BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°29

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

On s'intéresse à la suite d'entiers définie par

$U_1 = 1$, $U_2 = 1$ et, pour tout entier naturel n, par $U_{n+2} = U_{n+1} + U_n$.

Elle s'appelle la suite de Fibonnaci.

Écrire la fonction `fibonacci` qui prend un entier `n > 0` et qui renvoie l'élément d'indice `n` de cette suite.

On utilisera une programmation dynamique (pas de récursivité).

Exemples :

```
>>> fibonacci(1)
1
>>> fibonacci(2)
1
>>> fibonacci(25)
75025
>>> fibonacci(45)
1134903170
```

EXERCICE 2 (4 points)

Les variables `liste_eleves` et `liste_notes` ayant été préalablement définies et étant de même longueur, la fonction `meilleures_notes` renvoie la note maximale qui a été attribuée, le nombre d'élèves ayant obtenu cette note et la liste des noms de ces élèves.

Compléter le code Python de la fonction `meilleures_notes` ci-dessous.

```
liste_eleves = ['a','b','c','d','e','f','g','h','i','j']
liste_notes = [1, 40, 80, 60, 58, 80, 75, 80, 60, 24]

def meilleures_notes():
    note_maxi = 0
    nb_eleves_note_maxi = ...
    liste_maxi =  ...

    for compteur in range(...):
        if liste_notes[compteur] == ...:
            nb_eleves_note_maxi = nb_eleves_note_maxi + 1
```

```
            liste_maxi.append(liste_eleves[...])
        if liste_notes[compteur] > note_maxi:
            note_maxi = liste_notes[compteur]
            nb_eleves_note_maxi = ...
            liste_maxi = [...]

    return (note_maxi,nb_eleves_note_maxi,liste_maxi)
```

Une fois complété, le code ci-dessus donne

```
>>> meilleures_notes()
(80, 3, ['c', 'f', 'h'])
```

```python
def fibonacci(n: int) -> int:
    fib = [None, 1, 1]
    for i in range(3, n+1):
        f0 = fib[i-2]
        f1 = fib[i-1]
        f2 = f1 + f0
        fib.append(f2)
    return fib[n]

assert fibonacci(1) == 1
assert fibonacci(2) == 1
assert fibonacci(25) == 75025
assert fibonacci(45) == 1134903170
```

```python
liste_eleves = ['a','b','c','d','e','f','g','h','i','j']
liste_notes = [1, 40, 80, 60, 58, 80, 75, 80, 60, 24]

def meilleures_notes():
    note_maxi = 0
    nb_eleves_note_maxi = 0
    liste_maxi =  []

    for compteur in range(len(liste_eleves)):
        if liste_notes[compteur] == note_maxi:
            nb_eleves_note_maxi = nb_eleves_note_maxi + 1
            liste_maxi.append(liste_eleves[compteur])
        if liste_notes[compteur] > note_maxi:
            note_maxi = liste_notes[compteur]
            nb_eleves_note_maxi = 1
            liste_maxi = [liste_eleves[compteur]]

    return (note_maxi,nb_eleves_note_maxi,liste_maxi)



#  version pour codeboard.IOErrorimport Root.src.main as mn
# import unittest


# class validation(unittest.TestCase):
#     def test_1_statique(self):
#         print(mn.meilleures_notes())
#         self.assertTupleEqual(mn.meilleures_notes(), (80, 3, ['c', 'f', 'h']))

#     def test_2_alea(self):
#         from random import randint
#         n = randint(5, 20)
#         mn.liste_notes = [randint(0, 2*n//3) for _ in range(n)]
#         mn.liste_eleves = []
#         while len(mn.liste_eleves) != n:
#             nom = chr(randint(97, 122))
#             if nom not in mn.liste_eleves:
#                 mn.liste_eleves.append(nom)

#         def meilleures_notes_sol():
#             note_maxi = 0
#             nb_eleves_note_maxi = 0
#             liste_maxi =  []

#             for compteur in range(len(mn.liste_eleves)):
#                 if mn.liste_notes[compteur] == note_maxi:
#                     nb_eleves_note_maxi = nb_eleves_note_maxi + 1
#                     liste_maxi.append(mn.liste_eleves[compteur])
#                 if mn.liste_notes[compteur] > note_maxi:
#                     note_maxi = mn.liste_notes[compteur]
#                     nb_eleves_note_maxi = 1
#                     liste_maxi = [mn.liste_eleves[compteur]]

#             return (note_maxi,nb_eleves_note_maxi,liste_maxi)

#         self.assertTupleEqual(mn.meilleures_notes(), meilleures_notes_sol())


# if __name__ == '__main__':
#     unittest.main()
```