LMI 1 - Semestre 1

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO1 – Algorithmique et Programmation 1 – Groupe 4

Fiche de TD numéro 1 - Fonctions - Conditionnelle - Boucles

Exercice 1 : Décrivez ce qu'il se passe pour chaque instruction ci-dessous :
- dites si une instruction ne vous semble pas correcte
- sinon, donnez la valeur de chacune des variables de l'environnement

```
a = 3
b = 12.5
a = 2 * b
2a = 2 * a
double_a = 2 * a
multiple_b = 5b
b + 1 = 15
b = 3 + double_a
moyenne = (a + b + double_a) / 3
moyenne = (a + b + double_a) % 3
moyenne = (a + b + double_a) // 3
```

Exercice 2 : Spécifier puis écrire en Python une fonction `conversion_heures` qui prend en paramètre deux entiers qui représentent une durée exprimée en nombre de jours et nombre d'heures, et retourne le nombre d'heures total de cette durée.

```
>>> conversion_heures(12,3) # 12 jours et 3 heures
291
```

Exercice 3 : Spécifier puis écrire en Python une fonction `conversion_minutes` qui prend en paramètre trois entiers qui représentent une durée exprimée en nombre de jours, nombre d'heures et nombre de minutes, et retourne le nombre de minutes total de cette durée.

```
>>> conversion_minutes(10,5,45) # 10 jours, 5 heures et 45 minutes
14745
```

Exercice 4 : Spécifier puis écrire
- une fonction `heures` qui prend en paramètre un entier qui représente une durée exprimée en minutes et retourne le nombre d'heures de cette durée
- une fonction `jours` qui prend en paramètre un entier qui représente une durée exprimée en heures et retourne le nombre de journées de cette durée

```
>>> heures(185)
3
>>> jours(46)
1
```

Exercice 5 : Spécifier puis écrire en Python une fonction `conversion_jhm` qui prend en paramètre une durée en minutes et retourne cette durée en jours, heures et minutes. Quel est le type du retour ?

```
>>> conversion_jhm(14745) # équivalent à 10 jours, 5 heures et 45 minutes
(10, 5, 45)
```

Exercice 6 : Spécifier puis écrire en Python une fonction `ajout_durees` qui prend en paramètres deux durées en exprimées en jours, heures et minutes, et retourne la somme de ces deux durées.

```
>>> ajout_durees((12,3,45),(10,5,52))
(22, 9, 37)
>>> ajout_durees((12,20,45),(10,5,52))
(23, 2, 37)
```

Exercice 7 : Soit l'environnement suivant :

```
a = 6 < 5
b = 3
```

Évaluez les expressions booléennes suivantes :

```
not(a) and not(b+1<3))
False or (a and not(not(a) or not(b > 0)))
```

Exercice 8 : Simplifiez (le cas échéant) puis écrivez la table de vérité des expressions suivantes (ici, `a` et `b` sont des variables booléennes) :

```
not(not(a and b) or False)
not(not(not (a) and b) and not(a and not(b)))
not(a) or b
```

Exercice 9 : Spécifiez puis écrivez une fonction `multiple` qui prend deux valeurs entières en paramètres et teste si la première est un multiple de l'autre.

Exemples :

```
>>> multiple(56,4)
True
>>> multiple(58,4)
False
```

Exercice 10 : Spécifiez puis écrivez une fonction `annee_bissextile` qui prend le numéro d'une année et retourne `True` si cette année est bissextile, `False` sinon. Une année est bissextile si elle est un multiple de 4, sauf les siècles (1900 n'est pas bissextile), sauf tous les 400 ans (1600 est bissextile).

Exemples :

```
>>> annee_bissextile(2020)
True
>>> annee_bissextile(2021)
False
>>> annee_bissextile(2000)
True
>>> annee_bissextile(2100)
False
```

Exercice 11 : Spécifiez puis écrivez une fonction `nb_jours_annee` qui retourne le nombre de jours de l'année passée en paramètre.

Exemples :

```
>>> nb_jours_annee(2020)
366
>>> nb_jours_annee(2021)
365
```

Exercice 12 : Spécifiez puis écrivez une fonction `nbre_jours_mois` qui prend en paramètre un numéro de mois et une année et retourne le nombre de jours de ce mois.

Exemples :

```
>>> nbre_jours_mois(2,2020)
29
>>> nbre_jours_mois(2,2021)
28
>>> nbre_jours_mois(7,2021)
31
>>> nbre_jours_mois(9,2021)
30
```

Exercice 13 : Spécifiez puis écrivez une fonction `fact` qui prend un entier `n` positif en paramètre et retourne `n!`.

Exemples :

```
>>> fact(5)
120
>>> fact(0)
1
>>> fact(1)
1
```

Exercice 14 : Spécifiez puis écrivez une fonction `plus_petit_diviseur` qui prend en paramètre un entier positif et retourne le plus petit de ses diviseurs supérieur à 1.

Exemples :

```
>>> plus_petit_diviseur(4)
2
>>> plus_petit_diviseur(11)
11
```