TP n°1 pour démarrer

1 Apprendre à itérer

Exercice 1 (somme des entiers) Écrire un script Python qui calcule la somme :

$$1+2+3+\cdots+99+100.$$

On pourra compléter le script suivant :

```
s = 0
for k in range(........): # pour k de ... à ...
    s = ....
print(s)
```

Exercice 2 Pour une classe de 35 élèves, on peut démontrer que la probabilité qu'au moins deux élèves aient la même date d'anniversaire est le nombre $1-q$ où q est le nombre suivant :

$$q=\left(1-\frac{1}{365}\right)\times\left(1-\frac{2}{365}\right)\times\left(1-\frac{3}{365}\right)\times\cdots\times\left(1-\frac{34}{365}\right).$$

Calculer le nombre q avec Python. Paul pense que dans votre classe, il y a au moins 75% de chances que deux élèves aient la même date d'anniversaire. Qu'en pensez-vous ?

Exercice 3 Calculer 43×25 uniquement à l'aide d'additions.

Exercice 4 On part du nombre $u=1$. On fait des multiplications successives par 2. Par exemple, pour dépasser 10, il faut 4 multiplications :

$$1 \underset{\times 2}{\to} 2 \underset{\times 2}{\to} 4 \underset{\times 2}{\to} 8 \underset{\times 2}{\to} 16$$

Combien de multiplications sont nécessaires pour dépasser un milliard. On répondra en complétant le script Python suivant :

```
u = 1
n = 0
while ... :
    u = ...  # nouveau u
    n = ...  # nouveau n
print( ...)
```

Exercice 5 On part du nombre $u=0$. On lui ajoute 1 et on multiplie le résultat par 2. Puis on réitère... Au bout de combien d'itérations, va-t-on dépasser la valeur 100000 ?

2 Un opérateur important, l'opérateur modulo %

Dans 23, il y a 3 fois 7, et il reste 2. C'est la division du primaire, appelé division euclidienne que l'on peut écrire $23=3\times 7+2$. L'entier 3 est le quotient et l'entier 2 le reste. Python sait faire cette division :

```
>>> 23 // 7
3
```

```
>>> 23 % 7
2
```

La division euclidienne est très utile pour tester la divisibilité d'un entier par un autre entier. Par exemple 2047 est divisible par 23 car le reste de la division de 2047 par 23 vaut 0 :

```
>>> 2047 % 23
0
```

On dit aussi que 2047 est égal à 0 modulo 23, c'est pourquoi l'opérateur `%` est aussi appelé opérateur `modulo`. En particulier, un entier pair est un entier égal à 0 modulo 2 et un entier impair est un entier égal à 1 modulo 2.

Exercice 6 Convertir 42569 secondes en heures, minutes et secondes à l'aide des opérateurs `//` et `%`

Exercice 7 Écrire un script Python qui **affiche** les diviseurs communs de deux entiers. Par exemple $a = 40$ et $b = 60$. En déduire un moyen de programmer le pgcd de deux entiers.

Exercice 8 (Entier le plus divisible) On s'intéresse aux nombres de diviseurs d'un entier. Par exemple, l'entier $n = 10$ a pour diviseurs $1, 2, 5, 10$. Le nombre de diviseurs de 10 est donc 4.

1. Écrire un script qui détermine le nombre de diviseurs de l'entier $n = 224$.
2. Déterminer le plus petit entier ayant au moins 100 diviseurs.

3 Le besoin d'utiliser des fonctions

Dans le dernier exercice, on sent que dans le dernier code, on a recopié une partie du code de l'avant-dernière question. On peut éviter cela en utilisant le concept fondamental de fonction.

```
def nombre_diviseurs(n):
    '''Données: n un entier naturel
       Résultat: le nombre de diviseurs de n
    '''
    nb = 0 #
    for k in range(1, n+1): # pour k de 1 à n
        if n % k == 0:
            nb = nb + 1
    return nb
```

En exécutant dans la console `nombre_diviseurs(224)`, on récupère le nombre de diviseurs de 224.

Exercice 9 (Écriture de fonctions)

1. L' indice de masse corporelle IMC est le nombre obtenu en divisant son poids en kg par le carré de sa taille en mètres. Écrire une fonction `IMC(poids, taille)` qui renvoie l'IMC.
2. Écrire une fonction `norme(x,y)` qui renvoie la norme d'un vecteur de coordonnées (x, y).

Exercice 10 (Afficher c'est pas gagner) Paul a besoin d'utiliser la fonction f définie sur $\mathbb{R}$ par $f(x) = 2x + 1$. Il la programme avec le script suivant

```
def f(x):
   print(2*x+1)
```

Calculer ainsi $f(10), f(5)$ puis $f(f(5))$. Commenter puis corriger.

Exercice 11 Voici deux fonctions nommées `truc` et `bidule`.

```
def truc(x):
    print(x)
    return(2*x)
    print(3*x)
    return(4*x)
```

```
def bidule(x):
    print(x)
    print(2*x)
    return(3*x)
    print(4*x)
```

On exécute `truc(10)`.

1. Quelle(s) valeur(s) (est) sont affichée(s) ? Quelle valeur est renvoyée ?
2. Même question avec `bidule(10)`.

Exercice 12 Écrire une fonction `trinome(a,b,c)` qui renvoie les solutions réelles de l'équation du second degré $ax^2 + bx + c = 0$ avec a, b, c des réels et a non nul. S'il n'y a pas de solution, la fonction renverra l'objet None, et s'il y a deux solutions, elle renverra un tuple `(x1,x2)`.

Exercice 13 (Maximum de 2 et 3 nombres) Écrire une fonction `max2` qui renvoie le maximum de deux nombres. Faire ensuite de même avec `max3` pour 3 nombres. Remarque : pour programmer `max3`, il y a une façon élégante et très courte (deux lignes de code).

Exercice 14 (Fonction factorielle)

1. Écrire une fonction `factorielle(n)` qui prend en argument un entier naturel n et renvoie $n!$.
2. Déterminer avec Python, le plus petit entier n tel que $n! \geqslant 10^{500}$. Faire de même avec une «cible» à 10^{50000}. Si votre algorithme met plusieurs secondes, réfléchir à un moyen de diminuer la difficulté des calculs.

4 Utiliser un module existant

La fonction `print` est un exemple de fonction «built-in» au sens de fonction «primitive». On est souvent amené à utiliser des fonctions «non primitives» qui se trouvent dans des modules.

Par exemple, Python peut générer des nombres aléatoires. Il faut pour cela importer le module `random`.

```
>>> import random
>>> random.randint(1,6)  # génère un entier compris entre 1 et 6
1
>>> random.random() # génère un flottant compris entre 0 et 1
0.6469912430701616
```

Exercice 15 (Temps d'attente du double six)

1. Écrire un script qui modélise la situation suivante : on lance deux dés jusqu'à ce que la somme des deux dés soit égale à 12. Le programme doit afficher à la fin le nombre de lancers nécessaires jusqu'à l'obtention du double six.

2. Simuler 10000 lancers de deux dés et compter le nombre de fois où la somme 7 a été obtenue. Comment peut-on en déduire une estimation de la probabilité de faire 7 avec deux dés ?

5 Dessiner avec la tortue

Pour cette section, on utilise le module `turtle`, que l'on importe en exécutant `from turtle import *`. Si on utilise Pyzo, il est préférable d'utiliser TK (Tkinter) comme GUI (Graphical User Interface). Pour cela aller dans l'onglet Shell, puis Edit Shell Configurations et choisir TK comme GUI. Relancer alors votre shell.

L'instruction `forward(50)` permet d'avancer de 50 pixels et `right(30)` fait tourner la tortue de 30 degrés vers la droite.

Exercice 16 (Ma maison) Dessiner la maison de la figure 1 ci-dessous où tous les segments ont même longueur :

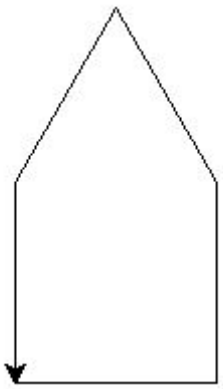

FIGURE 1 – Ma maison

Exercice 17 Dessiner avec Turtle la figure suivante 2

Exercice 18 (polygone régulier)

1. Tracer un triangle équilatéral, un carré puis un pentagone régulier.

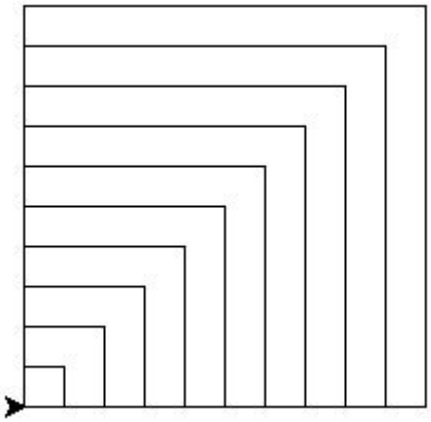

FIGURE 2 – Carrés emboîtés

2. On automatise tout cela ! Définir une fonction `polygone(n, l)` qui trace un polygone régulier à `n` côtés, chacun de longueur `l`, puis reproduire à l'aide de cette fonction la figure 3.

3. En déduire une méthode pour tracer un cercle.

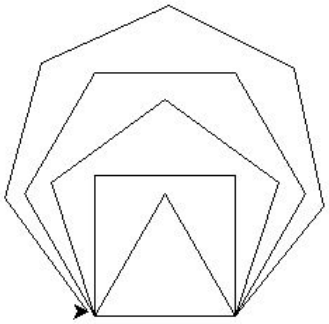

FIGURE 3 – polygones réguliers