Programmation structurée et fonctions

INFORMATIQUE COMMUNE - TP n° 1.2 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ indenter et enchaîner correctement les blocs Python
- ☞ coder une structure conditionnelle (et l'expression conditionnelle)
- ☞ coder une boucle avec un range sur l'intervalle d'entiers ouvert $[\![0, n[\![$
- ☞ coder une boucle tant que en précisant la condition de sortie
- ☞ passer d'un script à l'écriture d'une fonction
- ☞ coder et utiliser une fonction (paramètres formels, effectifs, retour)
- ☞ tester une fonction à l'aide d'un programme principal

A Des scripts aux fonctions

Pour résoudre les trois premiers exercices, on demande de créer :

1. dans un premier temps, un script qui affiche à l'écran le résultat,
2. puis, dans un deuxième temps, une fonction qui renvoie le résultat et un script qui appelle cette fonction.

Le code 1 est un exemple de script. Le code 2 est un exemple de script qui utilise une fonction.

Code 1 – Exemple de script

```
x = 3
sqrx = x * x
print("Squaring 3 gives ", {sqrx})
```

Code 2 – Exemple de fonction utilisée dans un script

```
def square(a):  # signature, a is a formal parameter
    x = a * a  # function body
    return x  # returned value

nine = square(3)  # 3 is an effective parameter (argument)
print("Squaring three gives ", nine)  # Squaring three gives 9.
```

Ⓡ On fera attention à ne pas utiliser de `print` dans les fonctions, réserver les `print` au programme principal.

A1. **Tarifs de la piscine.** À Brest, les tarifs d'entrée à la piscine en vigueur sont :

- Ticket individuel : 4,20 €,
- Tarif réduit : 2,75 €,
- Enfant 4 - 17 ans : 2,10 €,
- Enfant 0 - 3 ans : gratuit.

(a) Écrire un script qui affiche le tarif d'entrée de la piscine de la manière suivante : `4 ans --> Tarif : 2.1 euros`.
L'age est une variable du programme définie ainsi :

```
age = 4
```

(b) Améliorer ce script afin de prendre en compte les réductions éventuelles. Pouvez-vous utiliser une expression conditionnelle pour prendre en compte ces éléments ?

(c) Transformer ce script en une fonction Python. Le prototype de la fonction est `get_price(age, reduced)`. Le paramètre `age` est de type `int`, `reduced` est un paramètre de type `bool` et la fonction renvoie un type `float`.

(d) Tester la fonction à l'aide du programme principal suivant :

```
# valid parameters
  print("3 ans --> Tarif : ", get_price(3, False), " euros")
  print("26 ans --> Tarif : ", get_price(26, False), " euros")
```

(e) Modifier le programme principal afin de rendre les tests plus exhaustifs : tester des paramètres incongrus ou de type inadéquat.

Solution :

Code 3 – Piscine

```
def get_price(age, reduced):
    if age < 4:
        return 0.0
    elif 3 < age < 18:
        return 2.1
    else:
        return 2.75 if reduced else 4.2

# SCRIPT VERSION
age = 4
reduced = False
if age < 4:
    print(age, "ans --> Tarif : 0.0 euros")
elif 3 < age < 18:
    print(age, "ans --> Tarif : 2.1 euros")
else:
    if reduced:
        print(age, "ans --> Tarif : 2.75 euros")
    else:
        print(age, "ans --> Tarif : 4.2 euros")

# FUNCTION VERSION
# valid parameters
```

```
print("2 ans --> Tarif : ", get_price(2, False), " euros")
print("3 ans --> Tarif : ", get_price(3, False), " euros")
print("4 ans --> Tarif : ", get_price(4, False), " euros")
print("10 ans --> Tarif : ", get_price(10, False), " euros")
print("17 ans --> Tarif : ", get_price(17, False), " euros")
print("18 ans --> Tarif : ", get_price(18, False), " euros")
print("26 ans --> Tarif : ", get_price(26, True), " euros")
print("26 ans & not reduced --> Tarif : ", get_price(26, False), " euros")

# exotic parameters (tests for robustness)
print("3 ans & reduced --> Tarif : ", get_price(3, True), " euros")
print("3 ans & not reduced --> Tarif : ", get_price(3, False), " euros")
print("10 ans & reduced --> Tarif : ", get_price(10, True), " euros")
print("10 ans & not reduced --> Tarif : ", get_price(10, False), " euros")

# wrong parameters (tests for robustness)
print("3.5 ans --> Tarif : ", get_price(3.5, False), " euros")
print("11.5 ans --> Tarif : ", get_price(11.5, False), " euros")
print("26.335 ans --> Tarif : ", get_price(11.5, False), " euros")
```

A2. **Horaires des bus.** La fréquence de passage des bus varie au cours de la semaine : le week-end, ils passent toutes les 30 minutes. Les autres jours de la semaine, ils passent toutes les 10 minutes en heure creuse et toutes les 5 minutes en heure d'affluence (entre 7h et 9h puis entre 16h et 18h).

(a) Écrire un script qui calcule la fréquence de passage des bus en fonction de l'heure et du jour de la semaine.

(b) Transformer ce script en fonction Python. Le prototype de la fonction est `get_frequency(day, hour)`. `day` est de type `str`, `hour` est de type `int`. Cette fonction renvoie un type `int`.

(c) Tester la fonction à l'aide du programme principal suivant :

```
print("Fréquence : ", get_frequency("dimanche", 5))
print("Fréquence : ", get_frequency("lundi", 12))
print("Fréquence : ", get_frequency("mardi", 8))
print("Fréquence : ", get_frequency("jeudi", 17))
```

(d) Modifier le programme principal afin de rendre les tests plus exhaustifs : tester des paramètres incongrus ou de type inadéquat.

(e) Comment devrait se comporter la fonction `get_frequency` si les bus ne circulent pas entre 1h et 4h du matin ?

Solution :

Code 4 – Bus

```
def get_frequency(day, hour):
    if day == "Samedi" or day == "Dimanche":
        return 30
    #elif 1 <= hour <= 4:
    #    raise Exception("Le bus ne circule pas !")
    else:
        if 7 <= hour < 9 or 16 <= hour < 18:
```

```
            return 5
        else:
            return 10

#  SCRIPT VERSION
day = "Lundi"
hour = 15
if day == "Samedi" or day == "Dimanche":
    print(day, hour, " --> le bus passe toutes les 30 minutes.")
elif 1 <= hour < 4:
    print(day, hour, " --> le bus ne circule pas.")
else:
    if 7 <= hour < 9 or 16 <= hour < 18:
        print(day, hour, " --> le bus passe toutes les 5 minutes.")
    else:
        print(day, hour, " --> le bus passe toutes les 10 minutes.")

#  FUNCTION VERSION

days = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "
    Dimanche"]
hours = [h for h in range(24)]

for day in days:
    for hour in hours:
        print(day, hour, " --> Le bus passe toutes les ", get_frequency(day,
            hour), " minutes.")

print("Mercredi", 7.75, " --> Le bus passe toutes les ", get_frequency("
    Mercredi", 7.75), " minutes.")
```

A3. **Constante de Champernowne** La constante de Champernowne est un nombre réel univers qui se construit à partir de la suite croissante des entiers naturels :

$$C = 0,12345678910111213\ldots$$

(a) Écrire un script Python qui affiche les n premières décimales de ce nombre sous la forme suivante : `Champernowne constant 5 --> 0.12345`

(b) Transformer ce script en fonction Python. Le prototype est `champernowne(n)`. `n` est un type `int` et la fonction le nombre de Champernowne sous la forme d'une chaîne de caractère `str`.

(c) Tester la fonction dans un programme principal. Le test fera apparaître tous les nombres comportant de 1 à 20 décimales.

(d) Combien de décimales faut-il calculer pour voir apparaître la séquence "2022" dans la constante de Champernowne?

Solution : 499

Solution :

Code 5 – Champernowne

```
def champernowne(n):
    s = ""
    i = 1
    while len(s) < n + 1:
        s += str(i)
        i += 1
    return "0." + s[:n]



# SCRIPT VERSION
n = 1
s = "0."
i = 1
while len(s) < n + 2:
    s += str(i)
    i += 1
print("Champernowne ", n, " --> ", s)

# FUNCTION VERSION TEST
for i in range(1, 1000):
    print("Champernowne ", i, " --> ", champernowne(i))
    assert i == len(champernowne(i)) - 2
    if "2022" in champernowne(i):
        print("Found --> ", i, champernowne(i))
        break

# OTHER VERSION
i = 0
while not ("2022" in champernowne(i)):
    i = i +1
print("Found --> ", i, champernowne(i))
```

B Des fonctions qui renvoient un résultat

R À partir de maintenant, on n'écrira plus que des fonctions pour résoudre les exercices. Les `print` se feront dans le programme principal sauf nécessité de débogage.

B1. **Suites** On s'intéresse aux suites $(u_n)_{n\in\mathbb{N}}$ définie par $u_n = 2^{\sqrt{n}}$, et $(v_n)_{n\in\mathbb{N}}$ définie par $v_n = \sqrt{2^n}$.

(a) À l'aide du module `math`, créer une fonction qui renvoi la valeur de la suite u_n à un indice donné. Le prototype de la fonction est `suite_u(n)` où `n` est un paramètre de type `int`. La fonction renvoie un type `float`.

(b) Créer une fonction qui renvoi la valeur de la suite v_n à un indice donné. Le prototype de la fonction est `suite_v(n)` où `n` est un paramètre de type `int`. La fonction renvoie un type `float`.

(c) On souhaite à présent comparer les deux suites. Créer une fonction qui renvoie la différence de ces deux suites pour un indice n donné. Cette fonction a pour prototype `diff(n)` où `n` est un paramètre de type `int`. Elle renvoie un type `float`.

(d) Dans le programme principal, en utilisant une boucle, faire afficher la différence des deux suites pour n variant de 0 à 15 **inclus**. Pouvez-vous justifier ces résultats ?

Solution :

Code 6 – Suites

```
from math import sqrt


def suite_u(n):
    return 2 ** sqrt(n)


def suite_v(n):
    return sqrt(2 ** n)


def diff(n):
    return suite_u(n) - suite_v(n)


for i in range(16):
    print("u_", i, " - v_", i, " = ", diff(i))
```

B2. **Sommes**

(a) Écrire une fonction qui renvoie le résultat de $\sum_{k=1}^{10^n} \frac{1}{k}$. Cette fonction a pour prototype `somme(n)` où `n` est un paramètre de type `int`. Elle renvoie un type `float`. Tester la fonction dans un programme principal.

(b) Écrire une fonction qui renvoie le résultat de $\left(\sum_{k=1}^{10^n} \frac{1}{k}\right) - \ln(10^n)$. Cette fonction a pour prototype `diff(n)` où `n` est un paramètre de type `int`. Elle renvoie un type `float`. Tester la fonction dans un programme principal en utilisant une boucle pour n variant de 1 à 7 **inclus**. Pourrait-on calculer cette différence pour des valeurs de n beaucoup plus grandes ?

Solution : La durée du calcul de somme s'allonge exponentiellement avec n. On ne peut guère espérer calculer somme pour des n beaucoup plus grands dans un temps raisonnable.

Code 7 – Sommes

```
from math import log


def somme(n):
    acc = 0
    for i in range(1, 10 ** n + 1):
        acc += 1 / i
    return acc
```

```
def diff(n):
    return somme(n) - log(10 ** n)


for i in range(1, 22):
    print("#", i, " --> ", diff(i))
```

C Console art, génération de dessin avec des caractères

Un caractère (une lettre) est codé en machine par un certain nombre prédéterminé figurant dans la table ASCII. On peut connaître le code associé à un caractère à l'aide de la commande `ord`. Par exemple, `ord('a')` renvoie la valeur 97 en décimal, ce qui, codé en binaire sera stocké en mémoire sous la forme `1100001`. Certains caractères sont spéciaux, par exemple le caractère de contrôle pour revenir à la ligne ou celui pour effectuer une tabulation. Le retour à la ligne (Line Feed ou LF) est noté `"\n"` et il faut noter qu'il ne représente qu'un seul caractère.

C1. **Console art.**

On cherche à dessiner des motifs carrés de dimension quelconque sur la console Python avec des caractères. Pour cela, on va créer des fonctions Python dont le paramètre est la taille du carré `n` de type `int` et qui renvoie le motif sous la forme d'une chaîne de caractère.

(a) Créer une fonction de prototype `block(n)` qui renvoie une chaîne de caractères représentant un carré de côté n délimité par des étoiles contenant des points. Par exemple pour $n = 5$.

```
* * * * *
* . . . *
* . . . *
* . . . *
* * * * *
```

(b) Créer une fonction de prototype `square_diag(n)` qui renvoie une chaîne de caractères représentant les deux diagonales du carré avec des étoiles et le reste du carré par des points. Par exemple pour $n = 5$.

```
* . . . *
. * . * .
. . * . .
. * . * .
* . . . *
```

(c) Créer une fonction de prototype `losange(n)` qui renvoie une chaîne de caractères représentant un losange avec des étoiles et le reste du carré par des points. On veillera à ce que le paramètre d'entrée n soit impair. Les opérateurs division entière et modulo sont utiles! Par exemple pour $n = 5$.

```
. . * . .
. * . * .
* . . . *
. * . * .
```

```
. . * . .
```

Solution :

Code 8 – Console Art et procédures

```
def block(n):
    s = ""
    for row in range(n):
        for col in range(n):
            if row == 0 or row == n - 1 or col == 0 or col == n - 1:
                s = s + "* "
            else:
                s = s + ". "
        s = s + "\n"
    return s


def square_diag(n):
    s = ""
    for row in range(n):
        for col in range(n):
            if row - col == 0 or row + col == n - 1:
                s = s + "* "
            else:
                s = s + ". "
        s = s + "\n"
    return s


def losange(n):
    assert n % 2 == 1
    s = ""
    for row in range(n):
        for col in range(n):
            if (row + col) % (n - 1) == n // 2 or abs(row - col) == n // 2:
                s = s + "* "
            else:
                s = s + ". "
        s = s + "\n"
    return s


print(block(5))
print(square_diag(5))
print(losange(5))
```