EXERCICE 1 : QCM (8 points)

Question 1 : Indiquez les commandes correctes :

- [ ] msg = "bonjour'
- [ ] msg = bonjour
- [x] msg = 'bonjour'
- [x] msg, nom = "Bonjour Professeur"
- [x] msg = "bounjoir"
- [ ] msg = """Bonjour"
- [x] msg = """Bonjour"""
- [x] msg, nom = "Bonjour", "Professeur"

Question 2 :

soit la commande suivantes :

```
phrase = "    Il fait beau aujourd'hui    "
```

Indiquez les commandes qui affichent **True** :

Veuillez choisir au moins une réponse :

- [ ] print(phrase.iscapitalize())
- [ ] print(phrase.isdigit())
- [ ] print(phrase.isupper())
- [x] print(phrase.isalpha())
- [x] print(phrase.islower())

Question 3 :

soit les commandes suivantes :

```
phrase = "Il fait beau aujourd'hui"
print(phrase.find("b"))
```

Indiquez ce que va afficher la commande ***print*** ci-dessus

Veuillez choisir une réponse :

- ( ) 9
- ( ) 19
- ( ) 0
- (x) 8

Question 4 :

Quel est le mot clé permettant de déclarer une fonction :

Veuillez choisir au moins une réponse :

- [x] def
- [ ] function
- [ ] break
- [ ] declare
- [ ] return

Question 5 : Soit les commandes suivantes :

```
phrase = "Il fait beau aujourd'hui"
print(phrase[13:])
```

Indiquez ce qu'affiche la dernière commande

- ( ) il fait beau
- (x) aujourd'hui
- ( ) Il fait beau aujourd'hui

Question 6 : Soit les commandes suivantes :

```
phrase = "    Il fait beau aujourd'hui    "
print(phrase.strip())
```

Indiquez ce qu'affiche la dernière commande

- ( ) Il fait beau aujourd'hui *******
- ( ) ****** Il fait beau aujourd'hui
- (x) Il fait beau aujourd'hui
- ( ) . Il fait beau aujourd'hui

Question 7 : Soit la fonction suivante :

```
>>> def f(x,y=12):
        z = x+y
        return z
```

Ln: 14 Col: 4

qu'est ce qui sera afficher si je l'appel comme suit: **print**(f(12))

Veuillez choisir une réponse :

- (x) 24
- ( ) erreur
- ( ) 12

Question 8 : Dans le code suivant indiquez l'instruction qui ne sera jamais exécutée :

```
def stuff():
    print ("Hello")
    return
    print ("World")

stuff()
```

Réponse :

print("World")

Question 9 : Que va afficher le code suivant :

```
>>> a = 5
>>> def f(x, y=3):
        return x+y+a

>>> print(f(5,6))
```

Ln: 46 Col: 0

Réponse : 16

Question 10 : Que va afficher le code suivant :

```
>>> v = 10
>>> def f(x , y = 1):
        x = x+y
        return x+v

>>> print(f(v))
```

Ln: 34 Col: 4

Réponse : 21

Soit le programme suivant :

```
# -*- coding: utf-8 -*-
molecule = {}
molecule['H2O'] = 'eau'
molecule['O2'] = 'Bi-oxygène'
molecule['HCL'] = 'Acide chloridrique'
molecule['H2SO4'] = 'Acide sufurique'

for formule, description in molecule.items():
   print ("la formule chimique de ", description, " est ", formule)
```

Attention, la suite des questions suivantes sont liées au programme ci-dessus !

Question 11 : La variable « molecule » est :

- ○ Une liste
- ○ Un tuple
- ✓ Un dictionnaire

Question 12 : Le littérale « H2O » est :

- ○ Une valeur
- ○ Une donnée quelconque
- ✓ Une clé

Question 13 : L'instruction « molecule["H2O"]="eau" permet de :

- ○ Créer un nouvel élément du dictionnaire «molecule». Cet élément aura pour clé : «eau» et pour valeur «H2O»
- ✓ Créer un nouvel élément du dictionnaire «molecule». Cet élément aura pour clé : «H2O» et pour valeur «eau»

Question 14 : Dans la boucle « **for** » la méthode « **items**() » renvoi :

- ○ Uniquement les clés du dictionnaire
- ○ Uniquement les valeurs du dictionnaire
- ✓ Tous les éléments (items) du dictionnaire

Question 15 : Dans la boucle « **for** », lors de chaque itération, la variable « **formule** » reçoit :

- ○ La valeur du $i^{ème}$ élément du dictionnaire (i correspondant à l'itération de la boucle). Le dictionnaire est donc parcouru séquentiellement du début jusqu'à sa fin
- ✓ La clé du $i^{ème}$ élément du dictionnaire (i correspondant à l'itération de la boucle). Le dictionnaire est donc parcouru séquentiellement du début jusqu'à sa fin

Question 16 : Dans le programme ci-dessus, « **molecule.keys()** » renvoi :

- ○ Ensemble des valeurs
- ✓ Ensemble des clés

EXERCICE 2 (3 points) : Écrire un programme qui demande à l'utilisateur les coordonnées de deux points A et B dans le plan et qui calcule puis affiche la distance entre ces deux points selon la formule :

$$d = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

Les coordonnées de A sont (x_a, y_a) et les coordonnées de B sont (x_b, y_b).
Indication : Ne documentez pas votre programme

Exemple d'exécution

```
Donnez l'abscisse du point A : 10
Donnez l'odronnée du point A : 10
Donnez l'abscisse du point b : 20
Donnez l'odronnée du point b : 20
la distance entre A et B est :  14.142135623730951
```

```python
import math
xa = float(input("Donnez l'abscisse du point A : "))
ya = float(input("Donnez l'odronnée du point A : "))
xb = float(input("Donnez l'abscisse du point b : "))
yb = float(input("Donnez l'odronnée du point b : "))

distanceAB = math.sqrt((xa-xb)**2+(ya-yb)**2)
print("\nla distance entre A et B est : ", distanceAB)
```

EXERCICE 3 (3 points) : Ecrire une fonction « ***lireReel()*** » qui permet de lire à partir du clavier un nombre réel. Cette fonction doit avoir en entrée une chaine de caractères (un message à afficher) et comme résultat un nombre réel lu à partir du clavier. Bien évidement, vous devez avertir l'utilisateur d'éventuelles erreurs de saisie et garantir à la fin que seul des nombre réels seront renvoyés par cette fonction.

```python
x = lireReel("Donnez un nombre réel : ")
```

Exemple d'exécution

```
Donnez un nombre réel : Test
        ---> Vous devez saisir un nombre réel ...
Donnez un nombre réel : 12..5
        ---> Vous devez saisir un nombre réel ...
Donnez un nombre réel : 12.5
```

inputOutput.py

```python
"""
Fonction LireReel
"""
def lireReel(message):
    caractèresPermis = "0123456789."
    while True:
        r = input(message)
        erreur = False

        if r.count(".")>1:
            erreur = True

        for c in r:
            if c not in  caractèresPermis:
                erreur = True
                break

        if erreur:
            print("\n\t ---> Vous devez saisir un nombre réel ..."
        else :
            return float(r)
```

EXERCICE 4 (2 points) : En supposant que la fonction « ***lireReel***() » est dans un module appelé « ***inputOutput*** ». Je vous demande de reprendre l'exercice 2 en utilisant la fonction « ***lireReel***() ». Vous devez donc importer le module « ***inputOutput*** ».

```
"""
Exercice 4 de l'examen blanc
"""

import math

import inputOutput

xa = lireReel("Donnez l'abscisse du point A : ")
ya = lireReel("Donnez l'odronnée du point A : ")
xb = lireReel("Donnez l'abscisse du point b : ")
yb = lireReel("Donnez l'odronnée du point b : ")

distanceAB = math.sqrt((xa-xb)**2+(ya-yb)**2)
print("\nla distance entre A et B est : ", distanceAB)
```

EXERCICE 5 (4 points) : Je suppose que j'ai dans un fichier nommé « ***sommets.txt*** » les coordonnées des sommets d'un polygone. Sur chaque ligne de ce fichier j'ai l'abscisse et l'ordonnée, d'un sommet, séparées par une virgule Je vous demande d'écrire un programme qui calcul le périmètre de ce polygone. Attention, vous devez utiliser une fonction nommée « ***distance()*** » qui calcul la distance entre 2 sommets ***A*** et ***B*** en se basant sur les coordonnées du premier sommets et celles du second *(x0, ya)* et *(xb, yb)*.

exerice5.py

```
"""
Calcul du périmètre d'un polygone
"""

import math
def distance(xa, ya, xb, yb):
    return math.sqrt((xa-xb)**2+(ya-yb)**2)

f = open("sommets.txt", "r")
texte = f.read()

# je récupère les lignes de mon texte
lignes = texte.split("\n")

# Je supprimes les lignes vides
while '' in lignes:  lignes.remove('')
while '' in lignes:  lignes.remove('')

# je définit 2 dictionnaires abscisses et ordonnées:
# ils auront comme clé les numéros des sommets
# et comme valeur respectivement l'abscisse et l'orodnnée du sommet
abscisseDe = dict()
ordonnéeDe = dict()

# Je construit mes 2 dictionnaires à parir des lignes
# la ième ligne correspond au sommet i
nbrSommets = len(lignes)
for i in range(nbrSommets):
    abscisse = lignes[i].split(",")[0]
    ordonnée = lignes[i].split(",")[1]
    abscisseDe[i] = float(abscisse.strip())
    ordonnéeDe[i] = float(ordonnée.strip())

# je calcul le périmètre du polygonne en additionne les distances
# entre 2 sommets successifs en parcourant le liste des sommets

perimetre = 0
i = 0
while (i < nbrSommets - 1) :
    xa = abscisseDe[i]
    ya = ordonnéeDe[i]
    xb = abscisseDe[i+1]
    yb = ordonnéeDe[i+1]
    perimetre = perimetre + distance(xa, ya, xb, yb)
    i=i+1

print("Le périmètre de votre polygone est : ", perimetre)
```