BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°05

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)

Écrire une fonction `RechercheMinMax` qui prend en paramètre un tableau de nombres non triés `tab`, et qui renvoie la plus petite et la plus grande valeur du tableau sous la forme d'un dictionnaire à deux clés 'min' et 'max'. Les tableaux seront représentés sous forme de liste Python.

Exemples :

```
>>> tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]
>>> resultat = rechercheMinMax(tableau)
>>> resultat
{'min': -2, 'max': 9}
```

```
>>> tableau = []
>>> resultat = rechercheMinMax(tableau)
>>> resultat
{'min': None, 'max': None}
```

EXERCICE 2 (4 points)

On dispose d'un programme permettant de créer un objet de type `PaquetDeCarte`, selon les éléments indiqués dans le code ci-dessous.

Compléter ce code aux endroits indiqués par `#A compléter`, puis ajouter des assertions dans l'initialiseur de `Carte`, ainsi que dans la méthode `getCarteAt()`.

```
class Carte:
    """Initialise Couleur (entre 1 à 4), et Valeur (entre 1 à
13)"""
    def __init__(self, c, v):
        self.Couleur = c
        self.Valeur = v

    """Renvoie le nom de la Carte As, 2, ... 10,
       Valet, Dame, Roi"""
    def getNom(self):
        if ( self.Valeur > 1 and self.Valeur < 11):
            return str( self.Valeur)
        elif self.Valeur == 11:
            return "Valet"
        elif self.Valeur == 12:
            return "Dame"
        elif self.Valeur == 13:
            return "Roi"
        else:
            return "As"

    """Renvoie la couleur de la Carte (parmi pique, coeur,
carreau, trefle"""
    def getCouleur(self):
        return ['pique', 'coeur', 'carreau', 'trefle'
][self.Couleur - 1]

class PaquetDeCarte:
    def __init__(self):
        self.contenu = []

    """Remplit le paquet de cartes"""
    def remplir(self):
     #A compléter


    """Renvoie la Carte qui se trouve à la position donnée"""
    def getCarteAt(self, pos):
        #A compléter
```

Exemple :

```
>>> unPaquet = PaquetDeCarte()
>>> unPaquet.remplir()
>>> uneCarte = unPaquet.getCarteAt(20)
>>> print(uneCarte.getNom() + " de " + uneCarte.getCouleur())
```

```
6 de coeur
```

```python
from doctest import testmod


def rechercheMinMax(tab):
    """ Recherche du min et du max d'un tableau.

    Args:
        tab (list): tableau non trié de nombres entiers

    Returns:
        dict: dictionnaire ayant pour clés :
        'min' pour la valeur minimale
        'max' pour la valeur maximale

    Tests et Exemples:
    >>> tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]
    >>> resultat = rechercheMinMax(tableau)
    >>> resultat
    {'min': -2, 'max': 9}

    >>> tableau = []
    >>> resultat = rechercheMinMax(tableau)
    >>> resultat
    {'min': None, 'max': None}
    """
    n_tab = len(tab)
    if n_tab == 0:
        return {'min': None, 'max': None}

    # initialisation des valeurs maximales et minimales
    # avec le premier élément du tableau
    v_min = tab[0]
    v_max = tab[0]
    for i in range(1, n_tab):
        # parcours de tous les éléments non visités du tableau
        if tab[i] > v_max:
            # mise àjour de la valeur max
            v_max = tab[i]
        elif tab[i] < v_min:
            # mise à jour de la valeur min
            v_min = tab[i]

    # résultat sous la forme de dictionnaire
    return {'min': v_min, 'max': v_max}


assert rechercheMinMax([0, 1, 4, 2, -2, 9, 3, 1, 7, 1]) == {'min': -2, 'max': 9}
assert rechercheMinMax([]) == {'min': None, 'max': None}

testmod()
```

```python
class Carte:
    """Initialise Couleur (entre 1 à 4), et Valeur (entre 1 à 13)"""
    def __init__(self, c, v):
        # ajout des clauses de gardes sur
        # les couleurs et les valeurs
        assert 1 <= c <= 4, 'erreur de couleur'
        assert 1 <= v <= 13, 'erreur de valeur'
        self.Couleur = c
        self.Valeur = v

    """Renvoie le nom de la Carte As, 2, ... 10,
       Valet, Dame, Roi"""
    def getNom(self):
        if ( self.Valeur > 1 and self.Valeur < 11):
            return str( self.Valeur)
        elif self.Valeur == 11:
            return "Valet"
        elif self.Valeur == 12:
            return "Dame"
        elif self.Valeur == 13:
            return "Roi"
        else:
            return "As"

    """Renvoie la couleur de la Carte (parmi pique, coeur, carreau, trefle"""
    def getCouleur(self):
        return ['pique', 'coeur', 'carreau', 'trefle' ][self.Couleur - 1]

class PaquetDeCarte:
    def __init__(self):
        self.contenu = []

    """Remplit le paquet de cartes"""
    def remplir(self):
        for c in range(1, 5):
            for v in range(1, 14):
                self.contenu.append(Carte(c, v))

    """Renvoie la Carte qui se trouve à la position donnée"""
    def getCarteAt(self, pos):
        # clause de garde pour ne pas demander une
        # position plus grande que le tableau de cartes
        n = len(self.contenu)
        assert 0 <= pos < n, 'index out of range'

        return self.contenu[pos]


unPaquet = PaquetDeCarte()
unPaquet.remplir()
uneCarte = unPaquet.getCarteAt(20)
print(uneCarte.getNom(), "de", uneCarte.getCouleur())
assert str(uneCarte.getNom()) + " de " + str(uneCarte.getCouleur()) == "8 de coeur"
```