Algorithmique et programmation en Python

Description	Algorithmique	En langage Python

1 • Variables

Affectation
Donner une valeur à une variable. L'affectation $a \leftarrow a+1$ s'appelle une incrémentation.

$a \leftarrow 2$
a prend la valeur 2

Une affectation se fait avec =.
a +=1 signifie $a = a+1$.
Un test d'égalité se fait avec ==.

```
>>> a = 2
>>> a +=1
>>> a == 3
True
```

Affichage et saisie
Peu utilisé au lycée

Afficher n
Saisir n

```
print(n)
n = eval(input("Entrer n : "))
```

2 • Fonctions

Une fonction en Python est un programme réalisant une suite d'actions et/ou renvoyant une ou plusieurs valeurs.

Fonction le_nom :
Dessine un carré...

Fonction poly(x) :
Renvoie $x^3 - 2$

```
import matplotlib.pyplot as plt
def dessine_carre(c):
    plt.plot([c,c,-c,-c,c],[c,-c,-c,c,c])

def poly(x):
    return x**3-2
```

La saisie des variables est remplacée par les paramètres de la fonction.
Le résultat renvoyé par l'instruction **return** s'affiche automatiquement si une fonction est appelée dans la console.

```
>>> poly(3)
25
```

3 • Instructions conditionnelles

L'instruction conditionnelle ou le bloc d'instructions ne s'exécute que si la condition donnée est vérifiée.

On peut indiquer des instructions s'exécutant dans le cas contraire.
En Python, **elif** permet d'enchaîner les instructions conditionnelles.

Si < condition > faire :
Bloc d'instructions
Fin Si
Si < condition > faire :
bloc d'instructions
Sinon, faire :
bloc d'instructions
Fin Si

```
if IMC>=18.5 and IMC<25:
    print("Normal")
elif IMC<18.5:
    print("Dénutrition")
else:
    print("Surpoids/Obésité")
```

4 • Boucle non bornée : tant que ou while

La boucle **while** s'exécute tant qu'une condition donnée est vérifiée. En Python, l'indentation indique la fin d'une boucle.

Tant que < condition > faire :
bloc d'instructions
Fin tant que

```
n = 0
while 100*1.05**n < 200:
    n = n + 1
print(n)
```

5 • Boucle bornée : pour ou for

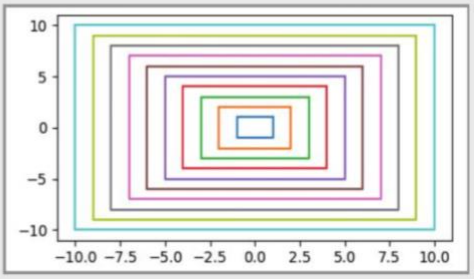
La boucle **for** s'exécute en énumérant un par un tous les éléments d'une liste ou tout autre objet dit itérable, comme les chaînes de caractères.

La fonction range() crée une suite d'entiers à parcourir.

- range(n) renvoie les n premiers entiers naturels : $0, 1, 2, \ldots, n-1$.
- range(n,p,k) renvoie la suite des entiers allant de n inclus à p exclu avec un pas de k.

Pour i allant de 1 à 10, faire :
bloc d'instructions
Fin pour

```
for i in range(1,11):
    dessine_carre(i) # fonction définie
plt.show()           # plus haut
```



Pour i allant de 0 à 100 avec un pas de 2, faire :

```
for i in range(0,101,2):
```

- Avec une liste de mots, le script ci-dessous, exécuté, renvoie le résultat ci-contre.

```
fruits = ["melon","pomme","banane"]
for mot in fruits:
    print(mot,"commence par",mot[0])
    print(mot,"termine par",mot[-1])
```

```
melon commence par m
melon termine par n
pomme commence par p
pomme termine par e
banane commence par b
banane termine par e
```