Lycée César Baggio — PTSI — 2021/2022
Devoir de contrôle d'informatique
Durée 2h

- Sans document, sans calculatrice ou autre dispositif électronique.
- Lorsque l'on demande d'écrire un code `Python`, on prendra soin de numéroter les lignes et de faire apparaître très clairement l'indentation.

Exercice 1. Déterminer la valeur retournée par chacun des appels demandés.

```
1 def f(x):
2     if x % 2 == 0:
3         return 3
4     if x % 4 == 0:
5         return 5
6     else:
7         return 7
```

1. `f(3)`
2. `f(6)`
3. `f(8)`
4. `f(1)`

Exercice 2. Déterminer la valeur retournée par chacun des appels demandés.

```
1 def f(x):
2     if x % 2 == 0:
3         a = 3
4     if x % 4 == 0:
5         a = 5
6     else:
7         a = 7
8     return a
```

1. `f(3)`
2. `f(6)`
3. `f(8)`
4. `f(1)`

Exercice 3. Déterminer la valeur retournée par chacun des appels demandés.

```
1 def f(x):
2     if x % 2 == 0:
3         a = 3
4     elif x % 4 == 0:
5         a = 5
6     else:
7         a = 7
8     return a
```

1. `f(3)`
2. `f(6)`
3. `f(8)`
4. `f(1)`

Exercice 4. Expliquer pourquoi les instructions suivantes renvoient des messages d'erreur.

1.

```
def f(x):
    a = 0
    if x % 2 == 0:
        a = 0
    else x % 4 == 0:
        a = 1
    return a
```

2.

```
def g(x):
    a = 0
    if x % 2 = 0:
        a = 0
    else:
        a = 1
    return a
```

Exercice 5. On donne la fonction `lapin`

```
def lapin(n):
    a = 0
    b = n + 1
    while a + 1 < b:
        x = (a + b) // 2
        if x * x <= n:
            a = x
        else:
            b = x
        print(a, b, x)
    return b - 1
```

1. Donner les valeurs affichées par l'instruction `print` lors de l'appel `lapin(52)`.

2. Quelle valeur retourne la fonction lors de l'appel `lapin(52)` ?

3. Donner les valeurs affichées par l'instruction `print` lors de l'appel `lapin(0)`.

4. Quelle valeur retourne la fonction lors de l'appel `lapin(0)` ?

Exercice 6. Si x et y sont deux nombres réels quelconques, on définit une fonction prenant comme variables x et y de la façon suivante

$$f(x,y) = \begin{cases} x^2 + y - 3 & \text{si } x > y \\ x^2 - y^2 + 12 & \text{si } x < y \\ x - 1 & \text{si } x = y. \end{cases}$$

1. Écrire une fonction `f` prenant comme paramètres deux flottants («nombres à virgule») `x` et `y` et calculant $f(x,y)$.
2. On définit une suite $\left(u_n\right)_{n\in\mathbb{N}}$ par $u_0 = 0$ et la relation

$$\forall n \in \mathbb{N}, u_{n+1} = f\left(u_n^2, u_n\right).$$

   Écrire une fonction `carotte` ayant pour argument un entier naturel `n` et que retourne la valeur de u_n.

Exercice 7.

1. Écrire une fonction `nb_chiffres` ayant pour arguments deux entiers `n` et `b` avec $n \geq 1$ et $b \geq 2$ et qui retourne l'entier p tel que

$$b^{p-1} \leq n < b^p.$$

   On utilisera uniquement les opérations sur les entiers sans import de bibliothèque.
2. Écrire un ensemble de tests fonctionnels pour votre fonction `nb_chiffres`.

Exercice 8. Écrire une fonction `seconddegre` avec trois paramètres `a`, `b`, `c`, et qui renvoie la liste contenant les racines réelles du trinôme du second degré

$$aX^2 + bX + c$$

- Préconditions : `a`, `b`, `c` sont trois flottants et `a` est non nul.

On obtiendra par exemple

Python 3 Shell

```
>>> seconddegre(1, -5, 6)
[2.0, 3.0]
>>> seconddegre(1, -2, 1)
[1.0]
>>> seconddegre(1, 1, 1)
[]
```

Exercice 9. On donne trois nombre a, b, c et on souhaite les ordonner dans l'ordre croissant. Voici donc mon programme:

```
1  def tri(a, b, c):
2      """
3      Retourne la liste triée contenant les trois nombres a, b, c
4      """
5      if a > b:
6          x = a
7          a = b
8          b = x
9      if b > c:
10         x = b
11         b = c
12         c = x
13     if a > c:
14         x = a
15         a = c
16         c = x
17
18     return [a, b, c]
```

Je l'essaie avec $a = 3$, $b = 1$ et $c = 2$.

Python 3 Shell

```
>>> tri(3, 1, 2)
[1, 2, 3]
```

1. Que font les lignes 5-6-7-8 dans le programme précédent ?
2. Écrire un ensemble de tests fonctionnels «crédible» `test_tri` pour ma fonction de tri.
3. Lesquels de vos tests échouent ? Expliquez ?

Exercice 10. Prenons un nombre entier naturel n non nul. Calculons la somme des carrés de ses chiffres. On obtient un nouveau nombre. Recommençons cette opération avec ce nouveau nombre, on obtient un troisième nombre.

Que se passe-t-il si on itère ces opérations plusieurs fois? Observons deux exemples.

- Avec $n = 19$, on obtient successivement

$$19 \to 82 \to 68 \to 100 \to 1 \to 1 \dots$$

et le 1 se répète indéfiniment.

- Avec $n = 200$, on obtient successivement

$$200 \to 4 \to 16 \to 37 \to 58 \to 89 \to 145 \to 42 \to 20 \to 4 \dots$$

et le cycle $4 \to 16 \to 37 \to 58 \to 89 \to 145 \to 42 \to 20$ se répète indéfiniment.

On peut démontrer qu'en appliquant un tel processus, à partir d'un entier quelconque non nul, on finit toujours par arriver sur l'une ou l'autre de ces deux situations, c'est-à-dire à arriver sur le nombre 1 ou sur le nombre 4.

On dira qu'un nombre entier naturel non nul n est **heureux** si le processus mène à 1, et **malheureux** s'il mène à 4.

1. Quelles sont les opérateurs Python permettant d'obtenir le quotient et le reste de la division euclidienne d'un entier `n` par un autre entier `p`.

2. Écrire une fonction `somme_carres` qui renvoie la somme des carrés des chiffres de l'entier passé en paramètre.

Python 3 Shell

```
>>> somme_carres(19)
82
>>> somme_carres(200)
4
```

3. Après avoir pris soin de bien numéroter les lignes de votre fonction `somme_carres`, simuler l'exécution de l'appel `somme_carres(3407)`. On présentera l'évolution des valeurs prises par vos variable sous forme d'un tableau. Préciser à chaque fois le numéro de la ligne exécutée.

4. Réalisez un prédicat (c'est-à-dire une fonction qui retourne un booléen) nommé `est_heureux` qui renvoie `True` si l'entier passé en paramètre est heureux, et `False` sinon.

Python 3 Shell

```
>>> est_heureux(19)
True
>>> est_heureux(200)
False
```

5. On appelle **altitude**, le terme maximum atteint par la suite obtenue à partir de l'entier *n*. Par exemple, l'altitude de 19 est 100 et l'altitude de 200 est 200.

   Écrire une fonction `altitude` ayant pour paramètre un entier naturel `n` et qui retourne l'altitude partant de `n`.

Python 3 Shell

```
>>> altitude(19)
100
>>> altitude(200)
200
```