

```
In [1]: (1e20 + -1e20) + 1
```

# Aparté : Tests et flottant

Nombre flottant : Représentation en base 2 (Norme IEEE 754)  $\mathbb{F} \subsetneq \mathbb{R}$

IEEE 754 Floating Point Standard



$$\text{number} = (-1)^s * (1.m) * 2^{e-127}$$

```
In [1]: (1e20 + -1e20) + 1  
Out[1]: 1.0
```

# Aparté : Tests et flottant

Nombre flottant : Représentation en base 2 (Norme IEEE 754)  $\mathbb{F} \subsetneq \mathbb{R}$

IEEE 754 Floating Point Standard



$$\text{number} = (-1)^s * (1.m) * 2^{e-127}$$

```
In [1]: (1e20 + -1e20) + 1
Out[1]: 1.0

In [2]: 1e20 + (-1e20 + 1)
```

# Aparté : Tests et flottant

Nombre flottant : Représentation en base 2 (Norme IEEE 754)  $\mathbb{F} \subsetneq \mathbb{R}$

## IEEE 754 Floating Point Standard



$$\text{number} = (-1)^s * (1.m) * 2^{e-127}$$

```
In [1]: (1e20 + -1e20) + 1
Out[1]: 1.0

In [2]: 1e20 + (-1e20 + 1)
Out[2]: 0.0
```

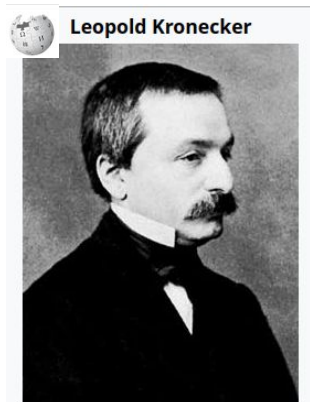
# Aparté : Tests et flottant

Nombre flottant : Représentation en base 2 (Norme IEEE 754)  $\mathbb{F} \subsetneq \mathbb{R}$

- Exemple :  $0.1 + 0.2 \neq 0.3$
- Non-associativité :  $(a + b) + c \neq a + (b + c)$
- Non-distributivité :  $a * (b + c) \neq a*b + a*c$

```
In [1]: (1e20 + -1e20) + 1
Out[1]: 1.0

In [2]: 1e20 + (-1e20 + 1)
Out[2]: 0.0
```



*Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk*  
*"God made the integers, all else is the work of man"*



<https://cc-fr.eu/webinars-replay/>  
<https://www.youtube.com/watch?v=JWe74pqFXWE>

# Aparté : Tests et flottant

---

Nombre flottant : Représentation en base 2 (Norme IEEE 754)

$$\mathbb{F} \subsetneq \mathbb{R}$$

- Exemple :  $0.1 + 0.2 \neq 0.3$
- Non-associativité :  $(a + b) + c \neq a + (b + c)$
- Non-distributivité :  $a * (b + c) \neq a*b + a*c$

Exemples d'erreur d'arrondi très connu

- Missiles Patriot, 1991 : erreur ~ 500 m après 100h d'erreur d'arrondie
- Vancouver Stock Exchange, 1982 : erreur de 52 % après deux ans, 524.811\$ à la place de 1098.892\$



• <https://cc-fr.eu/webinars-replay/>  
• <https://www.youtube.com/watch?v=JWe74pgFXWE>



# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, "*How reliable are results of computers*"]

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>

# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, “How reliable are results of computers”

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>]

float:  $P = -6.33825300e + 29$



Name	Common name	Radix	Digits <sup>[c]</sup>	Decimal digits <sup>[d]</sup>
binary32	Single precision	2	24	7.22

# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, “How reliable are results of computers”

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>]

float:  $P = -6.33825300e + 29$   
double:  $P = -1.1805916207174113e + 021$



Name	Common name	Radix	Digits <sup>[c]</sup>	Decimal digits <sup>[d]</sup>
binary32	Single precision	2	24	7.22
binary64	Double precision	2	53	15.95



# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, “How reliable are results of computers”

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>]

float:  $P = -6.33825300e + 29$   
double:  $P = -1.1805916207174113e + 021$   
long double:  $P = +5.76460752303423489188e + 17$



Name	Common name	Radix	Digits <sup>[c]</sup>	Decimal digits <sup>[d]</sup>
binary32	Single precision	2	24	7.22
binary64	Double precision	2	53	15.95

# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, “How reliable are results of computers”

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>]

float:  $P = -6.33825300e + 29$   
double:  $P = -1.1805916207174113e + 021$   
long double:  $P = +5.76460752303423489188e + 17$   
quad:  $P = +1.17260394005317863185883490452018380$



Name	Common name	Radix	Digits <sup>[c]</sup>	Decimal digits <sup>[d]</sup>
binary32	Single precision	2	24	7.22
binary64	Double precision	2	53	15.95
binary128	Quadruple precision	2	113	34.02



# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, *"How reliable are results of computers"*

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>]

float:  $P = -6.33825300e + 29$   
double:  $P = -1.1805916207174113e + 021$   
long double:  $P = +5.76460752303423489188e + 17$   
quad:  $P = +1.17260394005317863185883490452018380$   
fp16:  $P = \text{NaN}$



Name	Common name	Radix	Digits <sup>[c]</sup>	Decimal digits <sup>[d]</sup>
binary16	Half precision	2	11	3.31
binary32	Single precision	2	24	7.22
binary64	Double precision	2	53	15.95
binary128	Quadruple precision	2	113	34.02

V. Lafage (Université Paris-Saclay)



February 20th 2025



<https://cc-fr.eu/webinars-replay/>  
<https://www.youtube.com/watch?v=JWe74pqFXWE>



# Aparté : Tests et flottant

## Cursed Cancellation

⇒ higher degree polynomials can degrade high resolutions (cf RUMP)

$$P = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

with  $x = 77617$ ,  $y = 33096$  (coprime integers)

[S.M. RUMP, 1983, *"How reliable are results of computers"*

<https://www.tuhh.de/ti3/paper/rump/Ru83b.pdf>]

float:  $P = -6.33825300e + 29$   
double:  $P = -1.1805916207174113e + 021$   
long double:  $P = +5.76460752303423489188e + 17$   
quad:  $P = +1.17260394005317863185883490452018380$   
fp16:  $P = \text{NaN}$   
exact:  $P \approx -0.827396059946821368141165095479816292$   
 $P = -\frac{54767}{66192}$



Name	Common name	Radix	Digits <sup>[c]</sup>	Decimal digits <sup>[d]</sup>
binary16	Half precision	2	11	3.31
binary32	Single precision	2	24	7.22
binary64	Double precision	2	53	15.95
binary128	Quadruple precision	2	113	34.02

V. Lafage (Université Paris-Saclay)



February 20th 2025



<https://cc-fr.eu/webinars-replay/>  
<https://www.youtube.com/watch?v=JWe74pqFXWE>

# Métrique de test

---

## **Couverture de Code:**

- mesure de la **proportion du code exécutée** par les tests.



# Métrique de test

---

## Couverture de Code:

- mesure de la **proportion du code exécutée** par les tests.

Pourquoi c'est utile ?

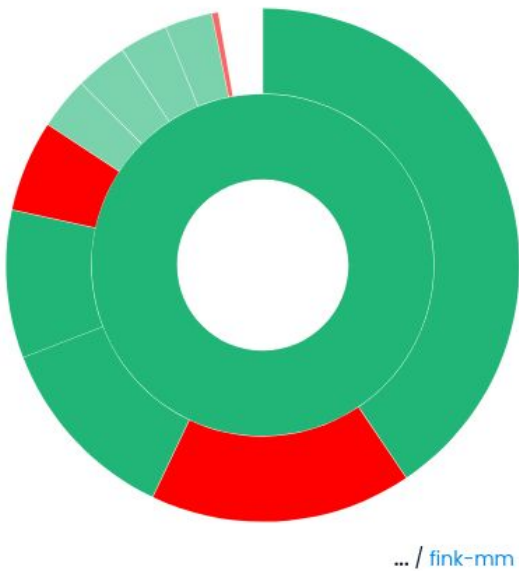
- Identifie les **zones non testées** (donc potentiellement risquées).
- Aide à **prioriser l'écriture de tests**.
- Soutient la **robustesse et maintenance** du logiciel.

```
$ pip install pytest-cov  
$ pytest --cov=mon_package
```

## Couverture de Code:

- mesure de la **proportion du code exécutée** par les tests.

Code tree	File list	fink-mm / fink_mm	Q Search for files			
Files ↑	Tracked lines	Covered	Partial	Missed	Coverage %	
..						
distribution	77	71	0	6	<div><div></div></div>	92.21%
gcn_stream	136	69	0	67	<div><div></div></div>	50.74%
observatory	363	343	0	20	<div><div></div></div>	94.49%
test	133	79	0	54	<div><div></div></div>	59.40%
utils	273	244	0	29	<div><div></div></div>	89.38%
__init__.py	3	3	0	0	<div><div></div></div>	100.00%
conftest.py	148	148	0	0	<div><div></div></div>	100.00%
fink_mm_cli.py	24	3	0	21	<div><div></div></div>	12.50%
init.py	62	56	0	6	<div><div></div></div>	90.32%
ztf_join_gcn.py	153	141	0	12	<div><div></div></div>	92.16%
Subtotal	1372	1157	0	215		



## Couverture de Code

```
144
145     elif topic in TOPICS_FORMAT["json"]:
146         try:
147             df = gr.parse_json_alert(gcn, logger, logs, is_test)
148         except Exception:
149             logger.error(
150                 "error while reading the json notice\n\n\tgcn: {}".format(gcn),
151                 exc_info=1,
152             )
153             return gcn_tracking
154
155     else:
156         logger.error(
157             "error while parsing the gcn file:\n\tttopic: {}\n\tgcn: {}".format(
158                 topic, gcn
159             )
160         )
161         raise Exception("bad gcn file format")
162
163     try:
164         if df is None:
165             if logs: # pragma: no cover
166                 logger.info("The gcn is not a real observation")
167             return gcn_tracking
168
169         triggerId = df["triggerId"].values[0]
170         timejd = df["triggerTimejd"].values[0]
171         current_id_mask = gcn_tracking["triggerId"] == triggerId
172         if current_id_mask.any():
173             current_track = gcn_tracking.loc[current_id_mask]
174             updated_tag = f"update_{current_track['nb_received'].values[0]}"
175             df["gcn_status"] = updated_tag
176             gcn_tracking.loc[current_id_mask, "nb_received"] += 1
177         else:
178             gcn_tracking.loc[len(gcn_tracking)] = [triggerId, timejd, 0]
179             updated_tag = "initial"
180             df["gcn_status"] = updated_tag
181
```

# Métrique de test

---

## Couverture de Code :

⚠ fort taux de couverture  $\neq$  tests de qualité.

- On peut avoir 95% de couverture avec des tests inutiles.
- **Couverture de code = Indicateur**

# Métrique de test

---

## Couverture de Code :

⚠ fort taux de couverture  $\neq$  tests de qualité.

- On peut avoir 95% de couverture avec des tests inutiles.
- **Couverture de code = Indicateur**

## Métrique de qualité de test

- **Mutation Testing Score**

Injection **automatique** de fautes dans le code (mutation)

- $\geq \rightarrow >$
- $+$   $\rightarrow -$
- `return x`  $\rightarrow$  `return 0`

Un test de qualité doit **échouer** sur ces mutations  $\rightarrow$  il « tue » le mutant.

# Métrique de test

---

## Couverture de Code :

⚠ fort taux de couverture  $\neq$  tests de qualité.

- On peut avoir 95% de couverture avec des tests inutiles.
- **Couverture de code = Indicateur**

## Métrique de qualité de test

- **Mutation Testing Score**

Injection **automatique** de fautes dans le code (mutation)

- $\geq \rightarrow >$
- $+$   $\rightarrow -$
- `return x`  $\rightarrow$  `return 0`

\$ pip install mutmut  
ou

\$ pip install cosmic-ray

Un test de qualité doit **échouer** sur ces mutations  $\rightarrow$  il « tue » le mutant.

# Documentation & Normes

---

Pourquoi documenter ?

- Transmettre le fonctionnement
- Réduire les erreurs et les malentendus
- Assurer la reproductibilité scientifique
- Aider les nouveaux arrivants
- Améliorer les tests

Contribue à la **robustesse**, **maintenabilité** et la **testabilité**

# Documentation & Normes

## Documentation utilisateur vs développeur

- Documentation utilisateur
  - Installation, exécution
  - Tutoriels / How-to
  - Exemples concrets

The screenshot shows the Fink documentation website. The sidebar on the left contains the following links: Home, Troubleshooting, Broker, Science Roadmap, Technological consideration, Fink science modules, Fink filters, Alert Classification, Accessing alert data, Science Portal, Cheat Sheet, Getting started (highlighted), Retrieving data, Space awareness, Solar System objects, Multi-messenger, Anomaly detection, fink-client, Data Transfer, Livestream, Xmatch, TOM Fink, Developer corner, Alert schemas, and Designing a science module. The main content area is titled 'Getting started' and contains the following text: '• ENDPOINT = <https://api.fink-portal.org/api/v1/latests> gives you access to objects based on their classification.' It then explains that 'ARGS' is a dictionary with different arguments to your query. It provides an example of accessing data for the object ID 'ZTF21aaxtctv' using the `requests` library. The code example is: 

```
import requests

# get data for ZTF21aaxtctv
r = requests.post(
    "https://api.fink-portal.org/api/v1/objects",
    json={
        "objectId": "ZTF21aaxtctv",
        "output-format": "json"
    }
)
```

 It then explains that the result is encapsulated inside `r` and can be accessed using `r.json()` and `r.content`, but for convenience you can also use Pandas to nicely format the output. The second code example is: 

```
import io
import pandas as pd

pdf = pd.read_json(io.BytesIO(r.content))
```

 It then explains that you get a DataFrame (table) whose each row is an alert emitted when the object varies, and the columns are the available fields (see the [schema page](#)). More options for this endpoint are discussed at [Search by name](#). The footer contains a link to the API documentation: <https://fink-broker.readthedocs.io/en/latest/>.



# Documentation & Normes

## Documentation utilisateur vs développeur

- Documentation développeur
  - Architecture (modules)
  - API (privée)
  - Comment contribuer

The screenshot shows the 'Alert schemas' page on the Fink documentation website. The page has a dark blue header with the Fink logo, 'Alert schemas' title, a search bar, and 'fink-broker v3.2.0+rc0' version information. A left sidebar lists navigation links: Fink filters, Alert Classification, Accessing alert data, Science Portal, Cheat Sheet, Getting started, Retrieving data, Space awareness, Solar System objects, Multi-messenger, Anomaly detection, fink-client, Data Transfer, Livestream, Xmatch, TOM Fink, Developer corner, Alert schemas (selected), Designing a science module, Designing a filter, Testing Fink, About, Source code, Release Notes, and Contributing. The main content area explains that different Fink services serve data from different sources and lists relevant schemas and data provenance per service. It features four sections: Science Portal, REST API, Livestream, and Data Transfer, each with a brief description of how they interact with the Fink Database. A note at the bottom states that Fink Science modules and filters rely on the same schema as the Livestream and Data Transfer services. The footer of the page shows 'Rubin/LSST' and a 'latest' button.

Alert schemas

The different services in Fink serve data from different sources, hence the schemas can be different from one service to another. We list here the relevant schemas and data provenance per service.

ZTF

**Science Portal**

The Science Portal relies on a REST API that queries the Fink Database, which contains aggregated data organized into column families based on data provenance. For more details, you can refer to the DB schema.

**REST API**

The REST API queries the Fink Database, which contains aggregated data organized into column families based on data provenance. For more details, you can refer to the DB schema.

**Livestream**

The Livestream service relies on the Fink Data Lake which is the concatenation of the ZTF alert schema and Fink added values.

**Data Transfer**

The Data Transfer service relies on the Fink Data Lake which is the concatenation of the ZTF alert schema and Fink added values.

Note that Fink Science modules and Fink filters rely on the same schema than the livestream and the Data Transfer services.

Rubin/LSST

<https://fink-broker.readthedocs.io/en/latest/>


# Documentation & Normes

## Standards et automatisations

### - Markdown

```
① README.md x
1  <p align="center">
2    
3  </p>
4
5  <div align="center">
6    <h1>Outfit</h1>
7
8    A fast, safe, and extensible Rust library for managing astrometric
      observations and determining preliminary orbits of small bodies. Outfit
      reads common observation formats (MPC 80-column, ADES XML, Parquet), performs
      initial orbit determination (IOD) with the Gauss method, manages
      observers (topocentric geometry), and interfaces with JPL ephemerides for
      accurate state propagation.
9    </div>
10
11  <p align="center">
12    <a href="https://crates.io/crates/outfit">
13      
14    </a>
15    <a href="https://docs.rs/outfit">
16      
17    </a>
18    <a href="LICENSE">
19      
20    </a>
21    <a href="https://github.com/FusRoman/Outfit/actions">
22      
23    </a>
24    <a href="https://codecov.io/gh/FusRoman/Outfit">
25      
26    </a>
27    <a href="https://www.rust-lang.org/">
28      
29    </a>
30  </p>
31
32  > Why Outfit?
33  > Modern asteroid pipelines need a library that is fast (Rust),
      reproducible, and easy to integrate in data-intensive workflows (batch
      files, Parquet, CI/benchmarks). Outfit re-implements classic OrbFit IOD logic
      with a memory-safe, modular design and production-grade ergonomics
      (features, docs, tests, benches). It is built to:
34  > - ingest large datasets efficiently (columnar Parquet, batch APIs);
35  > - run deterministic IOD with controlled noise and repeatable seeds;
36  > - interface cleanly with JPL ephemerides (e.g., DE440);
37  > - provide a clean API that composes well across projects.
38
39  ...
40
41  ## Table of Contents
42
43  - [Features](#features)
44  - [Installation](#installation)
45  - [Quick Start](#quick-start)
46  - [Data Formats](#data-formats)
47  - [Initial Orbit Determination](#initial-orbit-determination)
48  - [Observers & Reference Frames](#observers-reference-frames)
```

READMELicense



## Outfit

A fast, safe, and extensible Rust library for **managing astrometric observations** and **determining preliminary orbits** of small bodies. Outfit reads common observation formats (MPC 80-column, ADES XML, Parquet), performs **initial orbit determination (IOD)** with the **Gauss method**, manages observers (topocentric geometry), and interfaces with **JPL ephemerides** for accurate state propagation.

crates.io v2.0.0 docs **1.0.0** license CeCILL-C CC BY **1.0.0** codecov 100% MSRV 1.82+

### Why Outfit?

Modern asteroid pipelines need a library that is fast (Rust), reproducible, and easy to integrate in data-intensive workflows (batch files, Parquet, CI/benchmarks). Outfit re-implements classic OrbFit IOD logic with a memory-safe, modular design and production-grade ergonomics (features, docs, tests, benches). It is built to:

- ingest large datasets efficiently (columnar Parquet, batch APIs);
- run deterministic IOD with controlled noise and repeatable seeds;
- interface cleanly with JPL ephemerides (e.g., DE440);
- provide a clean API that composes well across projects.

### Table of Contents

- [Features](#)
- [Installation](#)
- [Quick Start](#)
- [Data Formats](#)
- [Initial Orbit Determination](#)
- [Observers & Reference Frames](#)
- [Cargo Feature Flags](#)
- [Performance & Reproducibility](#)
- [Roadmap](#)
- [Contributing](#)
- [License](#)
- [Acknowledgements](#)

### Features

- **Observation I/O**
  - MPC 80-column files
  - ADES XML files
  - Parquet batches for high-throughput pipelines
- **Observer management**
  - Lookup by MPC code
  - Topocentric geometry (geocentric & heliocentric positions, AU, [J2000])
- **Initial Orbit Determination**
  - Gauss method on observation triplets

## Standards et automatisisation

- Markdown (<https://squidfunk.github.io/mkdocs-material/>)

The screenshot displays the pyOutfit documentation website. The top navigation bar includes a home icon, the version '1.0.0', a search bar, and the pyOutfit logo with version 'v1.0.0' and icons for GitHub, PyPI, and a license. The left sidebar contains a table of contents with links to Home, Installation, Quickstart, Tutorials, pyOutfit Environment, IODParams, IOD from trajectories, Working with orbit results, Using pandas with pyOutfit, API, Overview, PyOutfit, Observer, IODParams, IODGauss, Orbital Elements, Observations, Trajectories, and Pandas Integration. The main content area is titled 'Installation' and describes the recommended ways to install py-outfit and set up an isolated environment. It mentions pre-built wheels for Linux x86\_64 (Python 3.12) and a Rust extension module. Below this is the 'Quick start (PyPI / pip)' section, which provides instructions for installing py-outfit using pip and verifying the installation with a Python command. The expected output is the class representation. A second code block shows how to check the version using importlib.metadata. The 'Using PDM (recommended for reproducible workflows)' section explains that PDM manages isolated environments and keeps metadata in pyproject.toml. It lists two steps: 1. Install PDM (user-level) using pip install --upgrade pdm, and 2. Initialize a new project directory (or reuse an existing one) using pdm init.

Home  
Installation  
Quickstart  
Tutorials  
pyOutfit Environment  
IODParams  
IOD from trajectories  
Working with orbit results  
Using pandas with pyOutfit  
API  
Overview  
PyOutfit  
Observer  
IODParams  
IODGauss  
Orbital Elements  
Observations  
Trajectories  
Pandas Integration

### Installation

This page describes the recommended ways to install `py-outfit` and set up an isolated environment. The package ships pre-built wheels for Linux x86\_64 (Python 3.12) and embeds a Rust extension module compiled from the Outfit core. If a wheel is not available for your platform, a local build from source will be attempted (Rust toolchain required).

### Quick start (PyPI / pip)

If you already have a clean Python 3.12 environment (e.g. `venv` or `virtualenv`):

```
pip install --upgrade pip
pip install py-outfit
```

Verify your installation:

```
python -c "import py_outfit as o; print(o.KeplerianElements)"
```

Expected output is the class representation (not an error). You can also check the version:

```
python -c "import importlib.metadata as m; print(m.version('py-outfit'))"
```

### Using PDM (recommended for reproducible workflows)

PDM manages isolated environments and keeps metadata in `pyproject.toml`.

1. Install PDM (user-level):

```
pip install --upgrade pdm
```

2. Initialize a new project directory (or reuse an existing one):

```
pdm init
```

Table of contents  
Quick start (PyPI / pip)  
Using PDM (recommended for reproducible workflows)  
Using Conda / Mamba  
Source build (fallback)  
Verifying functionality  
Selecting a parallel strategy  
Upgrading  
Uninstalling  
Troubleshooting  
Next steps

# Documentation & Normes

## Standards et automatisisation

- Markdown
- Type hints + docstring

```
149 def mean_and_std(values: Iterable[float]) -> tuple[float, float]:
150     """
151     ... Calcule la moyenne et l'écart-type (population) d'une séquence de réels.
152     ... La moyenne arithmétique est définie par :
153     ...
154     ...
155     ...
156     ... \\bar{x} = \\frac{1}{N} \\sum_{i=1}^N x_i
157     ...
158     ...
159     ... et l'écart-type (définition *population*) par :
160     ...
161     ...
162     ... \\sigma = \\sqrt{\\frac{1}{N} \\sum_{i=1}^N (x_i - \\bar{x})^2}
163     ...
164     ...
165     ... Parameters
166     ...
167     ... values : Iterable[float]
168     ...     Séquence de valeurs numériques (par exemple une liste ou un
169     ...     générateur de flottants).
170     ...
171     ... Returns
172     ...
173     ... mean : float
174     ...     Moyenne arithmétique des valeurs.
175     ... std : float
176     ...     Écart-type (population) des valeurs.
177     ...
178     ... Raises
179     ...
180     ... ValueError
181     ...     Si la séquence fournie est vide.
182     ...
183     ... data = list(values)
184     ... if not data:
```

mean\_and\_std ↗

```
mean_and_std(values: Iterable[float]) -> tuple[float, float]
```

Calcule la moyenne et l'écart-type (population) d'une séquence de réels.

La moyenne arithmétique est définie par :

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

et l'écart-type (définition *population*) par :

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}$$

PARAMETER	DESCRIPTION
values	Séquence de valeurs numériques (par exemple une liste ou un générateur de flottants). <b>TYPE:</b> Iterable[float]

RETURNS	DESCRIPTION
mean	Moyenne arithmétique des valeurs. <b>TYPE:</b> float
std	Écart-type (population) des valeurs. <b>TYPE:</b> float

RAISES	DESCRIPTION
ValueError	Si la séquence fournie est vide.

# Documentation & Normes

## Standards et automatisation

- Markdown
- Type hints + docstring + tests

```
$ pytest --doctest-modules good.py -vv -x
=====
platform linux -- Python 3.12.10, pytest-9.0.1, pluggy-1.6.0 -
cachedir: .pytest_cache
hypothesis profile 'default'
rootdir: /home/roman/Documents/Work/Mission/Roscoff_11_2025
configfile: pyproject.toml
plugins: hypothesis-6.148.2
collected 1 item

good.py::good.mean_and_std PASSED
=====
```

```
189
190 ---Exemples
191 -----
192 ---Exemple simple avec quelques valeurs :
193
194 ---->>> mean, std = mean_and_std([1.0, 2.0, 3.0])
195 ---->>> round(mean, 6)
196 ----2.0
197 ---->>> round(std, 6)
198 ----0.816497
199
200 ---Fonctionne aussi avec un générateur :
201
202 ---->>> data = (float(i) for i in range(4)) # 0, 1, 2, 3
203 ---->>> mean, std = mean_and_std(data)
204 ---->>> round(mean, 6)
205 ----1.5
206 ---->>> round(std, 6)
207 ----1.118034
208
209 ---Une séquence constante donne un écart-type nul :
210
211 ---->>> mean, std = mean_and_std([5.0, 5.0, 5.0])
212 ---->>> mean
213 ----5.0
214 ---->>> std
215 ----0.0
216
217 ---Une liste vide provoque une erreur :
218
219 ---->>> mean_and_std([])
220 ---Traceback (most recent call last):
221 ----|
222 ---ValueError: Impossible de calculer des statistiques sur une liste vide.
223 ---"""
```

# Documentation & Normes

## Standards et automatisatisation

- Markdown
- Type hints + docstring + tests

```
$ pytest --doctest-modules good.py -vv -x
=====
platform linux -- Python 3.12.10, pytest-9.0.1, pluggy-1.6.0 -
cachedir: .pytest_cache
hypothesis profile 'default'
rootdir: /home/roman/Documents/Work/Mission/Roscoff_11_2025
configfile: pyproject.toml
plugins: hypothesis-6.148.2
collected 1 item

good.py::good.mean_and_std PASSED
=====
```

### Exemples:

Exemple simple avec quelques valeurs :

```
>>> mean, std = mean_and_std([1.0, 2.0, 3.0])
>>> round(mean, 6)
2.0
>>> round(std, 6)
0.816497
```

Fonctionne aussi avec un générateur :

```
>>> data = (float(i) for i in range(4)) # 0, 1, 2, 3
>>> mean, std = mean_and_std(data)
>>> round(mean, 6)
1.5
>>> round(std, 6)
1.118034
```

Une séquence constante donne un écart-type nul :

```
>>> mean, std = mean_and_std([5.0, 5.0, 5.0])
>>> mean
5.0
>>> std
0.0
```

Une liste vide provoque une erreur :

```
>>> mean_and_std([])
Traceback (most recent call last):
...
ValueError: Impossible de calculer des statistiques sur une liste vide.
```

# Documentation & Normes

## Standards et automatisisation

- Markdown
- Type hints + docstring
- OpenApi/Swagger (<https://swagger.io/specification/>)

**Fink/ZTF object API** <sup>2.3.0</sup>  
[ Base URL: / ]  
[Swagger JSON](#)  
REST API to access data from Fink

**api/v1/objects** Get object data based on ZTF ID

**POST** /api/v1/objects Retrieve object data from the Fink/ZTF database based on their name

Parameters Try it out

Name	Description
<b>payload</b> <sup>required</sup> object (body)	<div>Example Value   Model</div> <pre>{  "objectId": "ZTF21afmfmix",  "withupperlim": false,  "withcutouts": false,  "cutout-kind": "Science",  "columns": "i:jd;i:magpsf,i:fid",  "output-format": "json"}</pre> <div>Parameter content type application/json</div>
objectId string (query)	single ZTF Object ID, or a comma-separated list of object names, e.g. "ZTF19acmdpyr,ZTF21aaxctcv" <input type="text" value="objectId"/>
withupperlim string (query)	If True, retrieve also upper limit measurements, and bad quality measurements. Use the column <b>d:tag</b> in your results: valid, upperlim, badquality. <input type="text" value="withupperlim"/>
withcutouts string (query)	If True, retrieve also cutout data as 2D array. See also <b>cutout-kind</b> . More information on the original cutouts at <a href="https://irsa.ipac.caltech.edu/data/ZTF/docs/ztf_explanatory_supplement.pdf">https://irsa.ipac.caltech.edu/data/ZTF/docs/ztf_explanatory_supplement.pdf</a> <input type="text" value="withcutouts"/>
cutout-kind string (query)	<b>Science</b> , <b>Template</b> , or <b>Difference</b> . If not specified, returned all three. <input type="text" value="cutout-kind"/>
columns string (query)	Comma-separated data columns to transfer, e.g. "i:magpsf,i:jd". If not specified, transfer all columns. <input type="text" value="columns"/>
output-format string (query)	Output format among json(default), csv, parquet, votable. <input type="text" value="output-format"/>

Responses Response content type application/json

# Documentation & Normes

---

## Normes et Conventions

*rendre le code lisible, cohérent et compréhensible par tous*



# Documentation & Normes

---

## Normes et Conventions

*rendre le code lisible, cohérent et compréhensible par tous*

- Standard de style
  - PEP8 (Python)
  - style, indentation, imports, nommage, lisibilité

# Documentation & Normes

---

## Normes et Conventions

*rendre le code lisible, cohérent et compréhensible par tous*

- Standard de style
  - PEP8 (Python)
  - style, indentation, imports, nommage, lisibilité

```
1  import sys,os
2
3  def add(a,b):return a+b
4
5  def main():
6      result=add(1,2)
7      print("Result is:",result)
8
9  if __name__=="__main__":
10     main()
```

# Documentation & Normes

## Normes et Conventions

*rendre le code lisible, cohérent et compréhensible par tous*

- Standard de style
  - PEP8 (Python)
  - style, indentation, imports, nommage, lisibilité

```
1 import sys,os
2
3 def add(a,b):return a+b
4
5 def main():
6     result=add(1,2)
7     print("Result is:",result)
8
9 if __name__=="__main__":
10     main()
```

```
$ flake8 bad_format.py
bad_format.py:1:1: F401 'sys' imported but unused
bad_format.py:1:1: F401 'os' imported but unused
bad_format.py:1:7: E271 multiple spaces after keyword
bad_format.py:1:12: E401 multiple imports on one line
bad_format.py:1:12: E231 missing whitespace after ','
bad_format.py:3:1: E302 expected 2 blank lines, found 1
bad_format.py:3:4: E271 multiple spaces after keyword
bad_format.py:3:12: E231 missing whitespace after ','
bad_format.py:3:15: E231 missing whitespace after ':'
bad_format.py:5:1: E302 expected 2 blank lines, found 1
bad_format.py:6:11: E225 missing whitespace around operator
bad_format.py:6:17: E231 missing whitespace after ','
bad_format.py:7:11: E201 whitespace after '('
bad_format.py:7:26: E231 missing whitespace after ','
bad_format.py:7:35: E202 whitespace before ')'
bad_format.py:9:1: E305 expected 2 blank lines after class or function definition, found 1
bad_format.py:9:12: E225 missing whitespace around operator
bad_format.py:9:24: E203 whitespace before ':'
bad_format.py:10:2: E111 indentation is not a multiple of 4
```

# Documentation & Normes

## Normes et Conventions

*rendre le code lisible, cohérent et compréhensible par tous*

- Standard de style
  - PEP8 (Python)
  - style, indentation, imports, nommage, lisibilité

```
1 import sys,os
2
3 def add(a,b):return a+b
4
5 def main():
6     result=add(1,2)
7     print("Result is:",result)
8
9 if __name__=="__main__":
10     main()
```

```
1  def add(a,b):
2      return a+b
3
4
5  def main():
6      result=add(1,2)
7      print(f"Result is:{result}")
8
9
10 if __name__=="__main__":
11     main()
```

\$ flake8 good\_format.py  
\$

# Documentation & Normes

---

## Normes et Conventions

*rendre le code lisible, cohérent et compréhensible par tous*

- Standard de style
  - PEP8 (Python)
  - style, indentation, imports, nommage, lisibilité
- Outils automatique
  - **Linters** : pylint, flake8... → détectent erreurs, incohérences, complexité inutile.
  - **Formateurs** : black, isort → formatent automatiquement le code selon les conventions.

# Ce qu'on a...

---

- Une architecture logicielle
  - Modules & interfaces
- Des tests
  - Unitaire
  - Integration
- Une documentation
  - Utilisateur
  - Développeur

# Ce qui manque

---

- Une architecture logicielle
  - Modules & interfaces
- Des tests
  - Unitaire
  - Integration
- Une documentation
  - Utilisateur
  - Développeur
- **Traçabilité des changements**
- **Travail collaboratif**
- **Automatisation**
- **Publication**

# Gestion de versions

---

## Pourquoi versionner ?

Problématique:

- Comment tracer les changements ?
- Comment revenir en arrière après un bug ?
- Comment maintenir une base stable tout en développant des changements ?
- Comment travailler en équipe ?
- Qui a changé quoi ?
- Comment reproduire des résultats ?



# Gestion de versions

---

## Un outil: **Git**

- **Historique complet** des modifications  
comprendre l'évolution, diagnostiquer un bug
- **Sécurité**  
pouvoir revenir à un état sain à n'importe quel moment
- **Expérimentation** sans risque  
branches indépendantes, essais, prototypes
- **Collaboration** structurée  
éviter les conflits, revoir le code, fusionner proprement
- **Reproductibilité** scientifique  
associer un résultat à une version exacte (avec un tag)

# Gestion de versions



- Gestion de l'historique

proto.py

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ git init
```

astuce: Utilisation de 'master' comme nom de la branche initiale. Le nom de la branche  
astuce: par défaut peut changer. Pour configurer le nom de la branche initiale  
astuce: pour tous les nouveaux dépôts, et supprimer cet avertissement, lancez :

astuce:

```
astuce: git config --global init.defaultBranch <nom>
```

astuce:

astuce: Les noms les plus utilisés à la place de 'master' sont 'main', 'trunk' et  
astuce: 'development'. La branche nouvellement créée peut être renommée avec :

astuce:

```
astuce: git branch -m <nom>
```

Dépôt Git vide initialisé dans /home/roman/Documents/Work/Mission/Roscoff\_11\_2025/detmes/.git/

```
$ git status
```

Sur la branche master

Aucun commit

rien à valider (créez/copiez des fichiers et utilisez "git add" pour les suivre)

```
$
```

# Gestion de versions



- Gestion de l'historique

proto.py

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ git status
Sur la branche master

Aucun commit

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    proto.py

aucune modification ajoutée à la validation mais des fichiers non suivis sont
$ git add proto.py
$ git status
Sur la branche master

Aucun commit

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)
    nouveau fichier : proto.py

$
```

# Gestion de versions



- Gestion de l'historique

proto.py

```
1 import math, sys
2
3 def run(x):
4     r = []
5     for v in x:
6         if v is None or v == "":
7             continue
8         v = float(v)
9         v = (v * 0.00123) + 0.42
10        if v < 0:
11            v = 0
12        r.append(v)
13    s = 0
14    for v in r:
15        s += v
16    m = s / len(r)
17    s2 = 0
18    for v in r:
19        s2 += (v - m) ** 2
20    s2 = math.sqrt(s2 / len(r))
21    if s2 > 0.1:
22        print("ALERT", m, s2)
23    else:
24        print("OK", m, s2)
25
26 if __name__ == "__main__":
27     data = []
28     for a in sys.argv[1:]:
29         data.append(a)
30     run(data)
31
```

```
$ git commit -m "prototype de la chaine de traitement"
[master (commit racine) c6e71f6] prototype de la chaine de traitement
1 file changed, 30 insertions(+)
create mode 100644 proto.py
$ git status
Sur la branche master
rien à valider, la copie de travail est propre
$ git log
commit c6e71f6cba32451d87cd72b6dfef40bc4c0bdf8b (HEAD -> master)
Author: Roman <roman.lemontagner@gmail.com>
Date:   Fri Nov 21 14:31:40 2025 +0100

    prototype de la chaine de traitement
$
```

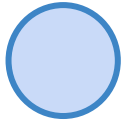
# Gestion de versions

---



- Gestion de l'historique

e71229



Prototype de la  
chaîne de  
traitement

# Gestion de versions



- Gestion de l'historique



Commande :

\$ **git init** (initialise le repo git)

\$ **git status** (check l'état du repo)

\$ **git add** (ajout des modifications à la validation)

\$ **git commit** (valide les modifications)

# Gestion de versions



- Gestion de l'historique

```
1 from dataclasses import dataclass
2
3 #==== Modèle de donnée ====
4
5
6 @dataclass
7 class RawSample:
8     """
9     Représente une mesure brute lue depuis la chaîne d'acquisition.
10
11     Parameters
12     -----
13     value: float
14     Valeur numérique brute, typiquement lue en unités ADU
15     (Analog-to-Digital Units).
16     """
17
18     value: float
19
20
21 @dataclass
22 class CalibratedSample:
23     """
24     Représente une mesure convertie en unités physiques (volts).
25
26     Parameters
27     -----
28     volts: float
29     Valeur de la mesure après calibration, exprimée en volts.
30     """
31
32     volts: float
```

```
$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
src/
```

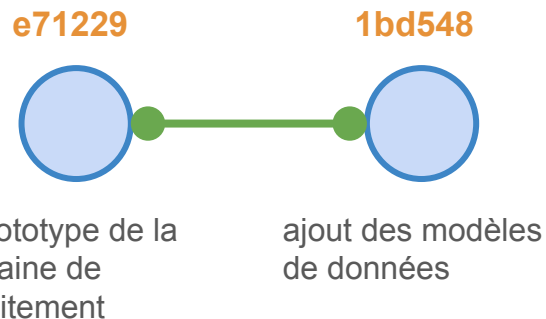
```
aucune modification ajoutée à la validation mais des fichiers non suivis
$ git add src
$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    nouveau fichier : src/__init__.py
    nouveau fichier : src/modele.py
```

```
$ git commit -am "ajout des modèles de données"
[master 0e6e614] ajout des modèles de données
 2 files changed, 32 insertions(+)
 create mode 100644 src/__init__.py
 create mode 100644 src/modele.py
$ git status
Sur la branche master
rien à valider, la copie de travail est propre
$
```

# Gestion de versions



- Gestion de l'historique





# Gestion de versions



# git

- Gestion de l'historique

```
$ rm proto.py
$ git status
Sur la branche master
Modifications qui ne seront pas validées :
  (utilisez "git add/rm <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de
    supprimé :      proto.py

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    src/calibration.py
    src/input.py
    src/stats.py

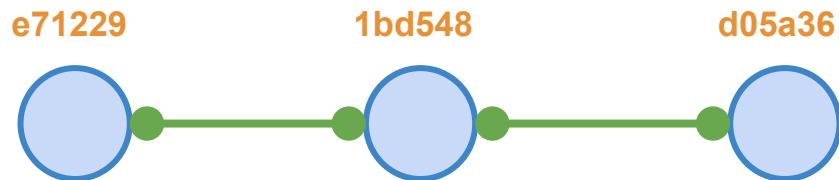
aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
$ git add proto.py src/
$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    supprimé :      proto.py
    nouveau fichier : src/calibration.py
    nouveau fichier : src/input.py
    nouveau fichier : src/stats.py

$ git commit -m "suppression de proto.py, ajout du reste de la chaine de traitement"
[master 6648350] suppression de proto.py, ajout du reste de la chaine de traitement
4 files changed, 135 insertions(+), 30 deletions(-)
delete mode 100644 proto.py
create mode 100644 src/calibration.py
create mode 100644 src/input.py
create mode 100644 src/stats.py
$ git status
Sur la branche master
rien à valider, la copie de travail est propre
$
```

# Gestion de versions



- Gestion de l'historique



Prototype de la  
chaine de  
traitement

ajout des modèles  
de données

suppression de  
proto.py, ajout du  
reste de la chaine  
de traitement

```
$ git log
commit 664835022a1d5bf5ba8834736002ac59086c24ea (HEAD -> master)
Author: Roman <roman.lemontagner@gmail.com>
Date:   Fri Nov 21 14:50:43 2025 +0100

    suppression de proto.py, ajout du reste de la chaine de traitement

commit 89aba1746ebfda0277877d49a20698507c4f36fc
Author: Roman <roman.lemontagner@gmail.com>
Date:   Fri Nov 21 14:47:21 2025 +0100

    ajout des modèles de données

commit 6b86786e4bb39d28c6153d1c0201fac0a4d6cc9d
Author: Roman <roman.lemontagner@gmail.com>
Date:   Fri Nov 21 14:46:40 2025 +0100

    prototype de la chaine de traitement
$
```

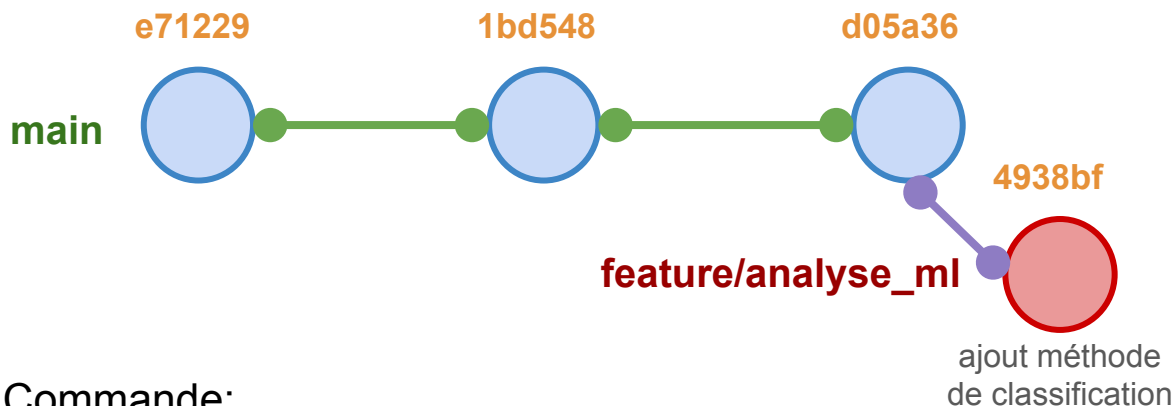
⚠ Ne jamais laisser de fichier inutile  
dans un repo git → **Historique**

# Gestion de versions



- Expérimentation / Ajout

Gestion de branche : travailler sans casser la version stable



Commande:

```
$ git switch -c feature/analyse_ml
```

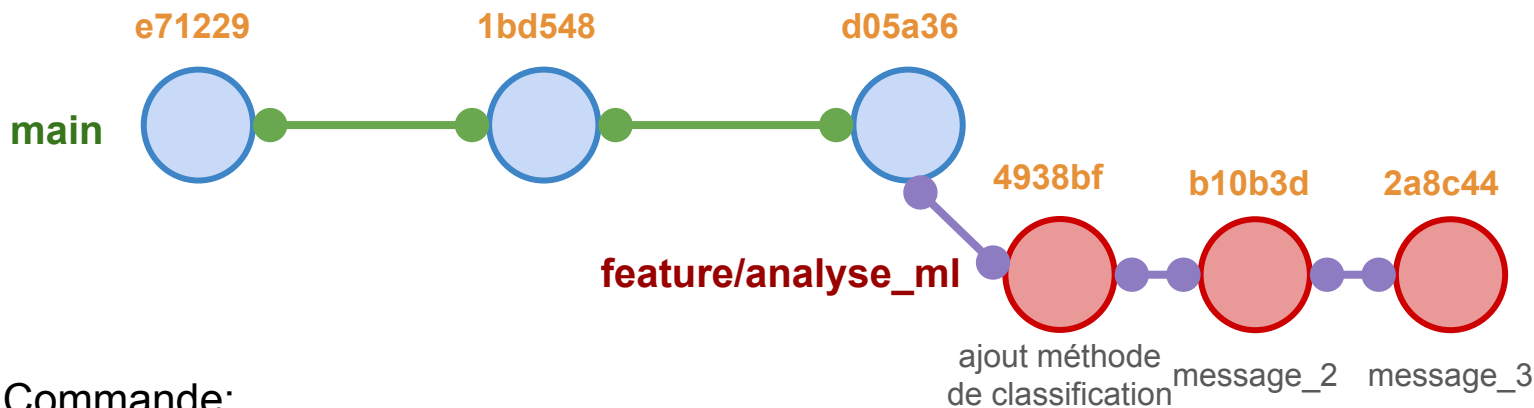
```
$ git commit -m "ajout méthode de classification"
```

# Gestion de versions



- **Expérimentation / Ajout**

**Gestion de branche : travailler sans casser la version stable**



Commande:

```
$ git commit -m "message_2"
```

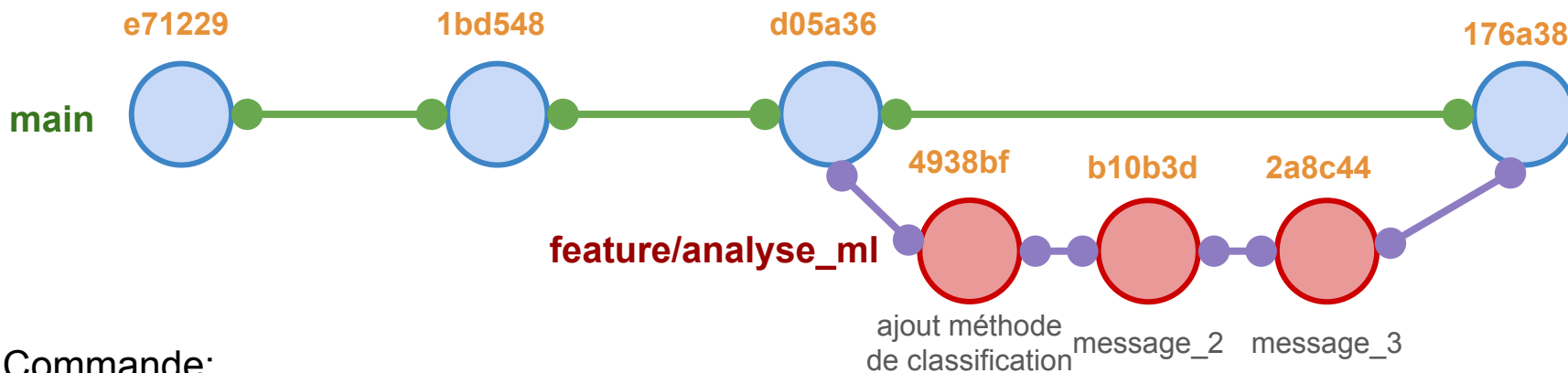
```
$ git commit -m "message_3"
```

# Gestion de versions



- Expérimentation / Ajout

**Gestion de branche : travailler sans casser la version stable**



Commande:

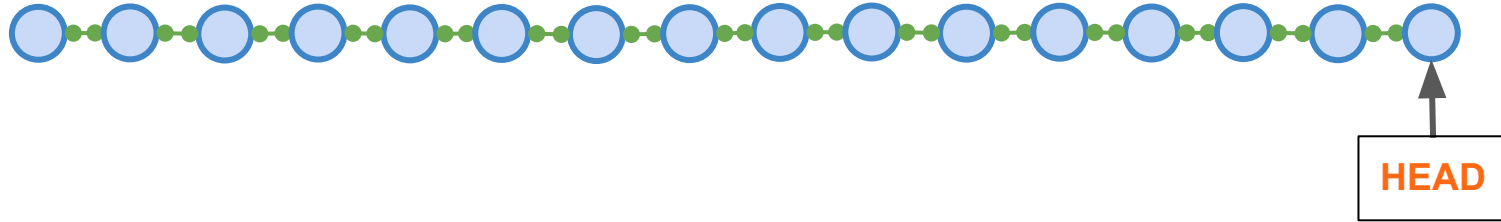
```
$ git switch main
```

```
$ git merge feature/analyse_ml
```

# Gestion de versions



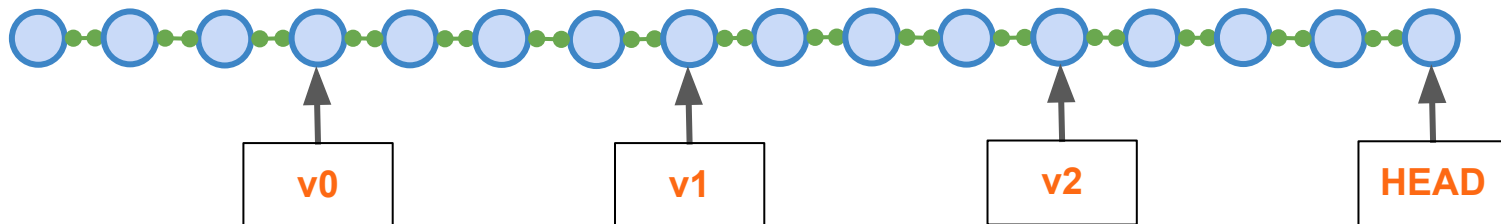
- Versionning (Tag)



# Gestion de versions



- Versionning (Tag)

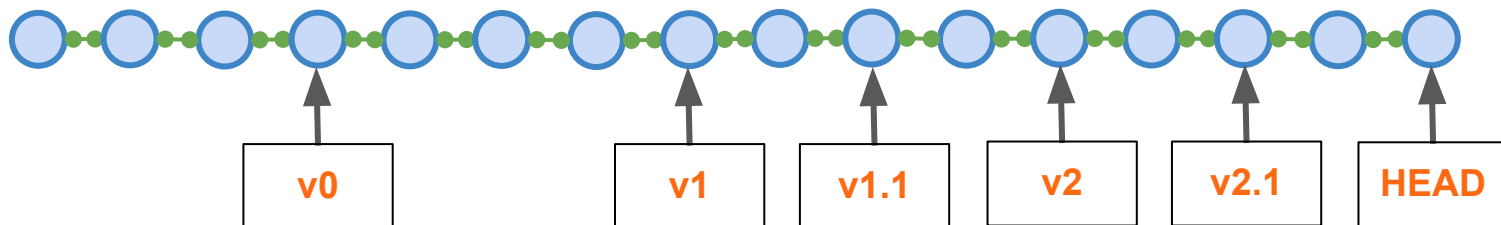


Une version (numéro et/ou nom) = une photo d'un logiciel

# Gestion de versions



- Versionning (Tag)



- Versionnement sémantique: MAJEUR.MINEUR.CORRECTIF
  - Majeur : changement non rétrocompatible
  - Mineur : changement rétrocompatible
  - Correctif : correction d'anomalies rétrocompatible

<https://semver.org/lang/fr/>



# Gestion de versions

---



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main

# Gestion de versions

---



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



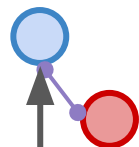
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



main

release v1

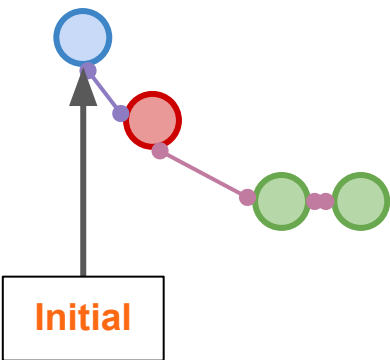
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



main

release v1

feature/f1

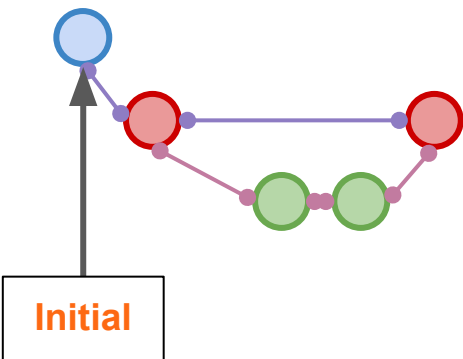
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



main

release v1

feature/f1

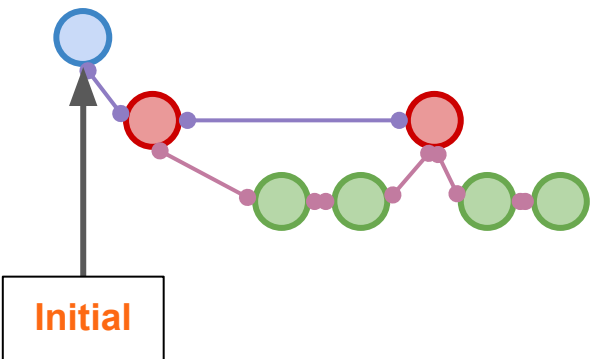
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



main

release v1

feature/f2

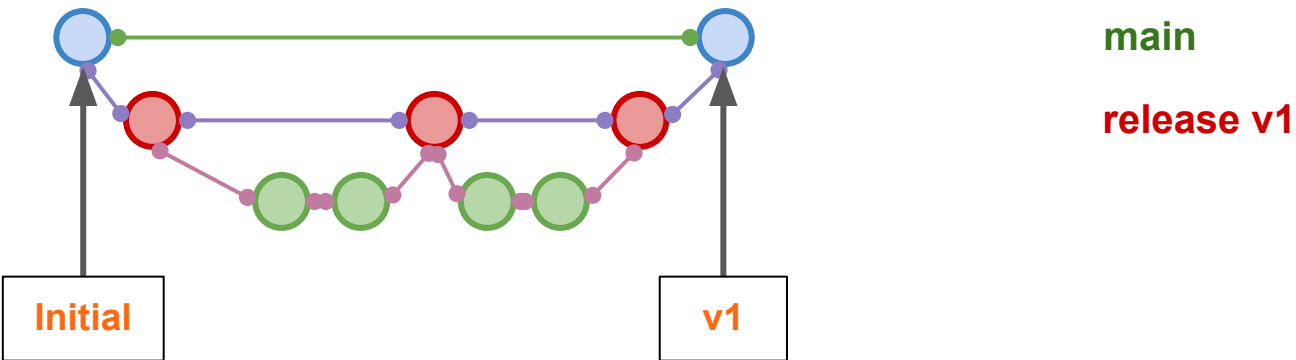
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



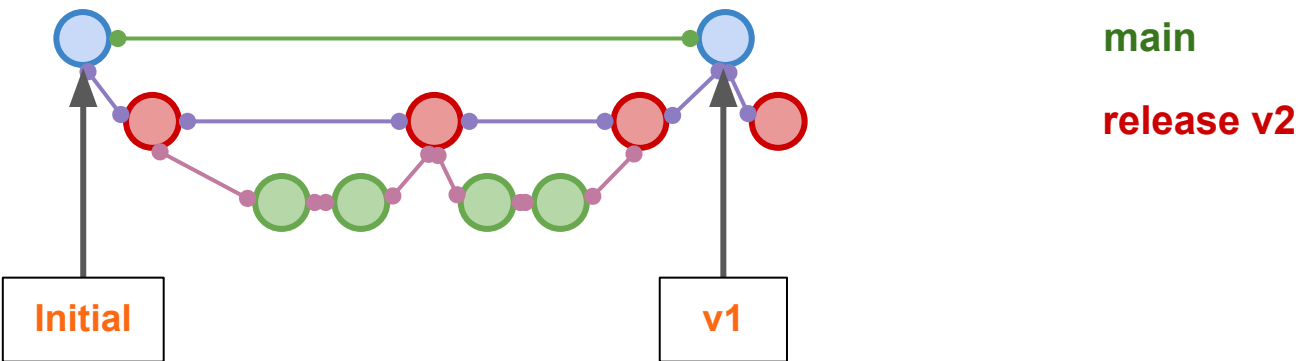
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)





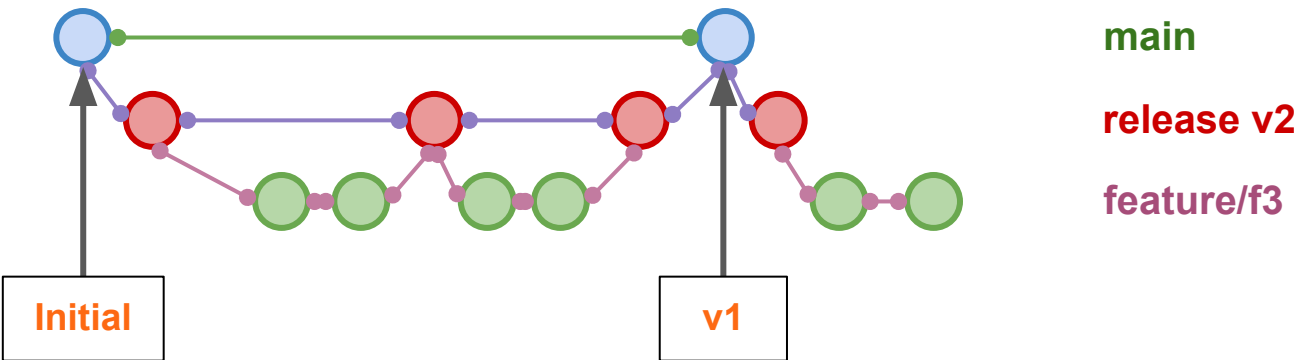
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



main

release v2

feature/f3

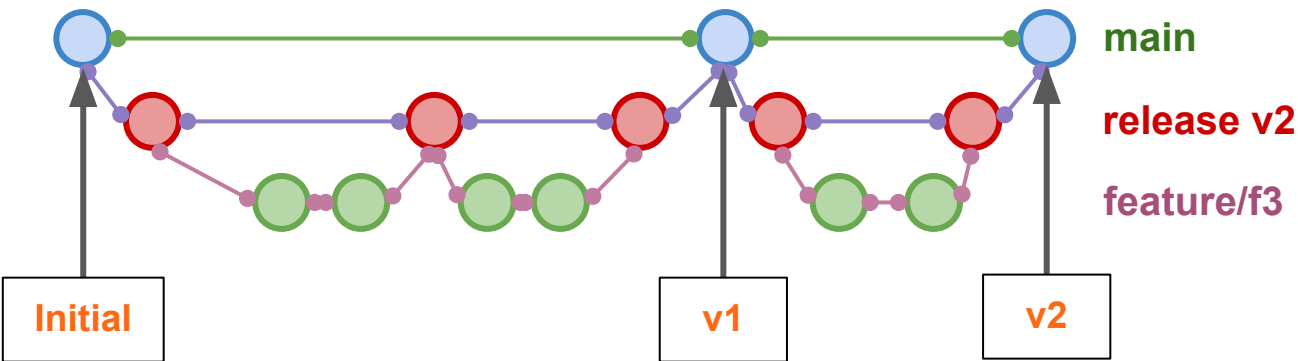
# Gestion de versions



- **Branche, Versionning, Release**

Quelque bonnes pratiques :

- On ne commit **jamais** dans la branche main
- Organisation en release de version (branche release, branche feature)



# Gestion de versions

---



quelque commandes :

- git init: initialise le repo
- git add : ajoute des modifs
- git commit : valide les modifs
- git switch : changer de branche
  - (ajout de l'option '-c' pour créer une branche)
- git merge : fusionner une branche
- git status : checker l'état du repo
- git log : voir l'historique

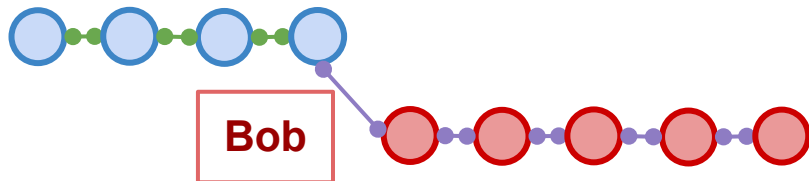
Conseil : Utiliser git même si je suis seul et sur un petit projet

# Gestion de versions

## Git + forge

- Travail collaboratif

**Gestion de branche : travailler en équipe**



**release**

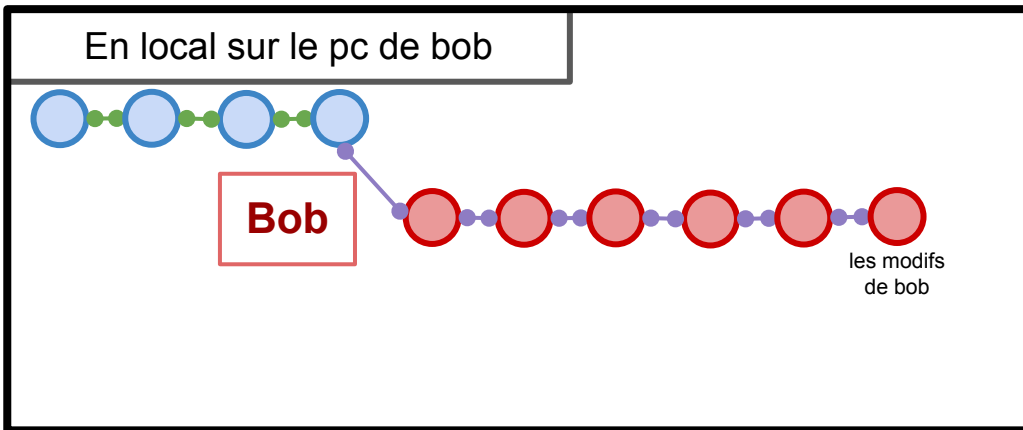
**feature/f\_bob**

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

Commande:

```
$ git add "bob_fichier.py"
```

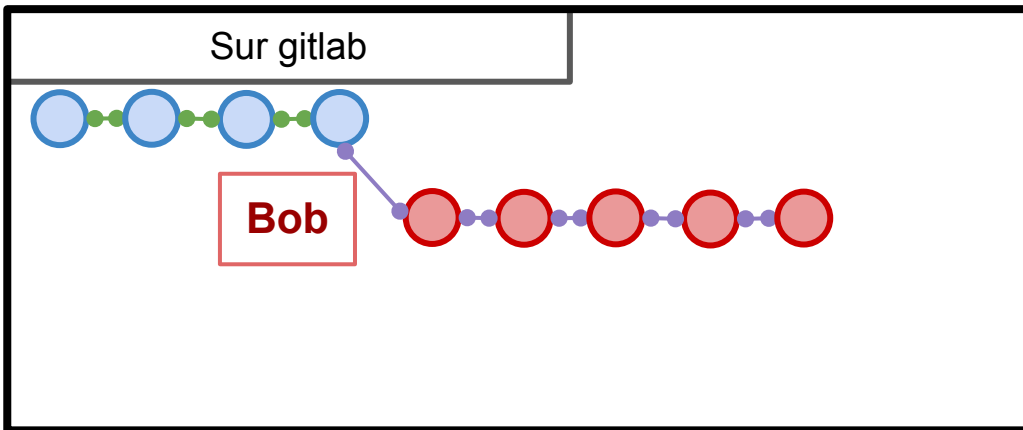
```
$ git commit -m "les mods de bob"
```

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

Commande:

```
$ git add "bob_fichier.py"
```

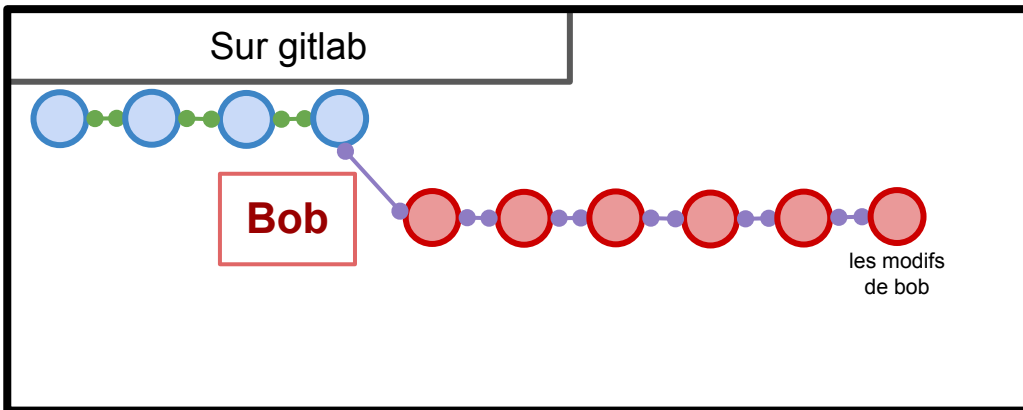
```
$ git commit -m "les mods de bob"
```

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

Commande:

```
$ git add "bob_fichier.py"
```

```
$ git commit -m "les modifs de bob"
```

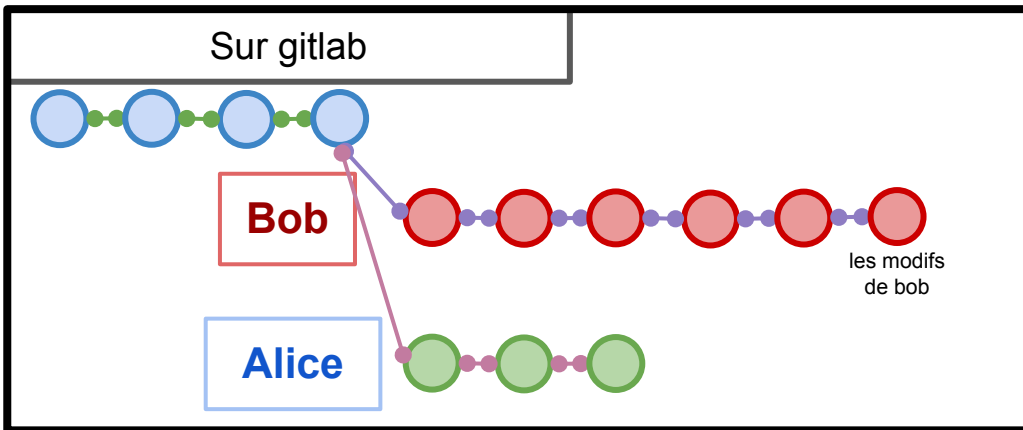
```
$ git push
```

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

feature/f\_alice

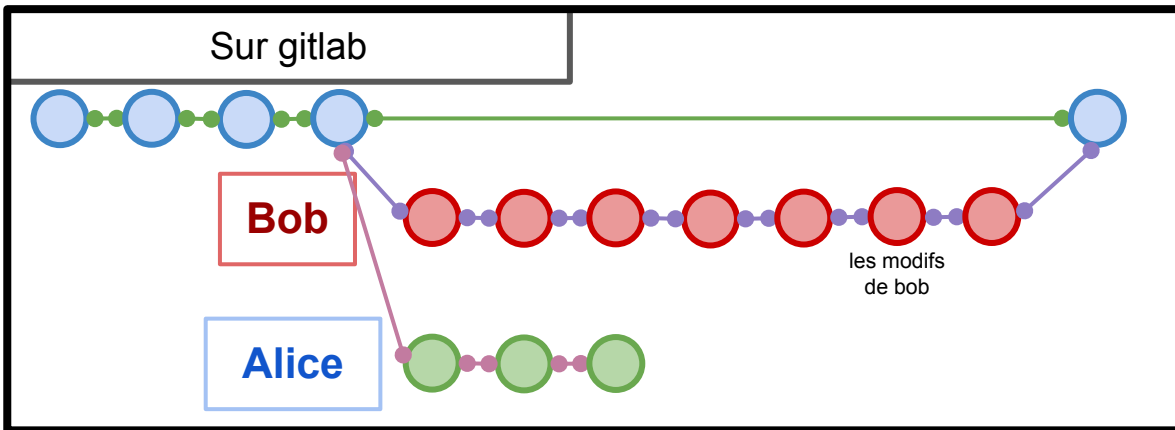


# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

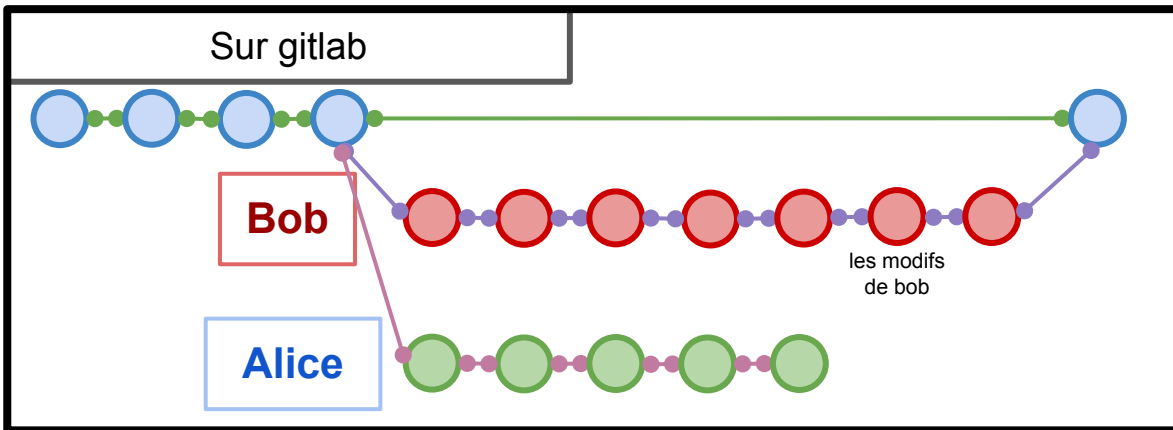
feature/f\_alice

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

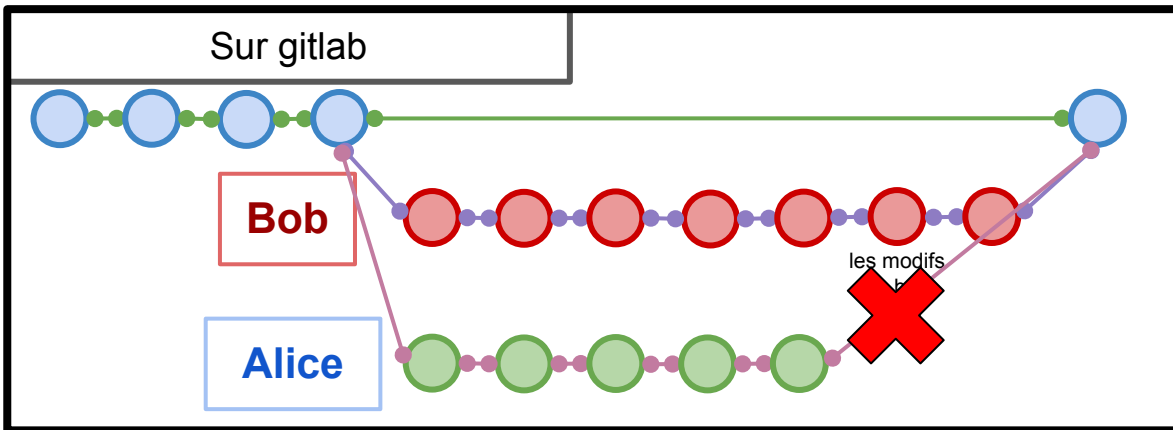
feature/f\_alice

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

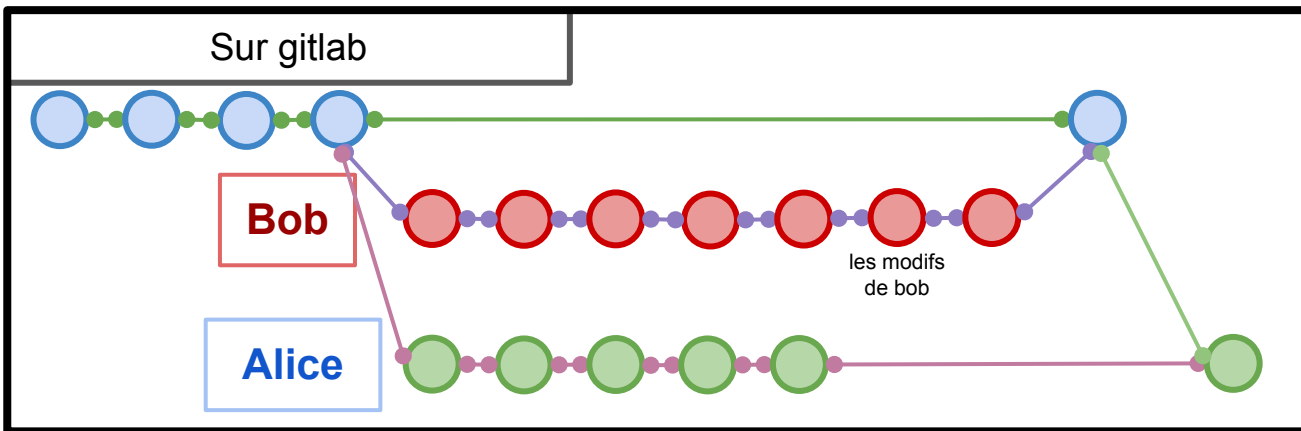
feature/f\_alice

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

feature/f\_alice

Commande:

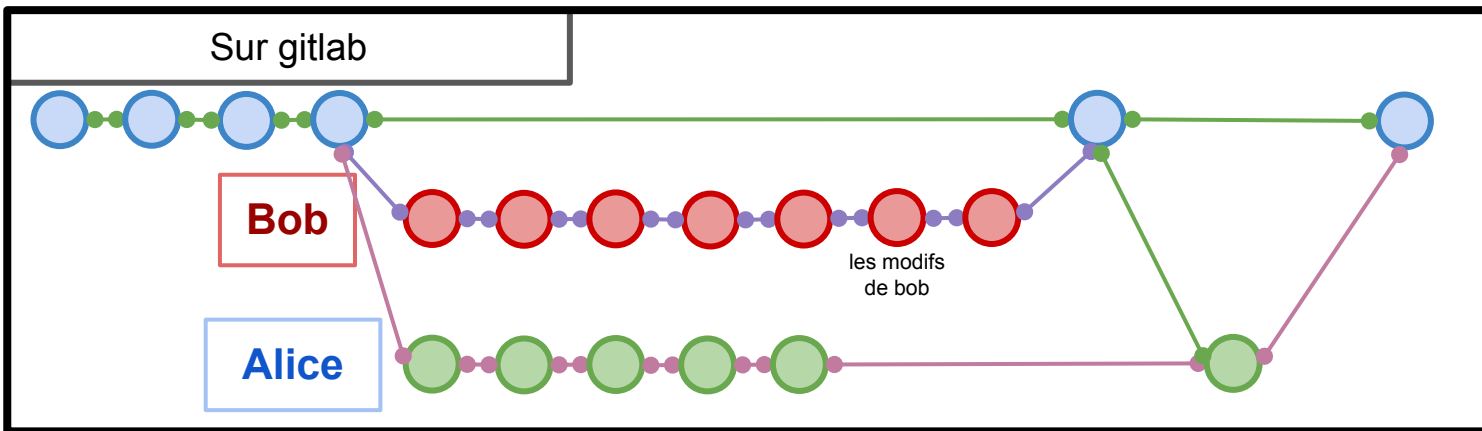
\$ **git pull**

# Gestion de versions

## Git + forge

- Travail collaboratif

### Gestion de branche : travailler en équipe



release

feature/f\_bob

feature/f\_alice

Commande:  
\$ **git push**

## Versionning / Release

The screenshot shows the GitHub repository for **fink-broker**. The top navigation bar includes links for Discussions, Actions, Projects, Security (4,880), Insights, and Settings. Below this, there are buttons for Releases and Tags, and a "Draft a new release" button. The main content area displays the **v3.2.0-rc0** release, marked as "Latest". A "What's Changed" section lists various updates, including switching to a self-hosted runner, updating processing for anomaly detection, and adding features to HBase tables. The release date is listed as Jan 17, and the release manager is @fjammes.

This screenshot shows a modal window titled "Switch branches/tags" over the **fink-broker** repository. The modal has a search bar and two tabs: "Branches" and "Tags". Under the "Branches" tab, a list of branches is shown, with **v3.2.0-rc0** selected. Other branches include **v3.1.3-rc1**, **v3.1.2-rc1**, **v3.1.1-rc1**, **v3.1**, **issues/661-v2.7**, **brb**, and **865-ci-science-ok**. A "View all tags" button is at the bottom of the modal.

The logos for **GitLab** and **GitHub** are displayed side-by-side.

**New Contributors**

- [@Knispel2](#) made their first contribution in [#929](#)

**Full Changelog:** [v3.1...v3.2.0-rc0](#)

**Contributors**

fjammes, anaismoller, and 3 other contributors

**Assets** 2

- [Source code \(zip\)](#)
- [Source code \(tar.gz\)](#)

## Gestion de ticket



astrolabsoftware / fink-broker

Q Type to search

<> Code Issues 39 Pull requests 4 Discussions Actions Projects Security 4,879 Insights Settings

is:issue state:open

Open 39 Closed 552

Author Labels Projects Milestones Assignees Types Newest

- ☐ [SSO] change RA to deg in the output xml **SSOFT**  
#1065 · JulienPeloton opened 2 days ago 4.0 1
- ☐ [Rubin] add a merge step **merge** **Rubin**  
#1063 · JulienPeloton opened 3 days ago 1
- ☐ [Rubin] Add hostless module **hostless** **Rubin** **science**  
#1062 · JulienPeloton opened 3 days ago
- ☐ Potential Fink improvement - TNS integration  
#1056 · ZacharyLane1204 opened 2 weeks ago 2
- ☐ [SSOFT] ephemerides are valid only for asteroids **bug** **SSOFT**  
#1055 · JulienPeloton opened 2 weeks ago 4.0
- ☐ [ML] Install light tensorflow **dependencies**  
**Task** #1052 · JulienPeloton opened 2 weeks ago 4.0
- ☐ [ZTF] add VAST filters **redistribution**  
#1051 · JulienPeloton opened 2 weeks ago 4.0
- ☐ Update SLSN bot format

## Revue de Code / Fusion de branche



The screenshot shows a GitLab pull request titled "Change hourangle to deg #1066". The pull request is open and shows a diff for the file `bin/ztf/archive_slsn_candidates.py`. The diff shows changes to the `SDSS_photoz` function, including a new `try` block for querying the SDSS server and a new `base_url` variable. The pull request is created by JulienPeloton and is linked to issue #1065. The interface includes tabs for Conversation, Commits, Checks, and Files. A comment from JulienPeloton states: "IMPORTANT: Please create an issue first before opening a Pull Request. Linked to issue(s): Closes #1065". Another comment from JulienPeloton asks: "What changes were proposed in this pull request?". A third comment from JulienPeloton asks: "How was this patch tested?". The pull request is marked as resolved by erusseil.

```
90 + def SDSS_photoz(ra, dec, radius=0.2):
91 +     """Retrieve photoz from SDSS"""
92 +     try:
93 +         query = f"""
94 +         SELECT TOP 1 p.objID, p.ra, p.dec, z.z AS photoz, z.zErr AS photozErr
95 +         FROM PhotoObj AS p
96 +         JOIN Photoz AS z ON p.objID = z.objID
97 +         JOIN dbo.fGetNearbyObjEq({ra}, {dec}, {radius}) AS n
98 +         ON p.objID = n.objID
99 +         ORDER BY n.distance
100 +     """
101 +
102 +     base_url = "https://skyserver.sdss.org/dr16/SkyServerWS/SearchTools/SqlSearch"
103 +     params = {"cmd": query, "format": "json"}
104 +
105 +     url = f"{base_url}?{urllib.parse.urlencode(params)}"
106 +     response = requests.get(url)
```







## Revue de Code / Fusion de branche







 **Open** Change hourangle to deg #1066  
JulienPeloton wants to merge 2 commits into `master` from `issue/1065/xml` 

[See analysis details on SonarQube Cloud](#)










 **Some checks were not successful**  
2 failing, 1 cancelled, 4 skipped, 9 successful checks

3 failing checks ▾

-   e2e: noscience, gha / call-workflow-passing-data / Build image (push) Failing after 1m ...
-   e2e: noscience, gha / call-workflow-passing-data / Run integration tests (hdfs) (pull\_request) C... ...
-   e2e: noscience, gha / call-workflow-passing-data / Run integration tests (s3) (pull\_request) Faili... ...

4 skipped checks ▾

-   e2e: noscience, gha / call-workflow-passing-data / Analyze image (push) Skipped 2 days ago ...
-   e2e: noscience, gha / call-workflow-passing-data / Push fink-broker image to IN2P3 registry (pu... ...
-   e2e: noscience, gha / call-workflow-passing-data / Push fink-broker image to IN2P3 registry (pu... ...

 **No conflicts with base branch**  
Merging can be performed automatically.

Merge pull request ▾ You can also merge this with the command line. [View command line instructions.](#)



# Integration Continue (CI)

---

Maintenir un niveau de qualité constant au court du temps

- détection précoce des erreurs
- automatisation

# Integration Continue (CI)

---

Maintenir un niveau de qualité constant au court du temps

- détection précoce des erreurs
- automatisation

*CI = exécution automatique de tâches à chaque modification (commit/PR)*

# Integration Continue (CI)

---

Maintenir un niveau de qualité constant au court du temps

- détection précoce des erreurs
- automatisation

*CI = exécution automatique de tâches à chaque modification (commit/PR)*

Que peut t'on faire dans une CI ?

→ A peu près tout

# Integration Continue (CI)

---

Maintenir un niveau de qualité constant au court du temps

- détection précoce des erreurs
- automatisation

*CI = exécution automatique de tâches à chaque modification (commit/PR)*

Que peut t'on faire dans une CI ?

Mais en général

- Build
- Tests (couverture de code)
- lint

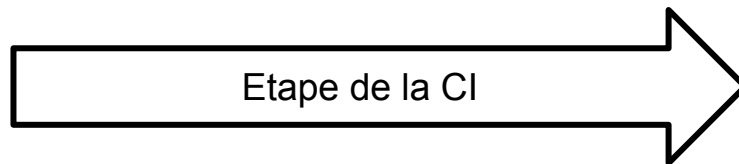
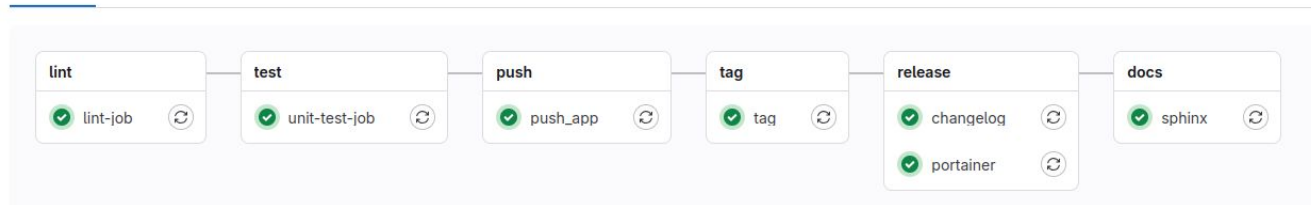
# Integration Continue (CI)

✓ Passed Roman Le Montagner created pipeline for commit 2eeb500f 6 months ago, finished 6 months ago

For integration

branch 7 jobs 29 minutes 27 seconds, queued for 1 seconds

Pipeline Jobs 7 Tests 0



- Execution sur **machine isolé**
- Chaque étape **indépendante**
- **S'arrête** dès qu'une étape échoue
- Résultat **visible** par tous

# Integration Continue (CI)

```
.gitlab-ci.yml x
.gitlab-ci.yml > {} lint > [ ] script
gitlab-ci - GitLab CI Configuration File (ci.json)
stages:
  - lint
  - test

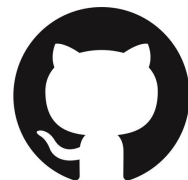
# Image officielle Python, simple et reproductible
image: python:3.11-slim

before_script:
  - python -m pip install --upgrade pip
  - pip install -r requirements.txt
  - pip install pytest pytest-cov black flake8 mypy

lint:
  stage: lint
  script:
    - black --check src/
    - flake8 src/
    - mypy src/

tests:
  stage: test
  script:
    - pytest --cov=src --cov-report=xml --cov-report=term
  artifacts:
    paths:
      - coverage.xml
    expire_in: 1 week
```

```
python-ci.yml x
python-ci.yml > {} jobs > {} build > [ ] steps > {} 4 > ru
1 name: Python CI
2
3 on:
4   push:
5   pull_request:
6
7 jobs:
8   build:
9     runs-on: ubuntu-latest
10
11     steps:
12       - name: Checkout repo
13         uses: actions/checkout@v4
14
15       - name: Set up Python
16         uses: actions/setup-python@v5
17         with:
18           python-version: "3.11"
19
20       - name: Install dependencies
21         run: |
22           python -m pip install --upgrade pip
23           pip install -r requirements.txt
24           pip install pytest pytest-cov black flake8 mypy
25
26       - name: Lint
27         run: |
28           black --check src/
29           flake8 src/
30           mypy src/
31
32       - name: Run tests
33         run: |
34           pytest --cov=src --cov-report=xml --cov-report=term
```



# Integration Continue / Déploiement Continue (CI/CD)

---

*CI = exécution automatique de tâches à chaque modification (commit/PR)*

*CD = déploiement automatique d'une version validée après la CI*

Publication de :

- packages Python
- images Docker
- documentation
- applications web (API/Flask...)

Réduit les erreurs humaines et accélère les cycles



# Integration Continue / Déploiement Continue (CI/CD)

---

*CI = exécution automatique de tâches à chaque modification (commit/PR)*

*CD = déploiement automatique d'une version validée après la CI*

Réduit les erreurs humaines et **accélère les cycles**

- Système en deux temps
  - Plateforme intégration
  - Plateforme production

Exemple:

- Intégration@Mésocentre UPSaclay - VirtualData
- Production@CC-IN2P3



# Packaging & Publication

---

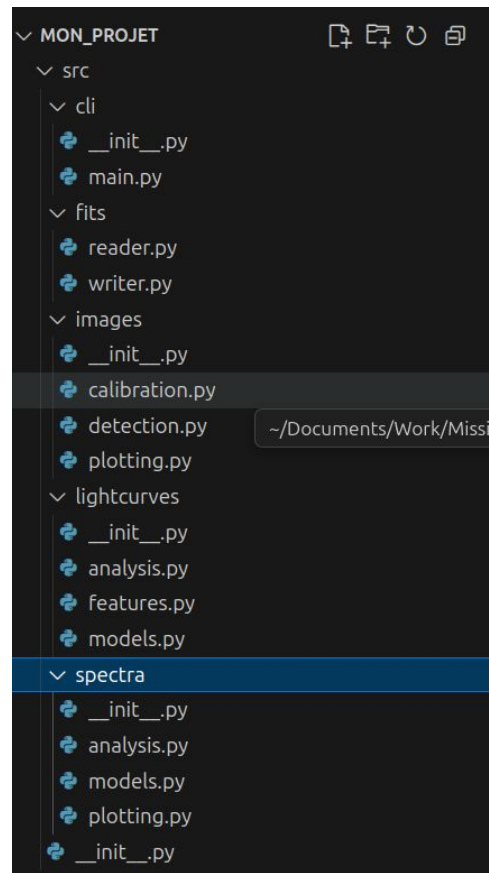
Pourquoi packager un logiciel ?

- Installation : ``pip install mon_paquet``
- Partage : Pypi, Github release...

# Packaging & Publication

Que contient un paquet ?

- le code (modules / paquets)



# Packaging & Publication

Que contient un paquet ?

- le code (modules / paquets)
- un manifeste (pyproject.toml)

```
pyproject.toml > {} project > [ ] dependencies
1  [project]
2  name = "grandma-gcn"
3  version = "9.2.0"
4  description = "Automated ingestion, analysis, and distribution of gravitational wave (GW)
5  authors = [{ name = "Roman", email = "roman.lemontagner@gmail.com" }]
6  dependencies = [
7      "gcn-kafka>=0.3.3",
8      "tomli>=2.0.2",
9      "pytz>=2025.2",
10     "slack-sdk>=3.35.0",
11     "fink-utils>=0.41.0",
12     "typing-extensions>=4.13.2",
13     "astropy>=7.0.2",
14     "astropy-healpix>=1.1.2",
15     "celery[redis]>=5.5.2",
16     "dotenv>=0.9.9",
17     "gwemopt @ git+https://github.com/FusRoman/old_gwemopt@v1.3.1",
18     "ligo-skymap>=2.4.0",
19     "yarl>=1.20.0",
20     "spherical-geometry>=1.3.3",
21     "sqlalchemy>=2.0.41",
22     "psycpg[binary]>=3.2.9",
23     "alembic>=1.16.2",
24     "python-dotenv>=1.1.1",
25     "flask>=3.1.1",
26     "gunicorn>=23.0.0",
27 ]
28 requires-python = ">=3.12"
29 readme = "README.md"
30 license = { text = "CeCILL-C" }
31
32 [build-system]
33 requires = ["pdm-backend"]
34 build-backend = "pdm.backend"
35
36
37 [tool.pdm]
38 distribution = true
39
```

# Packaging & Publication

Que contient un paquet ?

- le code (modules / paquets)
- un manifeste (pyproject.toml)

Métadonnées indispensables:

- Nom du package
- version, description courte.
- Long description (README).
- Dépendances.
- Auteur(s)
- License

```
pyproject.toml > {} project > [ ] dependencies
1  [project]
2  name = "grandma-gcn"
3  version = "9.2.0"
4  description = "Automated ingestion, analysis, and distribution of gravitational wave (GW)
5  authors = [{ name = "Roman", email = "roman.lemontagner@gmail.com" }]
6  dependencies = [
7      "gcn-kafka>=0.3.3",
8      "tomli>=2.0.2",
9      "pytz>=2025.2",
10     "slack-sdk>=3.35.0",
11     "fink-utils>=0.41.0",
12     "typing-extensions>=4.13.2",
13     "astropy>=7.0.2",
14     "astropy-healpix>=1.1.2",
15     "celery[redis]>=5.5.2",
16     "dotenv>=0.9.9",
17     "gwemopt @ git+https://github.com/FusRoman/old_gwemopt@v1.3.1",
18     "ligo-skymap>=2.4.0",
19     "yarl>=1.20.0",
20     "spherical-geometry>=1.3.3",
21     "sqlalchemy>=2.0.41",
22     "psycpg[binary]>=3.2.9",
23     "alembic>=1.16.2",
24     "python-dotenv>=1.1.1",
25     "flask>=3.1.1",
26     "gunicorn>=23.0.0",
27 ]
28 requires-python = ">=3.12"
29 readme = "README.md"
30 license = { text = "CeCILL-C" }
31
32 [build-system]
33 requires = ["pdm-backend"]
34 build-backend = "pdm.backend"
35
36
37 [tool.pdm]
38 distribution = true
39
```

# Packaging & Publication

Que contient un paquet ?

- le code (modules / paquets)
- un manifeste (pyproject.toml)
- une licence

```
LICENSE
1
2 CeCILL-C FREE SOFTWARE LICENSE AGREEMENT [Chat Ctrl+L Edit Ctrl+I]
3
4
5 ... Notice
6
7 This Agreement is a Free Software license agreement that is the result
8 of discussions between its authors in order to ensure compliance with
9 the two main principles guiding its drafting:
10
11 ...* firstly, compliance with the principles governing the distribution
12 ...of Free Software: access to source code, broad rights granted to
13 ...users,
14 ...* secondly, the election of a governing law, French law, with which
15 ...it is conformant, both as regards the law of torts and
16 ...intellectual property law, and the protection that it offers to
17 ...both authors and holders of the economic rights over software.
18
19 The authors of the CeCILL-C (for Ce[a] C[nrs] I[nria] L[ogiciel] L[ibre])
20 license are:
21
22 Commissariat à l'Energie Atomique - CEA, a public scientific, technical
23 and industrial research establishment, having its principal place of
24 business at 25 rue Leblanc, immeuble Le Ponant D, 75015 Paris, France.
25
26 Centre National de la Recherche Scientifique - CNRS, a public scientific
27 and technological establishment, having its principal place of business
28 at 3 rue Michel-Ange, 75794 Paris cedex 16, France.
29
30 Institut National de Recherche en Informatique et en Automatique -
31 INRIA, a public scientific and technological establishment, having its
32 principal place of business at Domaine de Voluceau, Rocquencourt, BP
33 105, 78153 Le Chesnay cedex, France.
34
35
36 ... Preamble
37
38 The purpose of this Free Software license agreement is to grant users
```

# Packaging & Publication

## Que contient un paquet ?

- le code (modules / paquets)
- un manifeste (pyproject.toml)
- une licence
- les tests
- la documentation

## Publication (Pypi)

py-outfit 1.0.0

pip install py-outfit

✓ Dernière version

Dernière version : 25 sept. 2025

Python bindings for Outfit (Rust) — Gauss IOD, ephemerides, etc.

Navigation

Description du projet

Historique des versions

Téléchargement des fichiers

Détails vérifiés

These details have been verified by PyPI

Maintenu par

LM

FusRoman

Détails non vérifiés

Ces détails n'ont pas été vérifiés par PyPI

Métadonnées

Licence : CeCILL-C Free Software License Agreement (CECILL-C)

Créé par : FusRoman

astronomy , orbit-determination , rust , pyo3 , maturin

Nécessite : Python >=3.12

Classifieurs

Development Status

5 - Production/Stable

Intended Audience

Description du projet

pyOutfit

High-performance Python bindings for the Outfit orbit-determination engine (Initial Orbit Determination, observation ingestion, orbital element conversions & batch processing) powered by Rust + PyO3.

pyOutfit

Python binding tests passing pyo3 v1.0.0 docs GitHub Pages python 3.12 build maturin license CeCILL-C

Upstream Outfit (Rust core)

crates.io v2.1.0 docs passing rust 1.82.0

Overview

pyOutfit exposes the Rust Outfit crate to Python with a thin, typed interface. It enables:

Gauss-based Initial Orbit Determination (IOD) with configurable numerical & physical filters.

Manipulation of multiple orbital element representations (Keplerian, Equinoctial, Cometary).

Efficient ingest of astrometric observations (single trajectories or large batches) with zero-copy / single-conversion paths.

Parallel batch processing for thousands of trajectories (opt-in).

Access & registration of observatories (MPC code lookup & custom definitions).

Rust performs all heavy numerical work; Python orchestrates workflows with minimal overhead.

Feature Highlights

Area	Highlights
IOD	Gauss method with configurable solver tolerances & physical filters

Novembre 2025

DETMES - Informatique des expériences

154

# Packaging & Publication

---

Que contient un paquet ?

- le code (modules / paquets)
- un manifeste (pyproject.toml)
- une licence
- les tests
- la documentation

Outils standard de gestion de paquet

- Poetry et PDM



# Packaging & Publication

---

Que contient un paquet ?

- le code (modules / paquets)
- un manifeste (pyproject.toml)
- une licence
- les tests
- la documentation

Outils standard de gestion de paquet

- Poetry et **PDM**



- Gestion de dépendances
- build automatique
- Environnement local au paquet

# Reproductibilité

---

Pourquoi la reproductibilité ?

- Pouvoir refaire l'expérience et obtenir les mêmes résultats.
- Faciliter la vérification, la correction et l'extension des travaux.
- Réduire l'effet "résultats miracles".
- Améliorer la fiabilité des pipelines numériques.

# Reproductibilité

---

Comment garantir la reproductibilité ?

Appliquer ce qu'on a vu dans ce cours

- Bonne architecture logicielle
- Tests (automatisé CI)
- **Gestion de version**
  - **Code et donnée**
- Packaging & Publication

# Reproductibilité

Comment garantir la reproductibilité ?

Appliquer ce qu'on a vu dans ce cours

- Bonne architecture logicielle
- Tests (automatisé CI)
- **Gestion de version**
  - Code et donnée
- Packaging & Publication

Conteneurisation (Docker/Singularity)



- CI/CD
- cloud
- microservice



SINGULARITYCE

- HPC
- No root
- Pipeline reproductible

## Comment garantir la reproductibilité ?

```
1  # Image de base Ubuntu
2  FROM ubuntu:22.04
3
4  # Mise à jour et installation de Python + pip
5  RUN apt-get update && apt-get install -y \
6      ... python3 \
7      ... python3-pip \
8      ... && rm -rf /var/lib/apt/lists/*
9
10 # Installation d'un paquet Python d'exemple (requests)
11 RUN pip3 install requests
12
13 # Répertoire de travail
14 WORKDIR /app
15
16 # Copier ton projet dans le conteneur (optionnel)
17 COPY . /app
18
19 # Commande par défaut : lance un shell
20 CMD ["bash"]
21
```

## Conteneurisation (Docker/Singularity)

- Un fichier recette de cuisine (Dockerfile)
  - OS (Linux, MacOS, Windows)
  - Dépendances système
  - Paquet python
- Execution rapide
  - docker build
  - docker run
- Plus léger qu'une VM
  - VM O(GB)
  - Conteneur O(MB)

# Reproductibilité

---

Une note sur le calcul flottant (Norme IEEE 754)

- Garantie de résultat déterministe et unique
  - $+$ ,  $-$ ,  $*$ ,  $/$
  - racine carré
  - FMA (Fused Multiply-Add :  $a + (b * c)$  )

# Reproductibilité

## Une note sur le calcul flottant (Norme IEEE 754)

- Garantie de résultat déterministe et unique
  - $+$ ,  $-$ ,  $*$ ,  $/$
  - racine carré
  - FMA (Fused Multiply-Add :  $a + (b * c)$  )
- **Aucune garantie pour tout le reste**
  - fonction transcendante

- $e^x$ ,  $2^x$ ,  $10^x$
- $e^x - 1$ ,  $2^x - 1$ ,  $10^x - 1$
- $\ln x$ ,  $\log_2 x$ ,  $\log_{10} x$
- $\ln(1+x)$ ,  $\log_2(1+x)$ ,  $\log_{10}(1+x)$
- $\sqrt{x^2 + y^2}$
- $1/\sqrt{x}$
- $(1+x)^n$  for  $x \geq -1$  (named *compound* and used to compute an [exponential growth](#), whose rate cannot be less than  $-1$ )<sup>[50]</sup>
- $x^{\frac{1}{n}}$
- $x^n$ ,  $x^y$
- $\sin x$ ,  $\cos x$ ,  $\tan x$
- $\arcsin x$ ,  $\arccos x$ ,  $\arctan x$ ,  $\operatorname{atan2}(y, x)$
- $\sin \pi x = \sin \pi x$ ,  $\cos \pi x = \cos \pi x$ ,  $\tan \pi x = \tan \pi x$  (see also: [Multiples of  \$\pi\$](#) )
- $\operatorname{asin} \pi x = \frac{1}{\pi} \arcsin x$ ,  $\operatorname{acos} \pi x = \frac{1}{\pi} \arccos x$ ,  $\operatorname{atan} \pi x = \frac{1}{\pi} \arctan x$ ,  
 $\operatorname{atan2} \pi(y, x) = \frac{1}{\pi} \operatorname{atan2}(y, x)$  (see also: [Multiples of  \$\pi\$](#) )
- $\sinh x$ ,  $\cosh x$ ,  $\tanh x$
- $\operatorname{arsinh} x$ ,  $\operatorname{arcosh} x$ ,  $\operatorname{artanh} x$

# reproductibilité → Science ouverte

---

Informatique repose sur des codes, des données, des pipelines, des modèles et des infrastructures

- **Open Source** : code, simulateur, pipeline, notebooks
- **Open Data** : format de donnée ouvert, public, métadonnées complètes (provenance, calibration...)
- **Open methods / Open workflows** : Jupyter, DAG versionnés
- **Open access** : publications accessibles et lien vers forges (github, gitlab ouvert...)



# Reproducibilité → Science ouverte

## Initiative pour des données de qualité

- FAIR (Findable, Accessible, Interoperable, Reusable)

The screenshot shows the pandas API reference website. The top navigation bar includes the pandas logo, links for 'Getting started', 'User Guide', 'API reference' (which is highlighted), 'Development', and 'Release notes'. A search bar with 'Search' and 'Ctrl + K' shortcuts is on the right, along with a version selector set to '2.3 (stable)' and social media icons. The left sidebar lists the 'Input/output' section, which is expanded to show a list of methods: `pandas.read_pickle`, `pandas.DataFrame.to_pickle`, `pandas.read_table`, `pandas.read_csv`, `pandas.DataFrame.to_csv`, `pandas.read_fwf`, `pandas.read_clipboard`, `pandas.DataFrame.to_clipboard`, `pandas.read_excel`, `pandas.DataFrame.to_excel`, `pandas.ExcelFile`, `pandas.ExcelFile.book`, `pandas.ExcelFile.sheet_names`, `pandas.ExcelFile.parse`, and `pandas.io.formats.style.Styler.to_excel`. The main content area is titled 'Input/output' and contains a sub-section for 'Pickling'. It lists two methods: `read_pickle` (filepath\_or\_buffer[, ...]) described as 'Load pickled pandas object (or any object) from file.', and `DataFrame.to_pickle` (path, \*, compression, ...) described as 'Pickle (serialize) object to file.'. Below this is a 'Flat file' section with `read_table` (filepath\_or\_buffer, \*, sep, ...) described as 'Read general delimited file into DataFrame.', and `read_csv` (filepath\_or\_buffer, \*, sep, ...) described as 'Read a comma-separated values (csv) file into DataFrame.'. The right sidebar, titled 'On this page', lists various file formats: Pickling, Flat file, Clipboard, Excel, JSON, HTML, XML, LaTeX, HDFStore: PyTables (HDF5), Feather, Parquet, ORC, SAS, SPSS, SQL, Google BigQuery, and STATA.

# Reproducibilité → Science ouverte

---

## Initiative pour la science ouverte

- FAIR (Findable, Accessible, Interoperable, Reusable)

- ⚠ FAIR ≠ Open

- ZENODO (<https://zenodo.org/>)

- développé par le CERN



- archive durable de production scientifique (*jeux de données, logiciels / dépôts Git, notebooks, papiers, rapports, posters, slides, etc.*)
  - DOI permanent et de version

# Conclusion

---

La conception d'un logiciel s'approche de celle d'un instrument

- Prototype
- Version stable
- Tests
- Normes et standard

On a aussi des outils

- Git, PDM, CI/CD

# Conclusion

Des outils puissants mais dangereux

- **Large Language Model (LLM)**
- A utiliser quand on sait ce qu'on fait
  - Code / Algorithmes (~)
  - Tests ✓
  - Refactoring ✓
  - Documentation ✓
  - écriture de CI ✓

