Contrôle 

NOM et PRÉNOM :

Barème sur 30 points.

Exercice 1 (3 points) –

On définit une liste de tuples comme suit :

```
L = [(12, 'Amy', (3,45,21), 'zarby'),
     (3, 'Bastien', (3,21,34), 'notgood'),
     (7, 'Zoé', (2, 53, 7), 'bad'),
     (42, 'Astrée', (3,4,51), 'global'),
     (666, 'Lucifer', (3,4,51), 'Méphisto'),
    ]
```

1. Quelle est la valeur de len(L) ? .....
2. Quelle est la valeur de L[1] ? .....
3. Quelle est la valeur de L [2][0] ? .....
4. Quelle est la valeur de len(L[0]) ? .....
5. Quelle est la valeur de len(L [0][2]) ? .....

Résolution.

1. Quelle est la valeur de len(L) ? 5 (c'est le nombre de tuples éléments de la liste)
2. Quelle est la valeur de L[1] ? (3, 'Bastien', (3,21,34), 'notgood')
3. Quelle est la valeur de L [2][0] ? 7
4. Quelle est la valeur de len(L[0]) ? 4
5. Quelle est la valeur de len(L [0][2]) ? 3

Exercice 2 (3 points) –

1. Avec le code suivant :

```
L = ['a', 'b', 'c', 'd']
for i, c in enumerate(L):
    print(i,  c)
```

quel sera l'affichage obtenu ?

2. Avec le code suivant :

```
L = ['a', 'b', 'c', 'd']
for i, c in enumerate(L):
    if i < len(L)-1:
        print(i, L[i+1])
```

quel sera l'affichage obtenu ?

Résolution.
On obtient :

```
0 a
1 b
2 c
3 d
```

puis

```
0 b
1 c
2 d
```

Exercice 3 (4 points) –
Complétez le corps de la fonction sommeElementsImpairs spécifiée ci-dessous :

```
def test_sommeElementsImpairs():
    assert sommeElementsImpairs([2, 3, 5, 4]) == 8
    assert sommeElementsImpairs([2, 4, 88, 100]) == 0
    assert sommeElementsImpairs([3, 7, 1]) == 11
    assert sommeElementsImpairs([]) == 0


def sommeElementsImpairs(tab):
    """
    tab est une liste d'entiers
    renvoie la somme des éléments impairs de cette liste
    """
    ..................................................
    ..................................................
    ..................................................
    ..................................................
    ..................................................
```

Résolution.
Un code possible :

```
def test_sommeElementsImpairs():
    assert sommeElementsImpairs([2, 3, 5, 4]) == 8
    assert sommeElementsImpairs([2, 4, 88, 100]) == 0
    assert sommeElementsImpairs([3, 7, 1]) == 11
    assert sommeElementsImpairs([]) == 0


def sommeElementsImpairs(tab):
    """
    tab est une liste d'entiers
    renvoie la somme des éléments impairs de cette liste
    """
    somme = 0
    for nombre in tab:
        if nombre%2 == 1:
            somme += nombre
    return somme
```

Exercice 4 (3 points) –
Compléter la fonction suivante :

```
def f(tab):
    """
    tab est une liste d'entiers
    renvoie la sous-liste constituée des éléments de tab
    qui sont multiples de 3
    """









.
```

Résolution.

```
def f(tab):
    """
    tab est une liste d'entiers
    renvoie la sous-liste constituée des éléments de tab
    qui sont multiples de 3
    """
    return [x for x in tab if x%3 == 0]
```

Exercice 5 (3 points) –
On dispose d'une liste de tuples de la forme (nom, prénom, année de naissance).
Par exemple :

```
personnes = [ ('Labrosse', 'Adam', 2000),
('Gemlamorte', 'Adèle', 1985),
('Auboisdormant', 'Abel', 2001),
('Etpan', 'Ahmed', 1975),
('Térieur', 'Alain', 1999),
('Térieur', 'Alex', 1976),
('Tanrien', 'Jean', 2010),
('Ouzi', 'Jacques', 1950),
('Deuf', 'John', 2006),
('Provist', 'Alain', 2011)
]
```

Ecrire un code python créant la liste des couples (prénom, année de naissance) des personnes qui sont nées après 1992.
On créera cette liste en compréhension.

Résolution.
Un code :

```
[(prénom, an) for (_, prénom, an) in personnes if an > 1992]
```

Exercice 6 (4 points) –
Compléter le corps de la fonction ci-dessous (en choisissant – suivant votre préférence – entre le principe du compteur par accumulation ou le principe du décompte par la longueur d'une liste).

```
def test_lapparition():
    assert lapparition(['python', 'java']) == 0
    assert lapparition(['javascript', 'scheme', 'prolog']) == 1
    assert lapparition(['typescript', 'ada', 'processing']) == 2

def lapparition(tab):
    """
    tab est une liste de chaînes de caractères
    renvoie le nombre d'éléments de la liste tab qui contiennent au moins un 'e'
    """




.
```

Résolution.
Un code :

```
def lapparition(tab):
    """
    tab est une liste de chaînes de caractères
    renvoie le nombre d'éléments de la liste tab qui contiennent au moins un 'e'
    """
    return len([mot for mot in tab if 'e' in mot])
```

Exercice 7 (3 points) –
On considère la fonction python ci-dessous.

```
def f(tab):
    """
    tab -- liste d'entiers

    renvoie ....................................
    """
    m = tab[0]
    for element in tab:
        if element > m:
            m = element
    return m
```

1. Quelle est la valeur de f([3, 666, 42, 5, 7]) ?

2. Compléter la chaîne de documentation de la fonction.

Résolution.
renvoie l'élément le plus grand de tab.
La valeur de f([3, 666, 42, 5, 7]) est 666.

Exercice 8 (7 points, QCM) –
Pour chaque QCM, entourez la bonne réponse (il y a une et une seule bonne réponse pour chaque question).

QCM 1 – Après les lignes :

```
>>> A = [42, 1789, 666]
>>> B = A
>>> B[2] = 2021
```

a) A a pour valeur [42, 1789, 666]
b) A a pour valeur [42, 1789, 2021]
c) A a pour valeur [42, 2021, 666]
d) Une erreur a été déclenchée.

Résolution.
A a pour valeur [42, 1789, 2021].

QCM 2 – Après les lignes :

```
A = [666, 42, 1789]
B = [x for x in A]
B[2] = 2021
```

a) A a pour valeur [666, 42, 1789]
b) A a pour valeur [666, 42, 2021]
c) A a pour valeur [666, 2021, 1789]
d) Une erreur a été déclenchée.

Résolution.
A a pour valeur [666, 42, 1789]

QCM 3 – Après les lignes :

```
def f(tab):
    tab[0] = 1515

dates = [1789, 2021, 1969]
f(dates)
```

dates désigne la liste :
a) [1789, 2021, 1969]
b) [1515, 2021, 1969]
c) [1515, 1789, 2021, 1969]

Résolution.
[1515, 2021, 1969]

QCM 4 – Avec la fonction suivante :

```
def f(tab):
    cumul = 0
    accumule = 0
    for k, x in enumerate(tab):
        cumul = cumul + k
        accumule = accumule + x
    return cumul, accumule
```

la valeur de f ([2,3,4]) est :
a) (6,9) b) None c) (9,3) d) (3,9)

Résolution.
(3, 9)

QCM 5 – Avec le code suivant :

```
A = [x for x in range(20, 42) if x%2 == 0 and x%3 == 0]
```

la valeur de A est :
a) [20, 21, 22, 24, 26, 27, 28, 30, 32, 33, 34, 36, 38, 39, 40]
b) [24, 30, 36, 42] c) [24, 30, 36]

Résolution.
[24, 30, 36]

QCM 6 – Avec le code suivant :

```
A = [3,4,5]
B = [10,11,12]
C = [ A[i]+B[i] for i in (0,1,2)]
```

la valeur de C est :
a) [0, 1, 2]
b) [0, 2, 4] c) [13, 15, 17] d) le code déclenche une erreur

Résolution.
[13, 15, 17]

QCM 7 – Avec le code suivant :

```
A = [3, 4, -5, -3, 1, 7, -10, 0, 2]
B = [a for a in A if a > 0]
```

la valeur de B est :
a) [3, 4, 1, 7, 0, 2] b) [3, 4, 1, 7, 2]
c) [a, a, a, a, a] d) [−5, −3, −10]

Résolution.
[3, 4, 1, 7, 2]

Exercice 9 (bonus, hors barème) –
On dispose de listes construites comme définies ci-dessous.

- Liste de personnes constituée de tuples (identifiant, nom, prénom, année de naissance).

```
personnes = [ ('pers1', 'Labrosse', 'Adam', 2000),
('pers2','Gemlamorte', 'Adèle', 1985),
('pers3','Auboisdormant', 'Abel', 2001),
('pers4','Etpan', 'Ahmed', 1975),
('pers5','Térieur', 'Alain', 1999),
('pers6','Térieur', 'Alex', 1976),
('pers7','Tanrien', 'Jean', 2010),
('pers8','Ouzi', 'Jacques', 1950),
('pers9','Deuf', 'John', 2006),
('pers10','Provist', 'Alain', 2011)
]
```

- Liste de cours constituée de tuples (identifiant du cours, identifiant de personne, intitulé du cours). L'identifiant de personne est lié à la liste personnes précédente et désigne la personne qui enseigne ce cours.

```
cours = [('crs1', 'pers2', 'théorie des graphes'),
('crs2', 'pers4', 'programmation objet'),
('crs3', 'pers6', 'programmation fonctionnelle'),
('crs4', 'pers8', 'théorie des automates'),
('crs5', 'pers10', 'base de données relationnelle'),
('crs6', 'pers3', 'réseaux'),
('crs7', 'pers9', 'logique')
]
```

Lecture de la seconde table : on lit par exemple avec le dernier tuple ('crs7', 'pers9', 'logique') que le cours de logique est identifié par le code crs7 et est assuré par la personne d'identifiant 'pers9', c'est à dire par John Deuf.

1. Créer la liste des tuples (intitulé de cours, nom de l'enseignant dispensant ce cours).
2. Un enseignant est une personne de la liste personnes qui enseigne au moins un cours. Créer la liste des noms des enseignants.

Résolution.

1. Un code possible :

```
[(nom_cours,nom_personne) for (ip, nom_personne,_,_) in personnes
for (_, ip2,nom_cours) in cours
if ip == ip2]
```

2. Un code possible :

```
[nom_personne for (ip, nom_personne,_,_) in personnes
for (_, ip2,_) in cours
if ip == ip2]
```

Un autre code possible :

```
[nom for (ip, nom, _,_) in personnes if ip in [ident for (_, ident,_) in cours]]
```