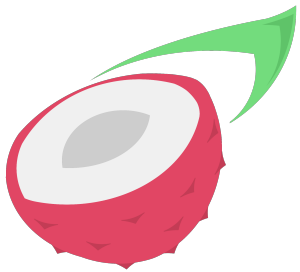


Litchi Pi

Python

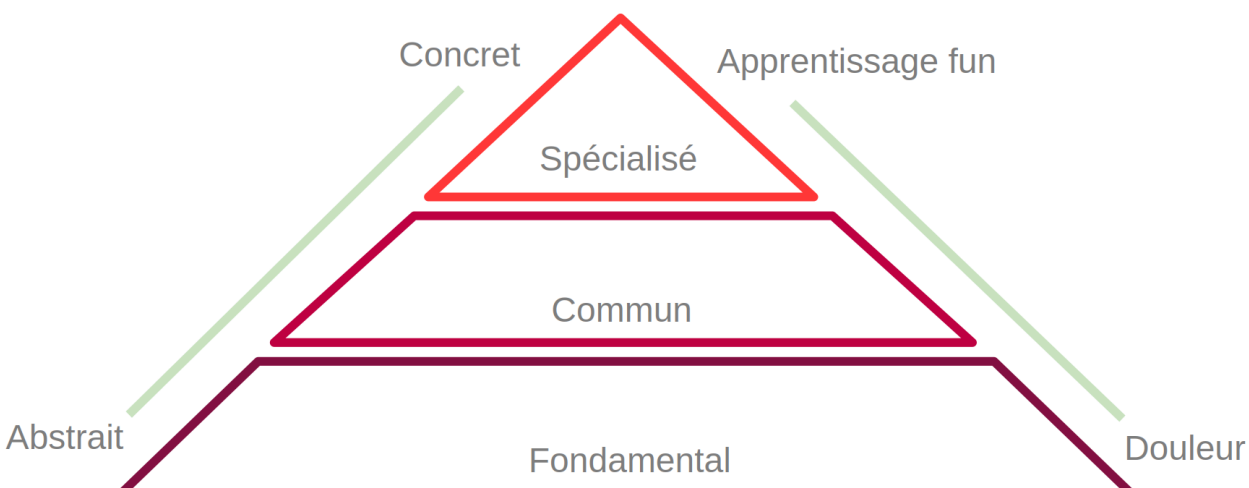
Algorithmique

3 Novembre 2025

Qui je suis-je donc ?

Timothée Cercueil

Ingénieur venant de l'**embarqué**, spécialisé en **Rust**, ai travaillé dans le kernel *Linux*, de l'IA, développé une blockchain.
Passionné de **sécurité informatique**, de rétrogaming, et de guitare classique



`Email: litchi.pi@pm.me` `Site web: litchipi.site` `LinkedIn: in/timothee-cercueil`

Introduction

But du cours: faire de vous des programmeurs en Python

Sujets abordés:

- Classes Python (et prog. objet)
- Fonctions avancées et Décorateurs
- Multithreading en Python
- Programmation de GUI

30h de cours, principalement du **TP**
Notation avec un **examen final** avec **rapport**

Introduction à la programmation

Algorithme: Série d'*instructions* permettant d'obtenir un *comportement* spécifique

Puzzle mental comme la création d'une recette de cuisine

```
gateau = Gateau()
gateau += (oeuf * 4)

gateau += (farine_gramme * 150)

while (not gateau.consistance_cremeuse()):
  gateau.battre_avec_fouet()

gateau += (sucre_gramme * 200)
gateau += (beurre_gramme * 100)

for tour in range(100):
  gateau.tour_de_cuillere()

four = Four()
four.cuire(gateau, temp=150, minutes=90)
```

Python: Language faisant *abstraction* des limites du système informatique

Permet de faire des algorithmes très simples et lisibles

Parfait pour apprendre l'informatique, même jusqu'à haut niveau

Mais performances très mauvaises comparées à d'autres languages

Objectif de ce cours

Vous faire *démarrer* votre aventure avec l'informatique, vous donner les *outils* pour réaliser vos algorithmes

Vous inciter à créer des *projets personnels* en Python pour **pratiquer l'esprit algorithmique**

Variables

Variable = *Boîte* dans laquelle on peut ranger des *données*

Chaque donnée a un *type*, permettant de savoir quelles opérations sont possibles sur elles

```
variable_a = 3                      # Type "int"
variable_b = "je suis une truite"   # Type "str"
variable_c = 1.234                  # Type "float"
variable_d = False                  # Type "bool"
```

Permet de stocker des *valeurs de retour*, et d'effecter des *opérations*

```
# Example d'opérations avec le type "int"
result = ma_fonction()
result = b + 5
result = a * 5
result = c - 10
resuld = d // 5
```

Conditions

Pilier central de l'algorithmique, permet d'exécuter des instructions différentes en fonction d'une *condition* précise

```
if variable == 3:
  print("Ma variable est égale à 3")

elif variable > 5:
  print("Ma variable est strictement supérieure à 5")

elif ((variable * 10) % 5) == 0:
  print("C'est cool")

else:
  print("Je suis une truite")
```

Permet de créer une *logique* à notre algorithme, sorte d'intelligence de la machine

Boucles

Pour répéter une opération, on utilise des *boucles*.

Il en existe 2 sortes (dans tous les languages):

Boucle while

S'exécute tant qu'une condition est valide *Attention aux boucles infinies*

```
while variable > 3:
  print("COUCOU")
  variable += 1
```

Boucle for

Exécute un code pour *chaque élément d'une séquence*

```
for i in range(10):
  print(i)
```

```
for i in [1, 2, 3, 4, 5]:
  print(i)
```

Afficher une donnée

Très utile pour *débugger* un programme et comprendre ce qu'il s'y passe

```
print(variable)

print("Message")

print("Ma variable", variable, "est cool")
```

Quand on développe un algorithme:

- On code la *logique* de l'algorithme, les étapes
- On *affiche* à différents endroits la valeurs de certaines variables
- On teste le code, en *validant* son fonctionnement

Le fonctionnement est valide lorsque la variable contient les *données attendues*, et que le *comportement* de l'algorithme est *correct* relativement aux **exigences**

Structures de données

Pour stocker un *ensemble* de données, on les organise en *structures de données*

Listes

```
my_list = [2, 3, 4]
my_list.append(12)
print(my_list[1])
```

Étagère où ranger les boîtes (données).

Pour choisir la boîte, on utilisera son **index**, c'est à dire *sa position* dans la liste, **à partir de 0**

Dictionnaires

```
my_dict = { "toto": 3, "coucou": { "je": "suis", "une": { "truite": True} } }
my_dict["tutu"] = False
print(my_dict["toto"])
```

Étagère avec *étiquettes uniques* sur chaque boîte (données).

On utilise une **clé** pour stocker une **valeur**, une unique clé aura une unique valeur.

Fonctions

Dans le cas où on veut appeller des instructions *plusieurs fois* en changeant ses *paramètres*, on peut **factoriser** ce code en une *fonction*.

```
# Fonction sans paramètre
def fonction():
  for i in range(10):
    print("Coucou")

fonction()
print("Je suis une truite")
fonction()
```

```
# Fonction avec paramètre
def fonction(arg):
  for i in range(10):
    print(arg)

fonction(3)
fonction("tutu")
```

TP

Exercices d’algorithmique