**Gérer la logique de programmation dans Python**

**Description**

*Vous allez créer une mini-calculatrice pour finaliser cette deuxième partie de votre cours. Vous pourrez mettre en place des structures conditionnelles, des boucles et des fonctions. Ne vous inquiétez pas, vous serez guidé tout au long des questions.*

Haut du formulaire

**Question 1**

Pour commencer, vous devrez créer une fonction qui affichera un message de bienvenue à l'utilisateur.

Quelle fonction permet d'afficher un message de bienvenue à l'utilisateur ?

def afficher\_message\_accueil:

print("Bienvenue sur la mini-calculatrice !")

function afficher\_message\_accueil():

print("Bienvenue sur la mini-calculatrice !")

def afficher\_message\_accueil():

print("Bienvenue sur la mini-calculatrice !")

function afficher\_message\_accueil()

print("Bienvenue sur la mini-calculatrice !")

*Il n'y a qu'une seule réponse qui respecte la syntaxe correcte pour une fonction. La fonction commence par le mot-clé def  , suivi du nom de la fonction et des parenthèses vides, puis se termine par un deux-points. Ensuite, le code de la fonction est indenté sous cette ligne. La fonction afficher\_message\_accueil()  affiche le message de bienvenue à l'utilisateur grâce à la fonction print()  .*

*Le mot-clé function  n’existe pas dans Python.*

**Question 2**

Quelle structure conditionnelle permet d’éviter la division par zéro, et d’afficher un message dans ce cas ?

if denominator == 0:

return numerator / denominator

else:

print("Erreur : division par zéro")

if denominator != 0:

print("Erreur : division par zéro")

else:

return numerator / denominator

if denominator != 0:

return numerator / denominator

else:

print("Erreur : division par zéro")

*La structure conditionnelle doit vérifier si le dénominateur est différent de zéro. Si c'est le cas, alors la fonction retourne la division de  numerator  par  denominator  . Sinon, un message d'erreur est affiché. Cela permet d'éviter la division par zéro qui provoquerait une erreur.*

**Question 3**

Quelle autre solution permet d'éviter cette erreur en utilisant la gestion d'exception ?

match

si/sinon

return

 try/except

*En utilisant la gestion d'exception, on peut utiliser un bloc try/except  pour gérer l'erreur de division par zéro.*

try:

result = numerator / denominator

return result

except ZeroDivisionError:

print("Erreur : division par zéro")

*Dans ce cas, si l'utilisateur essaie de diviser par zéro, une exception ZeroDivisionError  sera levée, et le programme exécutera le bloc except  pour gérer cette erreur et afficher un message approprié.*

**Question 4**

Il faut maintenant créer une fonction qui encapsule tout ce bloc de code.

Quelles fonctions correspondent à la division entre deux nombres ?

Attention, plusieurs réponses sont possibles.

def division(numerator, denominateur):

try:

result = numerator / denominateur

return result

except ZeroDivisionError:

print("Erreur : division par zéro")

def division(numerator, denominator):

if denominator != 0:

return numerator / denominator

else:

print("Erreur : division par zéro")

def division():

try:

result = numerator / denominateur

return result

except ZeroDivisionError:

print("Erreur : division par zéro")

def division(numerator, denominateur):

try:

result = numerator / denominateur

return result

except ZeroDivisionError:

print("Erreur : division par zéro")

*Pour définir une fonction en Python, il faut utiliser le mot-clé def  suivi du nom de la fonction et de ses paramètres entre parenthèses. Dans le cas de la fonction division, les deux paramètres sont numerator  et denominator  . Il est important de bien indenter le bloc de code correspondant à la fonction. En ce qui concerne la fonction division  , il existe deux façons de gérer le cas où le dénominateur est égal à zéro : soit avec une structure conditionnelle if/else  , soit avec la gestion d'exception try/except  .*

**Question 5**

Vous devez maintenant demander à l’utilisateur d’entrer les nombres qu’on souhaite additionner, soustraire, multiplier ou diviser. Il faut donc créer une fonction qui permette de récupérer ces deux nombres.

Quelle fonction permet de récupérer les nombres saisis par l'utilisateur ?

def saisir\_deux\_nombres():

num1 = float(input("Entrez le premier nombre : "))

num2 = float(input("Entrez le deuxième nombre : "))

return num1, return num2

def saisir\_deux\_nombres():

num1 = float(input("Entrez le premier nombre : "))

num2 = float(input("Entrez le deuxième nombre : "))

return num1, num2

def saisir\_deux\_nombres():

num1 = float(input("Entrez le premier nombre : "))

num2 = float(input("Entrez le deuxième nombre : "))

return num1

def saisir\_deux\_nombres():

num1 = float(input("Entrez le premier nombre : "))

num2 = float(input("Entrez le deuxième nombre : "))

*La fonction ne nécessite aucun paramètre, les parenthèses doivent donc être laissées vides. Pour récupérer les nombres, la fonction input()  est utilisée deux fois, en veillant à les convertir en nombres à virgule flottante. Les nombres sont ensuite renvoyés séparés par une virgule à l'aide du mot-clé return  . Il est important de noter que les deux nombres doivent être retournés, et qu'un seul return  est nécessaire.*

**Question 6**

La fonction suivante permet de demander à l’utilisateur quelle opération il souhaite effectuer :

def afficher\_menu():

print("=== MENU ===")

print("1. Addition")

print("2. Soustraction")

print("3. Multiplication")

print("4. Division")

choix = input("Entrez votre choix (1-4) : ")

return choix

Cependant, cette fonction contient une petite erreur logique : que se passe-t-il si l'utilisateur entre un caractère autre que 1, 2, 3 ou 4 ? Il est donc nécessaire d'ajouter un mécanisme qui permette de demander à l'utilisateur de répéter l'opération jusqu'à ce que le caractère entré corresponde aux options disponibles dans le programme. Vous allez donc ajouter une boucle à l’intérieur de la fonction.

Quelle boucle permet de s’assurer de la saisie de l’utilisateur ?

while choix in ["1", "2", "3", "4"]:

choix = input("Choix invalide. Entrez votre choix (1-4) : ")

for i in range("1", "2", "3", "4"):

choix = input("Choix invalide. Entrez votre choix (1-4) : ")

for choix is ok:

choix = input("Choix invalide. Entrez votre choix (1-4) : ")

while choix not in ["1", "2", "3", "4"]:

choix = input("Choix invalide. Entrez votre choix (1-4) : ")

*La boucle doit continuer à demander à l'utilisateur de saisir un choix tant que ce choix n'est pas inclus dans la liste  [“1” , "2", "3", "4"]   . L'utilisateur est obligé de choisir un des quatre choix proposés. Ainsi, la boucle while  est la seule boucle adaptée, car le nombre d’itérations n’est pas connu à l’avance, et continue tant que la condition n’est pas valide. Ici, tant que l’utilisateur n’entre pas le bon caractère.*

**Question 7**

Maintenant qu’on a récupéré le choix de l’utilisateur, vous devez mettre en place une structure qui permette de lancer le calcul demandé par l’utilisateur, et de récupérer le résultat.

Quelle structure conditionnelle permet de lancer la bonne fonction adéquate en fonction du choix de l’utilisateur ? (Le choix de l’utilisateur est contenu dans la variable choix  .)

match choix:

case '1':

result = addition(num1, num2)

case '2':

result = soustraction(num1, num2)

case '3':

result = multiplication(num1, num2)

case '4':

result = division(num1, num2)

case \_:

print("Choix invalide.")

match choix:

'1':

result = addition(num1, num2)

‘2’:

result = soustraction(num1, num2)

'3':

result = multiplication(num1, num2)

'4':

result = division(num1, num2)

\_:

print("Choix invalide.")

match choix:

case '1'

result = addition(num1, num2)

case '2'

result = soustraction(num1, num2)

case '3'

result = multiplication(num1, num2)

case '4'

result = division(num1, num2)

case \_

print("Choix invalide.")

*Dans ce cas, il faut utiliser le match case, qui est beaucoup plus simple qu’une structure conditionnelle if/else/elif  . Dans le match case, l’expression à évaluer est placée après le mot-clé match  , et chaque cas possible est défini avec le mot-clé case  . Le bloc de code associé à chaque cas est exécuté si la valeur de l'expression correspond au cas correspondant. Si aucun cas ne correspond, un cas par défaut (généralement noté \_   ) peut être utilisé pour exécuter un bloc de code par défaut.*

**Question 8**

On peut maintenant afficher le résultat avec un simple print,  print("Le résultat est : ", result)   .

J’y pense ! Il reste une dernière fonctionnalité intéressante que vous pouvez ajouter. Il s’agit de calculer la moyenne de  n   , nombre que l’utilisateur aura préalablement saisi dans le programme. Considérez que l’ensemble des nombres est stocké dans la liste nombres  . Vous allez devoir utiliser une boucle pour calculer la moyenne de tous les nombres stockés dans cette liste.

Comment calculer la moyenne ?

Pour calculer la moyenne de plusieurs nombres, vous devez les ajouter tous ensemble, puis diviser la somme par le nombre total de nombres.

Quelle fonction permet de calculer la moyenne des nombres stockés dans la liste nombres  ?

def calculer\_moyenne(nombres):

total = 0

while nombre in nombres:

total += nombre

moyenne = total / len(nombres)

return moyenne

def calculer\_moyenne(nombres):

total = 0

for nombre in nombres:

total += nombre

moyenne = total / len(nombres)

return moyenne

def calculer\_moyenne(nombres):

total = 0

for nombre in nombres:

total += nombre

moyenne = total / len(nombres)

return moyenne

def calculer\_moyenne(nombres):

total = 0

for nombres in nombre:

total += nombre

moyenne = total / len(nombres)

return moyenne

*Comme nous avons une liste et que nous connaissons le nombre d'éléments contenus dans la liste, il faut donc utiliser la boucle for  . Pour itérer sur la liste nombres  , il faut utiliser la syntaxe for nombre in nombres :   et ajouter à la variable total  initialisée à  0   chaque nombre de cette liste. Une fois la boucle terminée, il faut diviser le nombre contenu dans la variable total  par le nombre d'éléments dans la liste pour obtenir la moyenne.*

**Correction du code complet :**

def print\_welcome\_message():

print("Bienvenue sur la mini-calculatrice !")

def input\_two\_number():

num1 = float(input("Entrez le premier nombre : "))

num2 = float(input("Entrez le deuxième nombre : "))

return num1, num2

def print\_menu\_and\_get\_choice():

print("=== MENU ===")

print("1. Addition")

print("2. Soustraction")

print("3. Multiplication")

print("4. Division")

user\_choice = input("Entrez votre choix (1-4) : ")

while user\_choice not in ["1", "2", "3", "4"]:

user\_choice = input("Choix invalide. Entrez votre choix (1-4) : ")

return user\_choice

def sum(a, b):

return a + b

def substraction(a, b):

return a - b

def multiplication(a, b):

return a \* b

def division(a, b):

if b != 0:

return a / b

else:

print("Erreur : division par zéro")

def run\_calculation(user\_choice):

num1, num2 = input\_two\_number()

match user\_choice:

case '1':

result = sum(num1, num2)

case '2':

result = substraction(num1, num2)

case '3':

result = multiplication(num1, num2)

case '4':

result = division(num1, num2)

case \_:

print("Choix invalide.")

return result

if \_\_name\_\_ == '\_\_main\_\_':

print\_welcome\_message()

user\_choice = print\_menu\_and\_get\_choice()

result = run\_calculation(user\_choice)

print(result)

Bas du formulaire