# Exercice Python - Première année de Licence

# 1. Demandez à l'utilisateur de saisir un nombre entier et stockez-le dans une variable

nombre = int(input("Veuillez saisir un nombre entier : "))

# 2. Affichez le carré de ce nombre

carre = nombre \*\* 2

print("Le carré de", nombre, "est", carre)

# 3. Calculez la somme des chiffres de ce nombre en utilisant une boucle while

somme = 0

while nombre > 0:

somme += nombre % 10

nombre //= 10

print("La somme des chiffres du nombre est", somme)

# 4. Définissez une fonction qui prend un argument de type liste et retourne la somme de tous les éléments de la liste

def somme\_liste(liste):

somme = 0

for element in liste:

somme += element

return somme

# 5. Utilisez la fonction pour calculer la somme des éléments de la liste suivante : [2, 4, 6, 8, 10]

ma\_liste = [2, 4, 6, 8, 10]

somme = somme\_liste(ma\_liste)

print("La somme des éléments de la liste est", somme)

# 6. Définissez une classe Personne avec les attributs nom, prénom et âge, ainsi qu'une méthode afficher() qui affiche ces informations

class Personne:

def \_\_init\_\_(self, nom, prenom, age):

self.nom = nom

self.prenom = prenom

self.age = age

def afficher(self):

print("Nom :", self.nom)

print("Prénom :", self.prenom)

print("Âge :", self.age)

# 7. Créez une instance de la classe Personne et appelez sa méthode afficher()

personne1 = Personne("Durand", "Jean", 25)

personne1.afficher()

# 8. Définissez une fonction lambda qui prend un nombre en argument et retourne sa racine carrée

import math

racine\_carree = lambda x: math.sqrt(x)

# 9. Utilisez la fonction lambda pour calculer la racine carrée de 25

racine = racine\_carree(25)

print("La racine carrée de 25 est", racine)

# Exercice Python - Première année de Licence (suite)

# 1. Définissez une fonction qui prend deux nombres en argument et retourne leur produit

def produit(a, b):

return a \* b

# 2. Utilisez la fonction pour calculer le produit de 3 et 5

p = produit(3, 5)

print("Le produit de 3 et 5 est", p)

# 3. Définissez une classe Voiture avec les attributs marque, modèle et année, ainsi qu'une méthode afficher() qui affiche ces informations

class Voiture:

def \_\_init\_\_(self, marque, modele, annee):

self.marque = marque

self.modele = modele

self.annee = annee

def afficher(self):

print("Marque :", self.marque)

print("Modèle :", self.modele)

print("Année :", self.annee)

# 4. Créez une instance de la classe Voiture et appelez sa méthode afficher()

voiture1 = Voiture("Renault", "Clio", 2020)

voiture1.afficher()

# 5. Définissez une fonction qui prend un nombre en argument et retourne True s'il est pair, False sinon

def est\_pair(nombre):

if nombre % 2 == 0:

return True

else:

return False

# 6. Utilisez la fonction pour déterminer si 7 est pair ou impair

if est\_pair(7):

print("7 est pair")

else:

print("7 est impair")

# 7. Définissez une fonction qui prend une chaîne de caractères en argument et retourne True si elle est palindrome, False sinon

def est\_palindrome(chaine):

chaine\_inverse = chaine[::-1]

if chaine == chaine\_inverse:

return True

else:

return False

# 8. Utilisez la fonction pour déterminer si la chaîne "radar" est palindrome

if est\_palindrome("radar"):

print("radar est un palindrome")

else:

print("radar n'est pas un palindrome")

**Fonctions couramment utilisées :**

print() : affiche une valeur à l'écran

input() : récupère une entrée de l'utilisateur

len() : renvoie la longueur d'une chaîne de caractères, d'une liste ou d'un tuple

range() : renvoie une séquence de nombres

type() : renvoie le type de données d'une variable

int() : convertit une valeur en entier

float() : convertit une valeur en flottant

str() : convertit une valeur en chaîne de caractères

list() : crée une liste

dict() : crée un dictionnaire

abs() : renvoie la valeur absolue d'un nombre

round() : arrondit un nombre à un nombre spécifié de décimales

max() : renvoie la valeur maximale d'une séquence de nombres

min() : renvoie la valeur minimale d'une séquence de nombres

sum() : renvoie la somme des éléments d'une séquence de nombres

sorted() : trie les éléments d'une séquence dans l'ordre croissant ou décroissant

zip() : crée un objet zip contenant des paires de valeurs correspondantes à partir de deux ou plusieurs séquences

Variables couramment utilisées :

x, y, z : variables générales

i, j, k : variables d'itération dans les boucles

pi : valeur de pi (3.14159265359)

True, False : valeurs booléennes

None : valeur spéciale représentant l'absence de valeur

CONSTANTS : variables représentant des constantes, souvent écrites en majuscules (par exemple, PI = 3.14159)

index : variable utilisée pour suivre la position dans une boucle

count : variable utilisée pour suivre le nombre d'occurrences d'un élément dans une séquence

result : variable utilisée pour stocker le résultat d'une opération ou d'une fonction

TD1 : introduction

TD2 : instruction conditionnelle

TD3 : les instructions de répétition

TD4 : chaines listes et tuples

TD6 : dictionnaire – enregistrements

TD7 : fonctions et modules

TD8 : les fichiers

TD9 : révision

Exercices :

# Exercice 1: Afficher tous les nombres pairs entre 0 et 100 en utilisant une boucle for.

for i in range(0, 101, 2):

print(i)

# Exercice 2: Créer une fonction qui prend en entrée une liste et renvoie la liste avec tous les éléments uniques.

def uniques(liste):

return list(set(liste))

# Exercice 3: Créer une fonction qui prend en entrée une chaîne de caractères et renvoie True si la chaîne est un palindrome.

def est\_palindrome(chaine):

return chaine == chaine[::-1]

# Exercice 4: Écrire une fonction qui prend en entrée deux listes et renvoie une liste contenant les éléments communs aux deux listes.

def communs(liste1, liste2):

return list(set(liste1) & set(liste2))

# Exercice 5: Écrire une fonction qui prend en entrée une liste de nombres et renvoie la moyenne de ces nombres.

def moyenne(liste):

return sum(liste) / len(liste)

# Exercice 6: Écrire une fonction qui prend en entrée un nombre entier n et renvoie une liste de tous les nombres premiers inférieurs ou égaux à n.

def est\_premier(nombre):

if nombre <= 1:

return False

for i in range(2, int(nombre \*\* 0.5) + 1):

if nombre % i == 0:

return False

return True

def premiers(n):

liste\_premiers = []

for i in range(2, n + 1):

if est\_premier(i):

liste\_premiers.append(i)

return liste\_premiers

# Exercice 7: Écrire une fonction qui prend en entrée une liste de mots et renvoie un dictionnaire contenant le nombre d'occurrences de chaque mot dans la liste.

def compte\_occurrences(liste\_mots):

dictionnaire\_occurrences = {}

for mot in liste\_mots:

if mot in dictionnaire\_occurrences:

dictionnaire\_occurrences[mot] += 1

else:

dictionnaire\_occurrences[mot] = 1

return dictionnaire\_occurrences

# Exemples d'utilisation des fonctions :

liste1 = [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]

liste2 = [5, 6, 7, 8, 9, 10]

liste\_mots = ["bonjour", "monde", "bonjour", "python", "python", "programmation"]

# Exercice 2

print(uniques(liste1)) # [1, 2, 3, 4, 5, 6, 7]

# Exercice 3

print(est\_palindrome("racecar")) # True

# Exercice 4

print(communs(liste1, liste2)) # [5, 6, 7]

# Exercice 5

print(moyenne(liste1)) # 4.7

# Exercice 6

print(premiers(20)) # [2, 3, 5, 7, 11, 13, 17, 19]

# Exercice 7