Chapitre 1 : Variables, instructions, iterations .

I Mode interactif :

Ouvrir le logiciel *« Thonny »*
Les trois chevrons >>> constituent l'invite de commande Python qui indique que l'interprète attend des instructions. (Utilisez la console en bas à gauche de *Thonny*)

A tester :
```
>>> 1+2
>>> 2 + 5*(10 – 1/2)
>>> 1+2 * 3
```

A noter que la priorité des opérations est satisfaite.

Ex1: Testez les instructions suivantes :
```
>>> 7/2
>>> 7//2
>>> 7%2
>>> 7 ** 2
```

A quoi servent les opérations // et % ?
A quoi sert l'opération ** ?

Variables :

Les résultats calculés peuvent être mémorisés par l'interprète, afin d'être réutilisés plus tard dans d'autres calculs.

A tester :
```
>>> a = 3 + 2
>>> a
```

La notation a = permet de donner un nom à la valeur à mémoriser. L'interprète calcule le résultat de 3 + 2 et le mémorise dans a. On accède à la valeur mémorisée en utilisant le nom a.

A tester :
```
>>> cube = a * a * a
>>> ma_variable =30
>>> ma_variable2 = 40
```

On peut donner n'importe quel nom pour une variable en utilisant des caractères tels que des lettres, des chiffres ou tirets bas. Il est cependant recommandé de ne pas utiliser des caractères accentués et d'utiliser des caractères minuscules.

A tester :
```
>>> a = a+1
>>> a
```

Le calcul de la nouvelle valeur de a peut utiliser la valeur courante de a

Remarque : En informatique , on peut écrire l'instruction a += 1 , qui a le même effet que l'instruction a = a +1
Une variable peut être imaginée comme une petite boîte portant un nom (ou étiquette) et contenant une valeur.

A tester :

```
>>> a, b = 2, 3
>>> a = a + b
>>> a, b
```

On peut représenter les différents états de l'interprète au cours des différentes instructions par :

{a 2 b 3} 1er état

{a 5 b 3} 2ème état

Si on ajoute l'instruction suivante :

```
>>> d = a
```

on arrive à l'état suivant :

{a 5 b 3 d 5}

Il est important de noter que d et a ne sont pas deux noms pour la même boîte, mais deux boites différentes. Les variables a et d sont indépendantes. Si on modifie a , cela ne modifie pas d.

Typage des données :

A tester :

```
>>> a = 3
>>> type(a)
>>> b = "bonjour"
>>> type(b)
>>> c = 32.1
>>> type(c)
>>> d= TRUE
```

a est de type entier (int) , b est une chaîne de caractères (type string) , c un flottant (type float) et d est un booléen (type bool)

II Mode programme.

Le mode programme de Python consiste à écrire une suite d'instructions dans un fichier et à les faire executer par l'interprète Python. On évite ainsi de ressaisir à chaque fois les instructions dans le mode interactif.

A tester :

Recopier le programme suivant dans la fenêtre d'édition de *Thonny (partie supérieure gauche de Thonny)*

test1.py

```
1  print("Bonjour le monde") #affiche une chaine de caractères
2  a= 20 # affectation de la valeur 20 à la variable a
3  b= 30
4  print(a) # affiche la valeur de a
5  print(b)
6  print("la somme de", a, "et de ", b,"vaut" , a+b)
7  print("abc", end=" ")
8  print("def", end="")
9  print("gh")
10
```

Enregistrer le programme dans le dossier chap1 de votre répertoire personnel sous le nom de test1.py

Ex 2 : Quelle est l'utilité de end= "" dans l'instruction print ?

Interaction avec l'utilisateur :

A tester :

test2.py

```
1 a = input("Entrez un nombre :")
2 print ("Le nombre est ", a)
```

Enregistrer le programme dans le dossier chap1 de votre répertoire personnel sous le nom de test2.py

A tester :

test3.py

```
1 a = input("Entrez un nombre : ")
2 print ("Le carré du nombre ", a ," est  ", a**2)
```

Un message d'erreur apparaît : *« TypeError: unsupported operand type(s) for ** »*
En effet , l'opération input() attend une chaîne de caractères. Python ne peut pas calculer le carré d'une chaîne de caractères.
Pour résoudre ce problème, il faut convertir cette chaîne par un entier avec l'opération int()
Modifier le programme pour qu'il fonctionne.

Ex 3 :
Écrire un programme qui demande votre année de naissance et qui calcule l'age que vous aurez en 2050.
Enregistrer votre programme sous le nom ex3.py

Ex 4 :
Écrire un programme ex4.py qui demande à l'utilisateur d'entrer un nombre de secondes et qui l'affiche sous forme d'heures/minutes/secondes

Ex 5 :
On souhaite écrire un programme ex5.py qui demande à l'utilisateur un nombre d'œufs et affiche le nombre de boîtes de 6 œufs nécessaires à leur transport. On considère le programme suivant, qui utilise la division euclidienne.
n = int(input(" combien d'oeufs : "))
print("nombre de boites : ", n//6)
a) Sur quelles valeurs de n ce programme est-il correct ?
b) Pourquoi n'est-il pas correct de remplacer n//6 par n//6 + 1
c) Proposer une solution correcte.

Ex 6 :
On souhaite calculer les coordonnées du point d'intersection de deux droites données sous la forme :
$\begin{cases} y = ax + b \\ y = cx + d \end{cases}$
On suppose ici que a, b, c et d sont des entiers. Écrire un programme ex6.py qui demande ces quatres valeurs à l'utilisateur puis affiche les coordonnées du point d'intersection. Que se passe-t-il si les droites sont parallèles ?

III Bibliothèque Turtle.
Python possède un grand nombre de bibliothèques ou modules qui apportent des instructions supplémentaires spécialisées.
La bibliothèque apportent des instructions de dessins supplémentaires.
Pour charger la totalité des instructions de la bibliothèque turtle, on écrira :
from turtle import *

Quelques instructions :

Instruction	description
goto(x, y)	Aller au point
forward(d)	Avancer de la distance d
backward(d)	Reculer de la distance d
left(a)	Pivoter à gauche de l'angle a
right(a)	Pivoter à droite de l'angle a
circle(r,a)	Tracer un arc de cercle d'angle a et de rayon r
dot(r)	Tracer un point de rayon r
up()	Relever le crayon (et interrompre le dessin)
down()	Redescendre le crayon (et reprendre le dessin)
width(e)	Fixer à e l'épaisseur du trait
color(c)	Sélectionner la couleur c pour les traits
color(r, v, b)	Sélectionner la couleur pour les traits où r, v, b sont des nombres compris entre 0 et 1 dans la synthèse additive des couleurs.
begin_fill()	Activer le mode remplissage
end_fill()	Désactiver le mode remplissage
fillcolor(c)	Sélectionner la couleur c pour le remplissage
reset()	Tout effacer et recommencer à zéro
speed(s)	Définir la vitesse de déplacement de la tortue (s est compris entre 0 et 10)
ht()	Ne montre plus la tortue(seulement le dessin)

A tester :
En mode console, tester les instructions suivantes :

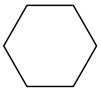
```
from turtle import *
forward(60)
left(120)
forward(60)
right(90)
circle(60, 300)
right(90)
forward(60)
goto(0, 0)
```

A tester : La spirale.
Recopier ce programme et l'enregistrer sous le nom test5.py dans le répertoire chap1.

```
# test5.py
from turtle import *
speed("slowest")
width(2)
circle(4, 180)
width(3)
circle(9, 180)
width(4)
circle(16, 180)
width(5)
circle(25, 180)
ht()          # cache la tortue
exitonclick() # Permet de garder la main
              # sur la fenêtre à la fin du programme
```

Cette spirale est constituée de demi-cercles dont les dimensions augmentent régulièrement.
Chaque demi-cercle a une épaisseur de trait de un supérieure à l'épaisseur du précédent et un diamètre qui est le carré de cette valeur . Il y a quatre demi-cercles soit deux tours complets.

Ex 7 :
Écrire les programmes qui permettent d'obtenir les figures suivantes :



hexagone.py

rectangle.py

flocon.py

A tester : boucle « for »
Écrire et tester le programme suivant boucle.py

```
1  # boucle.py
2  for i in range(10):
3      print(i)
```

Ex 8 :
Modifier les programmes hexagone.py , rectangle.py, flocon.py, test5.py en utilisant des boucles « for ».
Renommer ces programmes en hexagone_b.py, rectangle_b.py, flocon_b.py, spirale_b.py

Ex 9 :
Écrire les programmes qui permettent d'obtenir les figures suivantes :

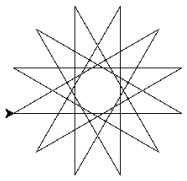


fig1.py

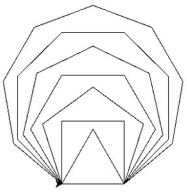


fig2.py

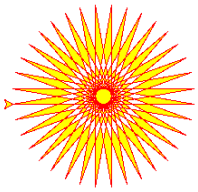


fig3.py

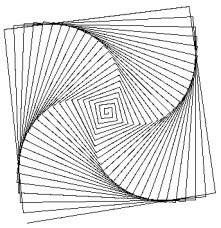


fig4.py