Les fonctions

I) Les fonctions en Python

---

1. Comment un informaticien fait bouillir de l'eau avec une casserole vide, un robinet et une plaque chauffante ? Il remplit la casserole, il fait chauffer la casserole jusqu'à ébullition.
2. Comment un informaticien fait bouillir de l'eau avec une casserole vide, un robinet et une plaque chauffante ? Il vide la casserole et ré-applique la réponse de la question 1

En informatique, on essaie d'être assez fainéant mais productif, c'est à dire lorsque l'on a fait quelque chose on essaie de le réutiliser. Pour ce faire on utilise des fonctions. Comme en mathématiques, *une fonction c'est juste quelque chose en entrée et qui donne quelque chose en sortie*. En informatique parfois ce quelque chose c'est rien.

I.1) Généralités

En Python, il est possible de définir des fonctions. Les fonctions en programmation permettent d'utiliser de multiple fois une séquence d'instructions qui dépendent d'un certain nombre de paramètres (on parle aussi d'arguments). En Python, la définition d'une fonction passe par la définition :

- du nom de la fonction ;
- de sa liste de paramètres en entrée (qui peut être vide) ;
- de la séquence d'instruction dans la fonction ;
- de la sortie de la fonction (qui est facultative en Python). Elle prend forme suivante :

In [ ]:
```
def nom_fonctions(a,b,c):
    instructions
    return sortie
```

Ici, la première ligne comporte :

- la commande `def` qui indique qu'une fonction va être définie
- `nom_fonction` qui désigne le nom de la fonction (choisie par l'utilisateur) ;
- des variables `a` , `b` et `c` qui sont les paramètres de la fonction (dont le nom est choisi par l'utilisateur), ils sont placés entre parenthèses et séparés les uns des autres par des virgule (il peut y en avoir n'importe quel nombre entier, y compris 0, dans ce cas on écrit `nom_fonction()`) ;
- un deux points `:`

Cette première ligne est suivie, après indentation, de la séquence d'instructions de la fonction qui se termine généralement (mais pas nécessairement) par un renvoi de la fonction indiqué par la commande return. Le renvoi permet de définir une valeur renvoyée en sortie. Le code d'une fonction peut contenir plusieurs renvois correspondant à différents cas (en utilisant des instructions conditionnelles, par exemple). Toutefois, l'exécution de la fonction n'effectue jamais qu'un seul envoi.

> **Exemple 1:**
>
> La fonction f prend un paramètre x et renvoie la valeur obtenue par l'instruction `2*x+3`. En mathématiques, elle correspond à une fonction affine de la forme $f : x \mapsto 2x + 3$.

In [ ]:
```
def f(x):
    return 2*x + 3
```

> **Exemple 2:**
>
> La fonction `puissance` prend deux paramètres a et n et renvoie en sortie a^n.
>
> On remarque dans le code suivant que l'indentation de la boucle s'ajoute à l'indentation de la définition de la fonction.

In [ ]:
```
def puissance(a,n):
    p = 1
    for k in range(n):
        p = p * a
    return p
```

> **Exemple 3:**
>
> La fonction `mini` prend deux paramètres a et b et renvoie le plus petit des deux. Ici le renvoi dépend d'une instruction conditionnelle.

In [ ]:
```
def mini(a,b):
    if a<b:
        return a
    else:
        return b
```

> **Exemple 4:**
>
> La fonction `bonjour` affiche Bonjour! à l'écran. Elle ne prend aucun paramètre et ne fait aucun renvoi.
>
> **Une fonction ne doit JAMAIS printer son résultat !! (0 direct sinon)**

In [ ]:
```
def bonjour():
    print("Bonjour!")
```

I.2) Fonctions prédéfinies

Dans Python, un certain nombre de fonctions sont déjà prédéfinies. Elles peuvent donc être utilisées directement sans avoir à les redéfinir. Quelques unes de ces fonc- tions ont déjà été évoquées dans ce cours (print,input). Le tableau suivant contient une liste de fonctions prédéfinies en Python qui pourront vous servir dans le cadre de ce cours. Cette liste n'est bien évidemment pas exhaustive et vous pourrez découvrir les autres fonctions prédéfinies en Python en consultant la documentation disponible via le lien suivant : https://docs.python.org/3/

Fonction	Description

`print(x)` | Affiche la valeur de `x` dans la console `input(txt)` | Affiche le texte de la chaîne de caractères `txt` dans la console. L'utilisateur doit alors taper une valeur dans la console qui sera renvoyée par la fonction `input` dès que l'utilisateur appuie sur la touche *Entrée* du clavier. `int(x)` | Renvoie la valeur de `x` convertie en entier lorsque cela est possible. `float(x)` | Renvoie la valeur de `x` en float quand c'est possible `len(x)` | Renvoie le nombre de d'objets (ou de caractères) de `x` `abs(x)` | Renvoie la valeur absolue de `x`

> **Exemple 5:**
>
> Dans le code suivant, on utilise les fonctions `float,input,print,round`

In [1]:
```
x = float(input("Veuillez rentrer une valeur pour x"))
print(f"La valeur arrondi de x vaut {round(x)}")
```

```
La valeur arrondi de x vaut 14
```

Remarquez l'encapsulation de la fonction input dans la fonction `float`. Ceci est nécessaire car la fonction `input` ne permet que de récupérer le texte taper par l'utilisateur (même si ce texte comporte des nombres). Ainsi, si l'on n'utilise pas la

fonction `float` et que l'utilisateur rentre la valeur `123.3` avant de valider par la touche Entrée, c'est la chaîne de caractère `"123.3"` qui sera stockée dans la variable `x`. La fonction `float` permet alors de convertir la chaîne de caractère `"123.3"` en `123.3` qui est un nombre flottant.

II) Bibliothèques

II.1) Importer des bibliothèques

En plus des fonctions prédéfinies qui sont accessibles directement, Python propose également un certain nombres de bibliothèques (aussi appelés modules) qui contiennent de nombreuses fonctions déjà programmées (ainsi que d'autres objets prédéfinis). Pour utiliser les fonctions d'une bibliothèque, il faut d'abord importer la bibliothèque. La manière la plus simple pour ce faire est d'utiliser la commande import suivi du nom de la bibliothèque.

> **Exemple 6:**
>
> L'instruction suivante permet d'importer la bibliothèque standard math.
> `import math`
>
> Pour utiliser une fonction d'une bibliothèque (ou n'importe quel autre type d'objet d'une bibliothèque), il suffit alors d'utiliser la syntaxe
> `nom_bibliotheque.nom_fonction`

> **Exemple 7:**
>
> Le code suivant permet d'importer la bibliothèque math dans Python pour utiliser la fonction sqrt qui permet de calculer une racine carrée 2. Ici, on affiche la valeur de $\sqrt{2}$
>
> ```
> import math
> print(math.sqrt(2))
> ```
> Une autre façon d'importer les fonctions et objets d'une bibliothèque est d'utiliser une instruction suivant la syntaxe ci-dessous :
>
> ```
> from nom_bibliotheque import *
> ```
> Cette instruction permet d'importer toutes les fonctions et objets d'une bibliothèque de manière à ce qu'ils puissent être utilisés sans rappeler le nom de la bibliothèque.

II.2) Fonctions aléatoires

Le langage Python permet également la définition de **fonctions aléatoires**. En informatique, une fonction aléatoire est une fonction dont la valeur renvoyée est le résultat d'un tirage aléatoire. Cette valeur n'est donc pas déterminée de manière unique

par la valeur des paramètres en entrée de la fonction. La bibliothèque random en Python contient un certain nombre de fonctions aléatoires prédéfinies et notamment la fonction randint qui renvoie un entier compris entre deux valeurs fournies en paramètres de manière aléatoire et selon une loi de probabilité uniforme 3.

> **Exemple 8:**
>
> Dans le code suivant, on importe les fonctions de la bibliothèque `random` afin de tirer au hasard soir 0, soit 1.
>
> ```
> import random as rd
> print(rd.randint(0,1))
> ```

Simulation

Une **simulation** est un processus qui calque un phénomène réel dans le but d'en prédire l'issue. Le résultat d'une simulation peut ainsi informer sur le résultat possible du phénomène. On utilise généralement les simulations en informatique pour recréer un phénomène qui coûterait trop cher à réaliser physiquement (comme pour une simulation de crash-test d'avion) ou qui est tout simplement impossible à réaliser phy- siquement (comme pour une simulation de l'évolution climatique du globe terrestre).

> **Exemple 10:** On peut utiliser la fonction `randint` comme précédemment pour simuler le lancer d'une pièce qui peut tomber sur pile (que l'on fait correspondre au 0) ou face (que l'on fait correspondre au 1). Si un lancer de pièce est facile à réaliser physiquement, le lancer d'un million de pièces par exemple devient plus compliqué alors qu'en simulation cela ne met que quelques secondes sur un ordinateur moderne. Dans le code Python suivant, une simulation permet de comptabiliser combien de fois une pièce tombe sur face en un million de lancers. C'est la variable p qui permet de stocker cette valeur.

In [1]:
```
import random as rd
p = 0
for k in range(1000000):
    p = p + rd.randint(0,1)
```

III) Exercices

Exercice 1

Déterminez les renvois des fonctions f et g définies ci-dessous pour les appels $f(2), f(0), g(1)$ et $g(3)$.

In [ ]:
```
def f(x):
    y = x**2-3
    return y
```

```
def g(x):
    y = x**2 - 3
    z = 3*x -y
    y = z + x
    return z
```

Exercice 2

On considère les trois fonctions Python suivantes.

Détaillez les calculs de $f1(5,1)$, $f2(5,1)$ et $f3(5,1)$. Vous listerez l'ensemble de toutes les affectations qui sont réalisées en indiquant si celles-ci ont lieu :

- avant le début de la boucle ;
- dans la boucle et, dans ce cas, au combientième tour de la boucle ;
- après la sortie de la boucle.

In [ ]:
```
def f1(x,y):
    z = 0
    while z<20:
        z = x+y
        x = 2*x
        y = 3*y
        z = y*z
    return z

def f2(x,y):
    z = 0
    while z < 20:
        z = x + y
        x = 2*x
        y = 3*y
    x = y*z
    return z

def f3(x,y):
    z = 0
    while z < 20:
        z = x + y
    x = 2*x
    y = 3*y
    return z
```

Exercice 3

Le tableau suivant donne la mention en fonction de la note N obtenue à la première session du baccalauréat.

Condition	Mention
$0 \leq N < 8$	Ajournée

Condition	Mention
$8 \leq N < 10$	Rattrapage
$10 \leq N < 12$	Sans Mention
$12 \leq N < 14$	Assez Bien
$14 \leq N < 16$	Bien
$N \geq 16$	Très bien

Définissez une fonction Python qui prend une note en paramètre et renvoie la mention correspondante sous forme de chaîne de caractère.

Exercice 4

1. Recopiez et complétez le code suivant de la fonction `lancer` pour qu'elle simule le lancer d'un dé équilibré à six faces.

```
import random as rd

def lancer():
    return rd.randint(...,...)
```

2. On souhaite réaliser une simulation du lancer de 1000 dés. Complétez le code suivant et exécutez le à la suite du code précédent afin de pouvoir remplir le tableau des effectifs correspondant à la série statistique de la simulation.