Atelier_3_Corr

N'oubliez pas d'utiliser des commentaires pour expliquer le fonctionnement de chaque partie de votre code.

Les execeptions en Python

Activité 1

Écrivez un programme qui demande à l'utilisateur de saisir deux nombres. Le programme doit afficher le résultat de la division du premier nombre par le deuxième nombre. Cependant, le programme doit également gérer les exceptions possibles :

Si l'utilisateur entre des valeurs qui ne sont pas des nombres, le programme doit afficher un message d'erreur et redemander les valeurs.

Si l'utilisateur entre 0 comme deuxième nombre, le programme doit afficher un message d'erreur indiquant que la division par zéro n'est pas autorisée et redemander le deuxième nombre.

Si une autre erreur se produit lors du calcul de la division, le programme doit afficher un message d'erreur générique

Solution

```
while True:
    try:
        numerateur = float(input("Entrez le premier nombre : "))
        denominateur = float(input("Entrez le deuxième nombre : "))

        if denominateur == 0:
            raise ZeroDivisionError("La division par zéro n'est pas autorisée.")

        resultat = numerateur / denominateur
        print("Le résultat de la division est :", resultat)
        break

    except ValueError:
        print("Veuillez saisir des nombres valides.")
    except ZeroDivisionError as e:
        print(e)
    except Exception as e:
        print("Une erreur est survenue lors du calcul de la division :", e)
```

La gestion des fichiers

Activité 1

Créez un fichier texte nommé "input.txt" contenant une liste de nombres séparés par des virgules.

Écrivez un programme Python qui lit les nombres à partir du fichier "input.txt", les additionne, soustrait 5 à chaque nombre et calcule le carré de chaque résultat.

Sauvegardez les résultats des opérations dans un nouveau fichier texte nommé "output.txt", chaque résultat étant sur une nouvelle ligne.

Solution

```
with open("input.txt", 'w') as fichier:
    fichier.write("10, 20, 30, 40, 50\n")
# Lire les nombres à partir du fichier
with open("input.txt", 'r') as fichier:
    nombres = fichier.readline().split(',')
    nombres = [int(nombre.strip()) for nombre in nombres]
# Effectuer les opérations et sauvegarder les résultats dans un nouveau fichier
if nombres:
    resultats = [(nombre + 5) ** 2 for nombre in nombres]
    try:
        with open("output.txt", 'w') as fichier_sortie:
            for resultat in resultats:
                fichier_sortie.write(str(resultat) + '\n')
    except Exception as e:
        print("Une erreur est survenue lors de l'écriture dans le fichier de sortie:", e)
else:
    print("Aucun nombre à traiter.")
```

Activité 2

Écrivez un programme qui demande à l'utilisateur de saisir le nom d'un fichier texte. Le programme doit ouvrir le fichier, lire son contenu et afficher le nombre de lignes qu'il contient. Cependant, le programme doit également gérer les exceptions possibles :

Si le fichier spécifié n'existe pas, le programme doit afficher un message d'erreur indiquant que le fichier n'a pas été trouvé.

Si une autre erreur se produit lors de la lecture du fichier, le programme doit afficher un message d'erreur générique.

Solution

```
while True:
    nom_fichier = input("Entrez le nom du fichier texte : ")

    try:
        with open(nom_fichier, 'r') as fichier:
```

```
            lignes = fichier.readlines()
            nb_lignes = len(lignes)
            print("Le fichier contient", nb_lignes, "ligne(s).")
        break

    except FileNotFoundError:
        print("Le fichier spécifié n'a pas été trouvé. Veuillez vérifier le nom du fichier.")
    except Exception as e:
        print("Une erreur est survenue lors de la lecture du fichier :", e)
```

Les procédures et fonctions

Activité 1

Écrivez une fonction appelée "somme_pairs" qui prend en entrée une liste d'entiers et renvoie la somme des nombres pairs présents dans cette liste.

Solution

```
def somme_pairs(liste_entiers):
    somme = 0
    for nombre in liste_entiers:
        if nombre % 2 == 0:  # Vérifie si le nombre est pair en utilisant l'opérateur modulo (%)
            somme += nombre
    return somme

# Exemple d'utilisation de la fonction
liste = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
resultat = somme_pairs(liste)
print("La somme des nombres pairs dans la liste est :", resultat)
```

Activité 2

Écrivez deux fonctions en Python pour calculer l'aire et le périmètre d'un rectangle en fonction de sa longueur et de sa largeur. Voici les étapes à suivre :

Créez une fonction appelée calculer_aire_rectangle qui prend en entrée deux paramètres longueur et largeur et renvoie l'aire du rectangle calculée en multipliant la longueur par la largeur.

Créez une fonction appelée calculer_perimetre_rectangle qui prend en entrée deux paramètres longueur et largeur et renvoie le périmètre du rectangle calculé en utilisant la formule : périmètre = 2 * (longueur + largeur).

Testez vos fonctions en appelant chacune d'elles avec différentes valeurs de longueur et de largeur, puis affichez les résultats.

Solution

```
# Fonction pour calculer l'aire du rectangle
def calculer_aire_rectangle(longueur, largeur):
```

```
    aire = longueur * largeur
    return aire
# Fonction pour calculer le périmètre du rectangle
def calculer_perimetre_rectangle(longueur, largeur):
    perimetre = 2 * (longueur + largeur)
    return perimetre
# Test des fonctions avec différentes valeurs de longueur et de largeur
longueur = 5
largeur = 3
aire = calculer_aire_rectangle(longueur, largeur)
perimetre = calculer_perimetre_rectangle(longueur, largeur)
print("Aire du rectangle :", aire)
print("Périmètre du rectangle :", perimetre)
```

Activité 3

Ecrivez un script qui crée un mini-système de base de données fonctionnant à l'aide d'un dictionnaire, dans lequel vous mémoriserez les noms d'une série des élèves, leur âge et leur taille.

Dans le dictionnaire, le nom de l'élève servira de clé d'accès, et les valeurs seront constituées de tuples (âge, taille), dans lesquels l'âge sera exprimé en années (donnée de type entier), et la taille en mètres (donnée de type réel).

Votre script devra comporter deux fonctions :

a. La fonction remplissage() qui permet de remplir le dictionnaire, les données du dictionnaires seront saisies par l'utilisateur.
b. La fonction consultation(). Dans la fonction de consultation permet de consulter les information, le couple (âge, taille) relatives au nom de l'élève fournit par l'utilisateur.

Le résultat de la requête devra être une ligne de texte bien formatée.

Solution

```
def remplissage():
    R='o'
    while R=='o':
        nom = input("\nEntrer le nom de l'élève ")
        age = int(input("Entrer l'age de %s " %nom))
        taille = float(input("Entrer la taille de %s " %nom))
        eleves[nom]=(age,taille)
        R=input("\nVoulez vous ajouter un autre élève. O/N? : ").lower()
def consultation():
    nom = input("\nEntrer le nom de l'élève à consulter ")
    if nom in eleves:
        age = eleves[nom][0]
        taille = eleves[nom][1]
        print("\nNom : %s - âge : %d ans - taille : %.2f m " %(nom,age,taille))
    else:
```

```
        print("l'élève %s n'existe pas!"%nom)
eleves ={}
remplissage()
consultation()
```

Activité 4

Écrivez une fonction nommée "mediane" qui prend en entrée une liste de nombres et renvoie la médiane des éléments de la liste. (Indice : pour une liste de longueur impaire, la médiane est l'élément du milieu une fois la liste triée ; pour une liste de longueur paire, la médiane est la moyenne des deux éléments du milieu.)

Testez cette fonction avec différentes listes de nombres pour vous assurer qu'elles fonctionnent correctement

Solution

```
def mediane(liste):
    liste_triee = sorted(liste)
    longueur = len(liste_triee)
    if longueur % 2 == 0:
        mediane_inf = liste_triee[longueur // 2 - 1]
        mediane_sup = liste_triee[longueur // 2]
        return (mediane_inf + mediane_sup) / 2
    else:
        return liste_triee[longueur // 2]
liste = [5, 2, 8, 1, 3, 7, 6, 4]
print(mediane(liste))
```

Activité 5

Écrivez un programme Python qui effectue les opérations suivantes sur une liste de mots:

c. Définir une fonction nommée "inversion_mots" qui prend une liste de mots en entrée et renvoie une nouvelle liste contenant les mots inversés. Par exemple, "hello" deviendra "olleh".

d. Définir une fonction nommée "mot_palindrome" qui prend une liste de mots en entrée et renvoie une nouvelle liste contenant uniquement les mots qui sont des palindromes (c'est-à-dire des mots qui restent les mêmes lorsqu'ils sont lus à l'envers).

Testez toutes les fonctions avec différentes listes de mots pour vous assurer qu'elles fonctionnent correctement.

Solution

```
def inversion_mots(liste_mots):
    nl=[]
    for mot in liste_mots:
```

```
        nl.append(mot[::-1])
    return (nl)
    #return [mot[::-1] for mot in liste_mots]

def mot_palindrome(liste_mots):
    nl = []
    for mot in liste_mots:
        if mot == mot[::-1]:
            nl.append(mot)
    return (nl)
    #return [mot for mot in liste_mots if mot == mot[::-1]]

# Test des fonctions
liste_mots = ["radar", "python", "Bonbon", "ana", "deed"]
print(inversion_mots(liste_mots))
print(mot_palindrome(liste_mots))
```

Activité 6(lambda)

Écrivez une fonction nommée "calculatrice" qui prend deux nombres et une chaîne de caractères représentant une opération mathématique ('+', '-', '*', '/'). Utilisez une expression lambda avec une série de conditions if-elif-else pour effectuer le calcul approprié en fonction de l'opération fournie. Renvoyez le résultat du calcul.

Solution

```
def calculatrice(nombre1, nombre2, operation):
    operations = {
        '+': lambda x, y: x + y,
        '-': lambda x, y: x - y,
        '*': lambda x, y: x * y,
        '/': lambda x, y: x / y
    }

    # Vérifier si l'opération est valide
    if operation not in operations:
        raise ValueError("Opération non valide")

    # Récupérer la fonction correspondant à l'opération
    fonction_operation = operations[operation]

    # Appeler la fonction avec les deux nombres en entrée
    resultat = fonction_operation(nombre1, nombre2)

    # Renvoyer le résultat
    return resultat

# Test de la fonction calculatrice
try:
    print(calculatrice(5, 3, '+'))   # 8
```

```
    print(calculatrice(5, 3, '-'))   # 2
    print(calculatrice(5, 3, '*'))   # 15
    print(calculatrice(6, 2, '/'))   # 3.0
    print(calculatrice(5, 3, '^'))   # Erreur : Opération non valide
except ValueError as e:
    print(str(e))  # Affiche l'erreur
```

Bon travail