LES FACULTÉS
DE L'UNIVERSITÉ
CATHOLIQUE DE LILLE

TD N°5 :
Programmation en Python
Les types structurés
QCM

I. QCM NIVEAU 1

Exercice N°1.

- Que fait le code suivant ? (un seul choix) :

```
texte = 'bonjour'
print(texte[1:4][1])
```

- produit une erreur à l'exécution
- affiche bonjour
- affiche b
- affiche o
- affiche n
- autre

Exercice N°2.

- Que fait le code suivant ? (un seul choix) :

```
texte = 'bonjour'
ma_liste = [texte, texte]
print(ma_liste)
```

- produit une erreur à l'exécution
- affiche [texte, texte]
- affiche ['bonjour', 'bonjour']
- affiche [bonjour, bonjour]
- affiche bonjourbonjour
- autre

Exercice N°3.

Que fait le code suivant ? :

```
texte = 'bonjour'
t1 = [texte, texte]
t1[1] = "Hello"
print(t1)
```

produit une erreur à l'exécution
affiche ['Hello', 'bonjour']
affiche ['bonjour', 'Hello']
autre

Exercice N°4.

- Que fait le code suivant ? (un seul choix :

```
texte = 'bonjour'
t1 = [texte, texte]
t2 = t1[:]
t1[1] = "Hello"
print(t2)
```

produit une erreur à l'exécution
affiche ['bonjour', 'bonjour']
affiche ['Hello', 'bonjour']
affiche ['bonjour', 'Hello']
autre

Exercice N°5.

- Que fait le code suivant ? (un seul choix :

```
def foo(liste):
    return liste + liste

x = [2,3]
x = foo(x)
print(x)
```

produit une erreur à l'exécution
affiche [2, 3, 2, 3]
affiche [[2,3],[2,3]]
affiche [2, 3]
autre

Exercice N°6.

- Que fait le code suivant ? (un seul choix :

```
def foo(liste):
    return [liste, liste]

x = [2,3]
x = foo(x)
print(x)
```

produit une erreur à l'exécution

affiche [2, 3, 2, 3]

affiche [[2,3],[2,3]]

affiche [2, 3]

autre

II. QCM NIVEAU 2

Exercice N°7. Sur les séquences

Avec la liste `s` initialisée à `s = [1,2,3,4,5,6,7]`, pour chacune des instructions dans les questions qui suivent, dites si elle (**mettre une croix dans la colonne correspondante**):

	Ajoute à *s* une composante valant 0 (a)	*Supprime la dernière composante de s* (b)	Supprime la première composante de *s* (c)	Une autre action (d)
`s.append(0)`				
`s[len(s):len(s)] = [0]`				
`s.extend([0])`				
`s.insert(len(s),0)`				
`s[0:1] = []`				
`del s[0]`				
`del s[-len(s)]`				
`del s[len(s)-1]`				
`s[len(s)-1:] = []`				

III. QCM NIVEAU 3

Exercice N°8. Strings (1)

Les chaînes de caractères en Python sont mutables.

- ○ Vrai
- ○ Faux

Exercice N°9. Strings (2)

Cocher la ou les expressions qui définissent correctement une chaîne de caractères en Python.

- ☐ `'spam"`
- ☐ `"spam"`
- ☐ `'spam"'`
- ☐ `'spam'`
- ☐ `'"'"`
- ☐ `"'"`
- ☐ `'"'"`

On veut afficher la chaîne de caractères `le numéro d'appel des urgences est le 112` et on a les variables suivantes qui sont définies :

```
text = 'des urgences'
numero = 112
```

Quelle(s) solution(s) est(sont) correcte(s) ?

- ☐ `'le numéro d'appel {} est le {}'.format(text, numero)`
- ☐ `"le numéro d'appel {text} est le {numero}"`
- ☐ `"le numéro d'appel {} est le {}".format(text, numero)`
- ☐ `f"le numéro d'appel {text} est le {numero}"`

Exercice N°10. Opérations sur les séquences

On se donne la variable suivante

```
chaine = "douarnenez"
```

Parmi les expressions suivantes, quelles sont celles qui s'évaluent à `True` ?

- [ ] `chaine.index('z') == len(chaine)`
- [ ] `chaine.index('z') == len(chaine) - 1`

Exercice N°11. Appartenance

Parmi les expressions suivantes, quelles sont celles qui s'évaluent à `True` ?

- [ ] `'daz' in chaine`
- [ ] `'nez' in chaine`

Exercice N°12. Slicing

Parmi les expressions suivantes, quelles sont celles qui s'évaluent à `True` ?

- [ ] `chaine[-3:] == 'nez'`
- [ ] `chaine[1:3] + chaine[3:5] + chaine[5:] == chaine[1:]`

Exercice N°13. Listes (1)

On a :

```
liste = [0, 1, 2, 3]
```

On veut **modifier** l'objet `liste` pour que sa valeur devienne `[0, 1, 4, 2, 3]`
Que faut-il faire ? (plusieurs réponses possibles)

- [ ] `liste[2] = 4`
- [ ] `liste[2] = [4]`
- [ ] `liste.insert(2,4)`
- [ ] `liste[2:3] = [4]`
- [ ] `liste[2:2] = [4]`

Exercice N°14. Listes (2)

À nouveau on a

```
liste = [0, 1, 2, 3]
```

On souhaite ***extraire*** et **retourner** le premier élément `0`, tout en la retirant de la liste.
Plus précisément on veut affecter à la variable `suivant` la valeur `0` de telle sorte qu'après l'exécution, `liste` ne contienne plus que `[1, 2, 3]`

Que faut-il faire ? (Plusieurs réponses possibles)

- [ ] `suivant = liste[0]`
- [ ] `suivant = liste.pop(0)`
- [ ] `del liste[0]`
- [ ] `suivant = liste[0]; del liste[0]`
- [ ] `suivant = del liste[0]`

Exercice N°15. Listes (3)

On a cette fois

```
liste = [1, 0, 3, 2]
```

On veut trier la liste en ordre décroissant et **en place**, c'est-à-dire **sans dupliquer** la liste ni ses éléments.

Faut-il faire : (plusieurs réponses possibles)

- [ ] `liste.sort(reverse=True)`
- [ ] `liste.sort()`
- [ ] `sorted(liste, reverse=True)`
- [ ] `liste.sort(); liste.reverse()`

Exercice N°16. Tuples (1)

On se donne en entrée

```
triple = (1, 2, 3,)
```

Parmi les expressions et instructions ci-dessous, lesquelles sont valides ?

- [ ] `triple[0]`
- [ ] `triple[:]`
- [ ] `triple[len(triple)]`
- [ ] `triple[0] = 0`

Exercice N°17. Tuples (2)

Quelles sont les expressions qui renvoient `True` ?

- [ ] `[ ( [ (1) ] ) ] == [ [ 1 ] ]`
- [ ] `(1,) == (1)`
- [ ] `[ (1,) ] [0][0] == 1`
- [ ] `[ (1), ] [0][0] == 1`

Exercice N°18. *Unpacking* (1)

On pose

Quelles sont parmi les affectations suivantes celles qui sont valables, et qui affectent 4 à `four` ?

- [ ] `( one, (two, three), ignored, ( ( four ) ) ) = quadruple`
- [ ] `( one, (two, three,), _, ( ( four, ), ) ) = quadruple`
- [ ] `( (one,), (two, three), _, [ [ four ] ] ) = quadruple`
- [ ] `( one, (two, three), _, [ [ four ] ] ) = quadruple`

Exercice N°19. *Unpacking* (2)

On cherche à écrire un code qui permette d'intervertir les deux derniers éléments dans une liste. On suppose que la liste en entrée a au moins deux éléments. Quelles sont parmi les variantes suivantes celles qui font bien ce qu'on veut ?

- [ ] `tmp = liste[-1]; liste[-1] = liste [-2]; liste[-2] = tmp`
- [ ] `liste.reverse(-2, -1)`
- [ ] `liste[-2], liste[-1] = liste[-1], liste[-2]`

- [x] `tmp = liste[-1]; liste[-1] = liste [-2]; liste[-2] = tmp` ✔
- [ ] `liste.reverse(-2, -1)`
- [x] `liste[-2], liste[-1] = liste[-1], liste[-2]` ✔

Exercice N°20. Dictionnaires (1)

Le type dictionnaire est un type :

- ○ immuable
- ○ mutable

Exercice N°21. Dictionnaires (2)

Est-il exact de dire que la recherche d'une clé dans un dictionnaire prend le même temps que la recherche d'un élément dans une liste (de même longueur) :

- ○ oui
- ○ non

- ○ oui
- ◉ non ✔

Exercice N°22. Dictionnaires (3)

Parmi les objets suivants, quels sont ceux qui peuvent être utilisés comme une clé dans un dictionnaire :

- ☐ `1`
- ☐ `[1, 2]`
- ☐ `(1, 2)`
- ☐ `([1, 2], [3, 4])`

Exercice N°23. Dictionnaires (4)

Pour modifier la valeur de la clé `'marc'` **dont on sait** qu'elle est **présente** dans le dictionnaire `annuaire`, on peut faire :

- ☐ `annuaire['marc'] = 50`
- ☐ `annuaire.get('marc', 50)`
- ☐ `annuaire.setdefault('marc', 50)`