EXERCICES

Exercices sur les instructions conditionnelles

S'entraîner à lire des branchements et à écrire des fonctions avec des instructions conditionnelles en Python.

Exercice 1

Exercice 1

1. Dessiner le schéma de branchement de la fonction `divise()` décrite ci-dessous.
2. Écrire une fonction `divise(a: int, b: int) -> float` qui prend en paramètres deux entiers `a` et `b` et renvoie le résultat de la division de `a` par `b`.

Cette fonction devra effectuer la division uniquement si `b` est différent de `0`.

PYTHON
```
>>> divise(6, 3)
2.0
```

□ Exercice 1 terminé.

Exercice 2

Exercice 2

Considérons le programme suivant :

PYTHON
```
if n > 2:
    n = n + 10
    res = (n // 2) + n
else:
    res = n * n
res = res - n
```

1. Faire la trace d'exécution du programme précédent :

Exercice 2

- une première fois avec `n` égal à `1` ;
- une seconde fois avec `n` égal à `3`.

2. Dessiner son schéma de branchement.

☐ Exercice 2 terminé.

Exercice 3

Exercice 3

Écrire une fonction `est_majeur(age: int) -> bool` qui prend en paramètre un âge entier et renvoie `True` si cet âge est supérieur ou égal à `18`, et `False` sinon.

Cette fonction doit comporter une instruction conditionnelle.

PYTHON
```
>>> est_majeur(16)
False
>>> est_majeur(18)
True
```

☐ Exercice 3 terminé.

Exercice 4

Exercice 4

Écrire une fonction `max2(a: int, b: int) -> int` qui prend en paramètres deux entiers `a` et `b` et renvoie l'entier le plus grand.

PYTHON
```
>>> max2(5, 13)
13
```

☐ Exercice 4 terminé.

Exercice 5

Exercice 5

En utilisant la fonction `randint()` du module `random`, écrire une fonction `max_alea(n: int) -> bool` qui prend en paramètre un entier `n` et renvoie `True` si `n` est plus grand que l'entier aléatoire créé, compris entre `1` et `10`.

□ Exercice 5 terminé.

Exercice 6

Exercice 6

Écrire une fonction `max3(a: int, b: int, c: int) -> int` qui prend en paramètres trois entiers `a`, `b` et `c` et renvoie l'entier le plus grand.

PYTHON
```
>>> max3(5, 7, 15)
15
```

□ Exercice 6 terminé.

Exercice 7

Exercice 7

Écrire une fonction `ordre_croissant(a: int, b: int, c: int) -> str` qui prend en paramètres trois entiers `a`, `b` et `c` et renvoie une chaîne de caractères indiquant leur ordre croissant.

PYTHON
```
>>> ordre_croissant(5, 13, 4)
'c < a < b'
```

□ Exercice 7 terminé.

Exercice 8 — Difficile

Exercice 8

Rappel n° 1 : dans un triangle, aucune longueur n'est strictement supérieure à la somme des deux autres.

Rappel n° 2 : un triangle équilatéral possède trois longueurs égales et un triangle isocèle en possède deux.

1. Écrire une fonction `test_triangle(a: int, b: int, c: int) -> str` qui prend en paramètres trois entiers représentant les longueurs d'un triangle et renvoie une chaîne de caractères.

La fonction doit renvoyer `isocele` s'il s'agit d'un triangle isocèle, `equilateral` s'il s'agit d'un triangle équilatéral ou `ce nest pas un triangle` s'il ne s'agit pas d'un triangle.

2. Modifier la fonction `test_triangle()` en ajoutant le fait qu'il peut s'agir d'un triangle rectangle. On pourra utiliser la fonction `test_pythagore()` pour cela.

Rappel pour cette fonction : `test_pythagore(a: int, b: int, c: int) -> bool` prend en paramètres trois entiers `a`, `b` et `c` et renvoie `True` si $a^2 + b^2 = c^2$, et `False` sinon.

□ Exercice 8 terminé.