COURS – Langage PYTHON – SNT

Sommaire

A) Variables

1) Affectation

L'affectation consiste à attribuer une valeur à une variable

```
a = 83
b = 9.23
c1 = "Un deux trois"
c2 = "Un \"deux\" trois"
c3 = "Un deux\ntrois"
```

2) Type d'une variable

Le type de la valeur d'une variable se retrouve avec la fonction type()

int	83 0 -23 2	Entier
float	9.23 0.0 -1.7e-6	Nombre à virgule
str	"Un deux" 'Un deux' "L'avion"	Chaîne de caractères
list	["3", 3, 3.0]	Liste
bool	True False	Booléen

Conversion de type :

Diverses fonctions permettent de changer le type d'une variable.

Rque : le type d'une variable peut être modifié en fonction des calculs

```
int("15")       # 15
int(-15.56)     # -15 (troncature)
float(-15)      # -15.0
float("-2e-3") # -0.002
str(15)         # "15"
str(-0.002)     # "-0.002"
```

3) Variables globales, locales

```
a = 15        # variable globale
def plusUn() :
  a = 0       # variable locale à la fonction plusUn()
  a = a + 1
  print(a)

plusUn()      # 1
print(a)      # 15
```

B) Entrées & Sorties

1) Entrées : input()

Saisie d'une chaîne : nom = input("Quel est votre nom ?")
Saisie d'un entier : n = int(input("Nombre de frères et sœurs"))
Saisie d'un nombre à virgule : x = float(input("Température à 12h00")

2) Sorties : print()

Affichage :
print("Bonjour !")
`print(2)`

```
a = -3
print(1)
print(2)
print("Le carré de ", a," est " , a * a)

print(1, end = " ") # end = " " évite le retour à la ligne
print(2, end = " ")
print(3, end = " ")
```

C) Les Nombres

1) Opérations sur les nombres

Les tableaux ci-contre indiquent les principales opérations possibles en langage PYTHON

Rque : tester ces différentes opérations sur une console IDLE

```
a = 2
print(a + 3)    # 5
print(a - 3)    # -1
print(3 * a)    # 6
print(15 / 6)   # 2.5
print(15 // 6)  # 2
print(15 % 6)   # 3
print(3 ** a)   # 9
```

+	Addition
-	Soustraction
*	Multiplication
/	Division
//	Division entière
%	Reste de la division
**	Puissance

2) Fonctions natives

Quelques fonctions mathématiques sont toujours disponibles

```
print(round(3.14159265, 2))  # 3.14
print(round(3.14159265, 4))  # 3.1416
print(int(3.14159265))       # 3
print(int(3.8))              # 3
print(abs(2.3))              # 2.3
print(abs(-2.3))             # 2.3
```

round()	Valeur approchée
int()	Partie entière
abs()	Valeur absolue

3) Fonctions mathématiques

La plupart des fonctions mathématiques nécessitent d'importer une bibliothèque, par exemple math ou numpy

```
from math import *
print(sqrt(16))   # 4.0
print(pi)         # 3.141592653589793
print(sin(pi/2))  # 1.0
print(cos(pi/4))  # 0.7071067811865476
```

4) Fonctions aléatoires

```
from random import *
a = random()  # nombre aléatoire dans [0;1[
```

```
From random import *
L = [7, 3, 8, 5, 6]
a = choice(L)
b = randint(5, 10)
```

D) Chaînes de caractères

1) Opérations sur les chaînes

Exemple : tester ces programmes sur une console IDLE

```
ch = "Pierre"
print(len(ch))
```

```
a = "Un cours"
b = " ça s'apprend !"
c = a + b  # concaténation
print(c)
```

2) Caractère d'échappement

L'exemple suivant donne un retour à la ligne automatique

```
txt='"Bonjour !", dit-elle.\n"Bonjour", répondit-il.
print(txt)
```

Remarques :

- \n insère un retour à la ligne.
- \' insère une apostrophe dans une chaîne délimitée par des apostrophes.
- De même, \" insère des guillemets dans une chaîne délimitée par des guillemets.

3) Accès aux caractères individuels

Exemple : il est possible de nommer chaque caractère d'une chaîne de caractères

```
ch = "Constance"
print(ch[0], ch[4], ch[8])
```

4) Accès aux sous-chaînes

Exemple : On peut effectuer un affichage par tableaux ou listes

→ Tester ce programme sur une console PYTHON

```
nb = "123456789"
print(nb[1:3])
print(nb[0:len(nb)-2])
print(nb[:-2])
print(nb[2:len(nb)])
print(nb[2:])
```

E) Les Listes

1) Opérations sur les listes

Une liste est un conteneur indexé d'éléments séparés par des virgules, l'ensemble étant enfermé entre crochets

```
liste = ["a", "b", 20, 30, "cd"]
print(liste[0], liste[2], liste[4]) # a20cd
liste[0] = "e"                      # ["e", "b", 20, 30,"cd"]
len(liste)                          # 5
del(liste[3])                       # ["e", "b", 20,"cd"]
liste.append("a")                   # ["e", "b", 20,"cd","a"]
print(liste.index("b"))             # 1
liste.remove("b")                   # ["e", 20,"cd","a"]
```

```
liste = [1, 3, 2, 4]
min(liste)                      # 1
max(liste)                      # 4
liste.reverse()                 # [4, 2, 3, 1]
liste.sort()                    # [1, 2, 3, 4]
liste.sort(reverse=True)        # [4, 3, 2, 1]
```

2) Avec la fonction range()

Syntaxe :
range(début, fin, pas)
La fonction list() convertit le type range en type list.

```
list(range(5))      # [0, 1, 2, 3, 4]
list(range(2,5))    # [2, 3, 4]
list(range(0,5,2))  # [0, 2, 4]
```

F) Bloc d'instructions

1) Instructions conditionnelles

Avec un « if »

```
nombre = -10
if nombre>0 :
  print("Le nombre choisi est positif")
print("Fin du programme")
```

Avec un « if … else »

```
nombre = 10
if nombre>0 :
  print("Le nombre choisi est positif")
else :
  print("Le nombre choisi est négatif ou nul")
print("Fin du programme")
```

Test de plusieurs valeurs

```
n = 10
if n<0 :
  print("Le nombre est négatif")
elif n==0 :
  print("Le nombre est égal à zéro")
else :
  print("Le nombre est positif")
```

Opérateur de test

if n==0 :	n est égal à zéro
if n>0 :	n est positif
if n!=34 :	n est différent de 34
if (n>0) and (n<10) :	n est compris strictement entre 0 et 10
if 0<n<10 :	n est compris strictement entre 0 et 10
if (n<0) or (n>10) :	n est négatif ou strictement supérieur à 10
if n%5==0 :	n est divisible par 5

Différences entre =, == et != :

```
n = 10          # affectation
n == 0 # False  # test d'égalité
n = 0           # affectation
n == 0 # True   # test d'égalité
n != 0 # True   # test d'inégalité
```

2) Instructions répétitives

Boucle bornée « for … in … »

```
for n in range(3) :
  print("Bonjour")
print("Fin")
```

Rque :

- Toutes instructions indentées font partie de la boucle
- range(10) définit 10 itérations mais la première valeur est 0 donc la dernière est 9

```
for n in range(10) :
  print(n)  # affichage des valeurs de 0 à 9
```

```
for n in range(1,10) :
  print(n)  # affichage des valeurs de 1 à 9
```

Boucle non bornée « while »

Le nombre d'itérations n'est pas toujours connu à l'avance.

```
gagne = 0
while gagne == 0 :
  n=int(input("Entrer un nombre"))
  if n == 10 :
    gagne = 1
    print("Gagné")
```

3) Sous programmes

Les Fonctions

Une fonction prend communément un ou plusieurs paramètres et retourne une valeur :

```
1  def puissance(x,n) :
2    y = x**n
3    return y
4
5  print(puissance(3,2)) # 9
```

Les Procédures

Une procédure exécute une suite d'instructions sans retourner de valeur :

```
1  def table7() :
2    for i in range(1,11) :
3      print(7*i)
4
5  table7()  # 7 14 21 28 35 42 49 56 63 70
```

Une procédure prend éventuellement un ou plusieurs paramètres :

```
1  def table(base) :
2    for i in range(1,11) :
3      print(base*i)
4
5  table(8)  # 8 16 24 32 40 48 56 64 72 80
```

G) Déclaration de bibliothèques

1) Sans rappel du nom de la bibliothèque

A réserver aux programmes simples. Il est possible d'importer seulement les fonctions désirées.

```
1  from numpy import *
2  print(sqrt(9))
3  print(sin(2))
```

```
1  from numpy import sqrt, sin
2  print(sqrt(9))
3  print(sin(2))
```

2) Avec rappel du nom de la bibliothèque

Évité les conflits de variable ou de fonction.

```
1  import numpy
2  print(numpy.sqrt(9))
3  print(numpy.sin(2))
```

3) Avec un alias

Donne un nom plus court à la bibliothèque.

```
1  import numpy as np
2  print(np.sqrt(9))
3  print(np.sin(2))
```

H) Autres types natifs

1) Les Tuples (ou n-uplets)

En Python, on évite les mots tableau ou vecteur qui prêtent à confusion, ou alors on précise de quoi il s'agit.

```
1  t = (1, "a", 2.3)
2
3  print(len(t))
4  print(t[1])
5  t[1] = "b" # Erreur, les éléments d'un tuple sont immuables
```

2) Les Ensembles

La fonction set() convertit le type list en type set (ensemble en français).

```
1  e1 = {"moineau", "couleuvre", "renard", "renard"}
2  e2 = {"moineau", "gardon", "lion"}
3  e3 = {"moineau"}
4
5  print(e1)      # "renard" ne compte qu'une fois
6  print(len(e1))
7  for p in e1:
8    print(p)
9  print(e1 | e2) # Union
10 print(e1 & e2) # Intersection
11 print(e1 - e3) # Différence
```

3) Les Dictionnaires

```
1  d = {"oiseau":"moineau", "reptile":"couleuvre", "mammifere":"renard"}
2
3  print(d["reptile"])
4  for p in d:
5    print(d[p])
6  d["poisson"] = "gardon" # Ajout d'un élément
7  del d["reptile"]        # Retrait d'un élément
8  d["oiseau"] = "pinson"  # Modification d'un élément
9  print(d)
```