Calculs

Les 4 opérations : **+ - * /**
Puissances : ******
Division entière : **// %**
Fonctions : **abs() round()**

Variables

Noms des variables : 1er caractère = **a**...**zA**...**Z_** suivi de **a**...**zA**...**Z_0**...**9**
(accents possibles, distinction casse min/MAJ)

type(var)	Retourne le type de la variable var
del var	Supprime la variable var

Affectations

n = 23	Entier
x = 23.0	Décimal flottant
s = "23"	Chaîne de caractères
x, y = 1, 5	Affectation multiple
x, y = y, x	Permutation des valeurs de x et y
i = 0xa0	Hexadécimal
i = 0b101010	Binaire

```
a += 1
```

Définition de fonctions

```
def cube(x):
    """docstring de la fonction :
    Entrée : x doit être un nombre (int ou float)
    Sortie : renvoie le cube de x"""
    return x**3
```

Assertions

```
assert condition, "message si condition fausse"
```

Entrées - sorties

```
n = int(input("entrez un entier: "))
x = float(input("entrez un décimal: "))
print("J'ai",age,"ans")        # Ancienne méthode
print(f"J'ai {age} ans")       # f-string
print(i,end="")                # Pas de saut de ligne
```

Tests

```
if a==0:
    print("c'est nul")
elif a>0:
    print("c'est strictement positif")
else:
    print("c'est strictement négatif")
```

Comparaisons et expressions booléennes

```
== != < <= > >= and or not
a = (2**3==9)
```

Évaluation « paresseuse »

1==1 or 2/0	Vaut **True** et ne déclenche pas d'erreur
1==0 and 2/0	Vaut **False** et ne déclenche pas d'erreur

Boucle while

```
a = 0          # Initialiser la variable de condition
while a < 10:
    print(a)
    a += 1     # Et la modifier dans la boucle
```

Boucle for

```
for i in range(5):
    print(i)
for i in range(1,5):
for i in range(1,5,2):
for i in [3,7,4,9]:
for i in (3,7,4,9):
for i in "coucou":
```

Séquences

Tuples, listes, chaînes de caractères et objets range ont en commun d'être des suites ordonnées d'éléments. On peut les parcourir avec une boucle for et leur appliquer les opérations ci-dessous :

`e in s`	Vrai si e est un élément de s
`s+t`	Concaténation de s et t
`s*n`	Répète n fois la séquence s
`s[i]`	Élément d'indice i
`s[0]`	Premier élément
`s[-1]`	Dernier élément
`s[i:j:p]`	Slice
`s[:5]`	Du début jusqu'à l'indice 5 non compris
`s[3:]`	De l'indice 3 jusqu'à la fin
`s[:]`	Utile pour dupliquer une séquence
`s[::-1]`	La séquence à l'envers
`len(s)`	Nombre d'éléments de la séquence s
`min(s), max(s)`	Minimum et maximum de la séquence s
`sum(s)`	Somme des éléments contenus dans s
`s.count(x)`	Nombre d'occurrences de x dans s
`s.find(x)`	Indice de la 1ère occurrence de x
`s.rfind(x)`	Indice de la dernière occurrence (reverse find)

Chaînes de caractères

Les chaînes de caractères sont non mutables.

```
txt = ""
txt = "Je m'appelle Bond"
txt = 'Je m\'appelle Bond'
txt = """sur plusieurs lignes
  ou avec guillemets 'simples' ou "doubles" """
```

En Python, le caractère d'échappement est le backslash : fin de ligne=`\n`, tabulation=`\t`, guillemet=`\"`, ou `\'`, backslash=`\\`

`",".join(s)`	Converti la liste s en chaîne en utilisant "," comme séparateur
`txt.split(",")`	Converti la chaîne txt en liste en utilisant "," comme séparateur
`txt.lower()`	Converti txt en minuscule et renvoie la chaîne obtenue
`txt.upper()`	Converti txt en majuscule et renvoie la chaîne obtenue
`txt.lstrip()`	Supprime dans txt les espaces de gauche et renvoie la chaîne obtenue
`txt.rstrip()`	Espaces de droite
`txt.strip()`	Espaces de gauche et de droite
`txt.replace(old,new)`	Renvoie une chaîne identique à txt où les occurrences de old sont remplacées par new
`chr(x)`	Renvoie le caractère de numéro Unicode x
`ord(char)`	Renvoie le numéro Unicode du caractère char

Listes

Les listes sont mutables.

```
l = []
l = [1, 2, 3, 4, 5]
l[i] = 6
```

`l[i:j] = s`	Remplace dans l la sous-liste l[i:j] par les valeurs de la séquence s
`del l[i:j]`	Équivalent à l[i:j]=[]
`l.append(x)`	Ajoute l'élément x à la fin de la liste l
`l.clear()`	Vide la liste l
`l.index(e)`	Renvoie l'index du premier élément de l égal à e
`l.extend(s)`	Ajoute à la fin de la séquence l les éléments de la séquence s
`l.insert(i,x)`	Insère l'élément x à l'indice i et décale la suite de la liste l
`l.pop(i)`	Renvoie et supprime l'élément d'indice i de la liste l
`l.remove(x)`	Retire de la liste l le premier élément tel que l[i]==x (Bien comprendre la différence avec del l[x] !)
`l.reverse()`	Renverse l'ordre des éléments de la liste l
`l.sort()`	Trie les éléments de la liste l
`l = [[1, 2], [3, 4], [5, 6]]`	Liste de liste (= tableau à 2 dimensions). Ici, l[2] = [5, 6] et l[2][0] = 5

Tuples

Les tuples sont non mutables.

```
t = ()
t = (1, 2, 3, 4, 5, 6)
```

Dictionnaires

Les dictionnaires sont mutables

```
d = {}
d = {"nom":"Turing",
     "prénom":"Alan",
     "profession":"mathématicien"}
```

`d[k]=v`	Associe la valeur v à la clé k
`d[k]`	Renvoie la valeur associée à la clé k et déclenche une erreur si cette clé n'existe pas
`d.get(k)`	Même chose sans déclencher d'erreur si k n'existe pas
`d.clear()`	Vide le dictionnaire d
`d.pop(k)`	Supprime et renvoie la valeur associé à k
`len(d)`	Nombre d'associations clé-valeur dans le dictionnaire
`for i in d.keys()`	Parcours le dictionnaire par clés
`for i in d.values()`	Parcours le dictionnaire par valeurs
`for i,j in d.items()`	Parcours le dictionnaire par clés en ayant les valeurs associées

Compréhensions de listes

```
l=[x for x in range(50)]
l=[x**2 for x in range(50) if x%3==0]
```

Modules

`help(math)`	Affiche une aide rapide du module math
`import math` `s = math.sqrt(4)`	Importation - Méthode 1 : sqrt() doit être précédé du nom du module
`from math import sqrt` `s = sqrt(4)`	Importation - Méthode 2 : sqrt() n'est pas précédé du nom du module
`from math import *`	Importation - Méthode 3 : à éviter

Module Math

`sqrt(x)`	Racine carrée
`floor(x)`	Arrondi à l'entier inférieur
`ceil(x)`	Arrondi à l'entier supérieur
`gcd(x,y)`	PGCD de x et y
`frexp(x)`	Retourne m et e tels que x = m * 2^e
`sin(x), cos(x), tan(x), asin(x)`	
`pi`	

Module Random

`random()`	Flottant sur [0 ; 1[
`randint(a,b)`	Entier entre a et b inclus
`choice(s)`	Elément au hasard dans la séquence s
`shuffle(l)`	Mélange la liste l

Module Os

`chdir(path)`	Change le répertoire en cours
`getcwd()`	Renvoie le répertoire en cours
`listdir(path)`	Renvoie la liste des fichiers et répertoires contenus dans path
`mkdir(path)`	Crée un nouveau répertoire
`remove(path)`	Supprime un fichier
`rmdir(path)`	Supprime un répertoire vide
`rename(src,dst)`	Renomme un fichier ou un répertoire

Fichiers textes

`f=open(path, mode="a", encoding="utf8")`

Les principaux modes sont : `"r"` (read - par défaut), `"w"` (write), `"+"` (read and write), `"a"` (append)

Si on veut lire le fichier octet par octet sans utiliser d'encodage texte, on peut ajouter `"b"` au mode (binary).

`f.read()`	Renvoie une chaîne avec tout le contenu de f
`f.readline()`	Renvoie la ligne suivante du fichier
`f.readlines()`	Renvoie la liste des lignes du fichier
`f.write(s)`	Écrit la chaîne s dans le fichier f
`f.writelines(l)`	Écrit les lignes contenues dans la liste l dans le fichier f
`f.tell()`	Position en cours du pointeur de fichier
`f.seek(i,0)`	Déplace le pointeur de fichier de i octets en partant du début
`f.close()`	Vide le buffer d'écriture et ferme le fichier (A ne pas oublier !!)

Gestion des erreurs d'exécution

`try:` `  r=a/b`	Opération dangereuse dans le try
`except ZeroDivisionError:` `  print("division par 0")`	Code exécuté uniquement si l'erreur est de type "ZeroDivisionError"
`else:` `  print(r)`	Code exécuté uniquement si il n'y a pas eu d'erreur
`finaly:` `  print(a,b)`	Code exécuté que l'on soit passé par le except ou par le else