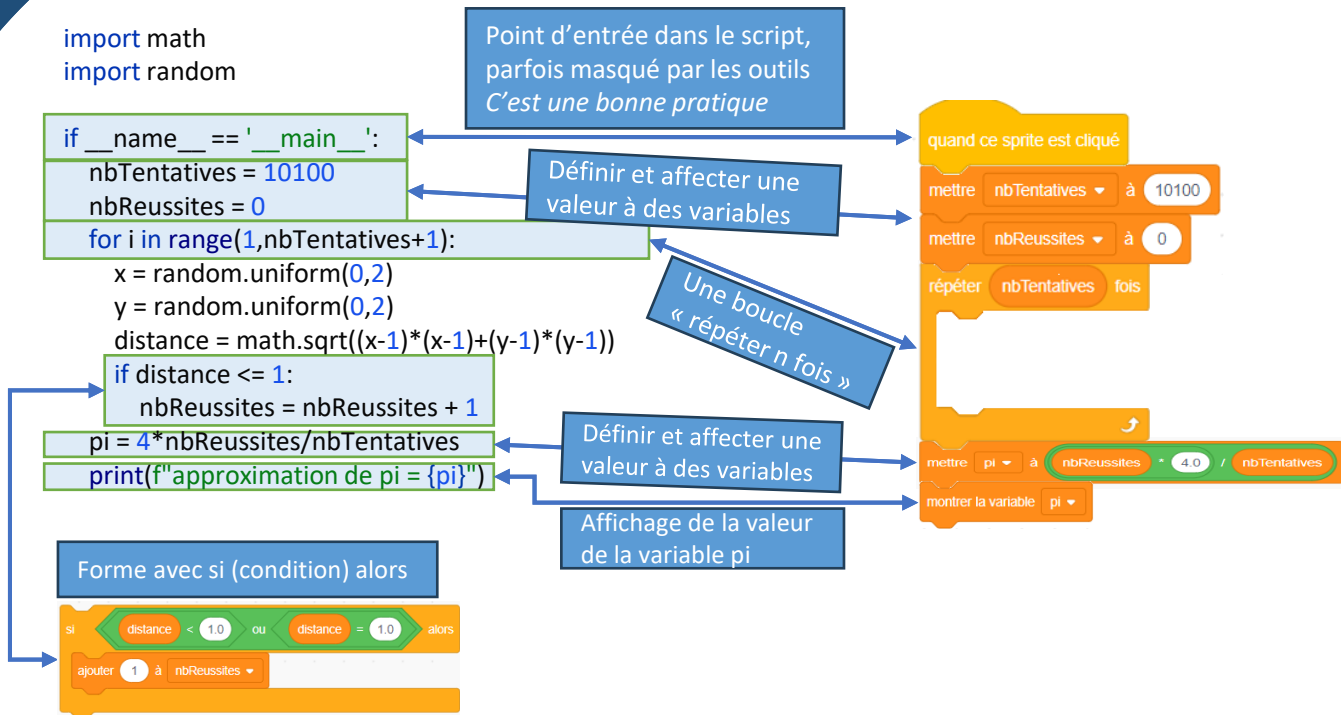
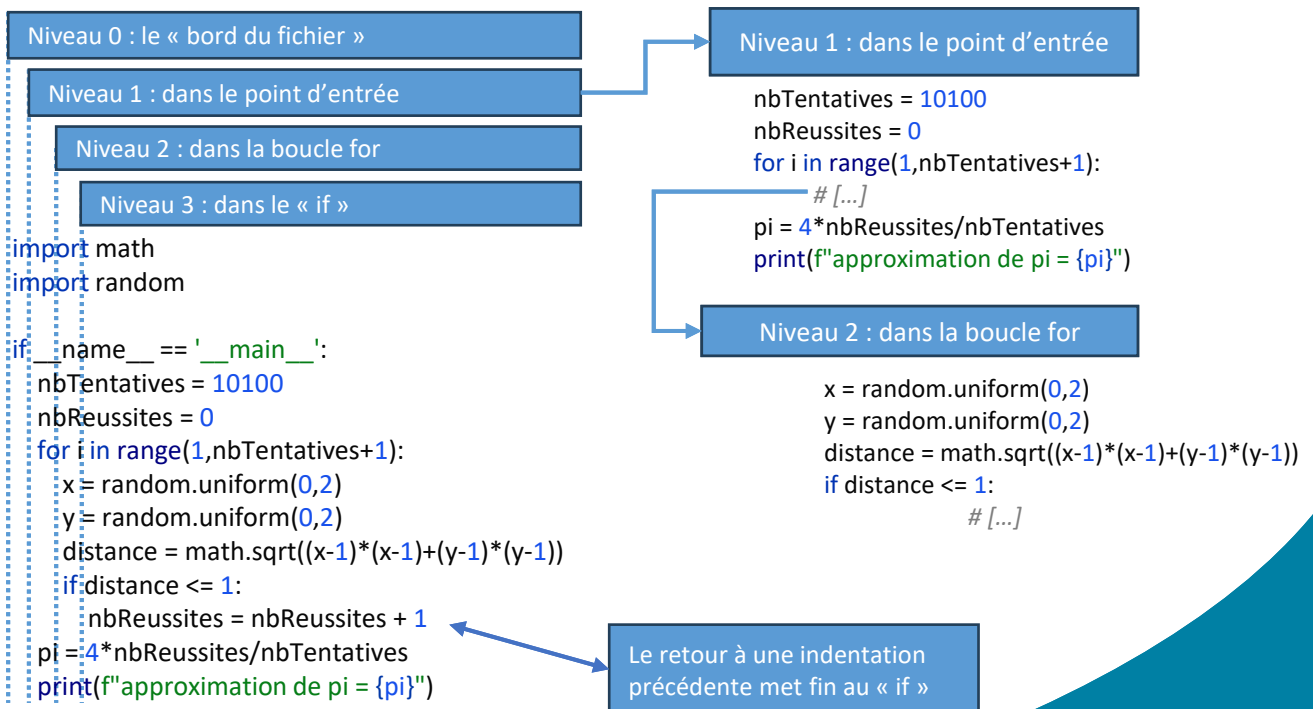


Correspondance python <-> block / scratch



Structuration du code avec les indentations



Memo Python

Version 1

Instructions et éléments de Python

Affectation : `nom_variable = valeur`

Variable : « symbole » qui représente une valeur changeante

Nom d'une variable : explicite, pour savoir ce que c'est

un commentaire commence par un

Type « courant » des variables :

- Nombre entier (int) , Nombre « réel » (float) ← `int()`, `float()`
- Expression booléenne (boolean, **True** / **False**)
- Chaîne de caractères (string / str) ← `str()`

Il existe des mots-clés réservés par Python. Exemple : « if », « else », « for », « while », etc.

print : pour afficher du texte
input : pour lire un texte tapé au clavier

Instruction conditionnelle « if » et les conditions

L'instruction conditionnelle commence par « if » suivi d'une expression booléenne et d'un « : »

```
if (note >= 16):  
    print("c'est une mention TB")  
elif (note >= 14):  
    print("c'est une mention B")  
elif (note >= 12):  
    print("c'est une mention AB")  
elif (note >= 10):  
    print("c'est une mention passable / sans mention")  
else:  
    print("malheureusement, le bac n'est pas obtenu")
```

elif permet d'enchaîner des cas différents, si le 1^{er} n'est pas vérifié.
ici on regarde si note est entre 14 et 16 (exclu). Ce test n'est fait que si le premier est faux. S'il est vrai, on sera bien entre 14 et 16

Ici on regarde si note est entre 12 et 14 (exclu). Ce test n'est fait que si les deux premiers sont faux. S'il est vrai, on sera bien entre 12 et 14

Et ainsi de suite : si tous les tests précédents sont faux, ce test-là est effectué...

Finalement, il est possible de terminer le « if » avec un « else » (sinon) pour le cas où toutes les conditions précédentes sont fausses

des booléens : **True** (vrai) et **False** (faux) et d'autres expressions entre booléen :

- **a and b** : (et) expression qui est vrai uniquement si a vaut vrai et si b vaut vrai
- **a or b** : (ou) expression qui est vrai si a vaut vrai ou si b vaut vrai ou les deux valent vrais.
- **not a** : (not = négation, le contraire) expression qui vaut vrai si a est faux

Les égalités / inégalités mathématiques forment des expressions booléennes :
(`==`, `!=`, `<`, `>`, `<=`, `>=`)

Le mot-clef « while » définit une boucle « tant que »

la boucle continue tant que la condition est vraie, cela peut être une expression plus compliquée

Les « : » sont obligatoires

```
while nbAllumettes > 0:  
    print(f"il reste {nbAllumettes} allumettes")  
    coupJ = int(input("Combien d'allumettes prenez-vous ? 1, 2 ou 3 : "))  
    nbAllumettes = nbAllumettes - coupJ  
    print(f"Après votre tour, il reste {nbAllumettes} allumettes")
```

La condition doit évoluer dans la boucle, dans le bon sens, sinon c'est une boucle sans fin / une boucle infinie qui bloque le script

Ce qui est répété est marqué par l'indentation. Le contenu de la boucle peut être n'importe quelles instructions (d'autres boucles, des « ifs », etc.)

Le mot-clef « for » définit une boucle « pour »

Variable de boucle dont la valeur va varier entre les bornes fixées

Définition des bornes, qui vont de la 1^{re} valeur (incluse) à la 2^e valeur (exclue)

```
for j in range(y - 1, y + 2):
```

Les « : » sont obligatoires

```
#y = 47192 et pageweb = open("carte.html", "w")
```

```
pageweb.write(f"<img src=\"https://tile.openstreetmap.org/17/67324/{j}.png\">\n")  
pageweb.write(f"<img src=\"https://tile.openstreetmap.org/17/67325/{j}.png\">\n")  
pageweb.write(f"<img src=\"https://tile.openstreetmap.org/17/67326/{j}.png\">\n")
```

Ce qui est répété pour chaque valeur de la variable de boucle est marqué par l'indentation. Le contenu de la boucle peut être n'importe quelles instructions (d'autres boucles, des « ifs », etc.)

Boucle non bornée - while :

Boucle bornée - for :

Memo Python

Version 1

Fonction :

