------------------------------------------------------------

CREATION DE LA DEFINITION DE LA FONCTION « DEMANDER NOM » :

------------------------------------------------------------

*"""  
Création d'une fonction "demander\_age" pour structurer le code : on utilise def pour définir la fonction.  
Attention, tout ce qui est à l'intérieur de def n'existe qu’à l'intérieur de la fonction def et pas sur le reste du code.  
Les fonctions possèdent donc un contexte.  
"""*

def demander\_nom():  
 reponse\_nom = "" # Nom de la variable locale a cette fonction. Même si vide, Python comprend variable de type "chaine de caractères"  
 while reponse\_nom == "":  
 reponse\_nom = input("Quel est votre nom ? ") # Si l'utilisateur ne rentre rien, ça boucle  
 # La fonction input est bloquante, on attend une réponse de l'utilisateur  
 return reponse\_nom # On retourne la valeur du nom

------------------------------------------------------------  
CREATION DE LA DEFINITION DE LA FONCTION « DEMANDER AGE » :

------------------------------------------------------------  
def demander\_age(nom\_personne): # Entre () -> paramètre de cette fonction, qui permet de rendre une fonction indépendante  
 age\_int = 0  
 # Autre possibilité : global age -> âge dans le contexte global et pas local a la fonction. Ici renvoie à age = 0.  
 while age\_int == 0: # while est une boucle. Signifie "tant que..."  
 age\_str = input(nom\_personne + " Quel est votre âge ? ") # -> âge reçu de l'utilisateur  
 # age\_str -> âge au format "chaîne de caractères"  
 try: # Permet "d'essayer"  
 age\_int = int(age\_str) # -> conversion en nombre  
 except: # Si try ne fonctionne pas, afficher l'exception ci-dessous :  
 print("ERREUR: Vous devez rentrer un nombre pour l'âge.")  
 return age\_int

-------------------------------------------------------  
AFFICHER LES INFORMATIONS D’UNE PERSONNE :

-------------------------------------------------------  
def afficher\_informations\_personne(nom, age, taille=0):  
 # taille=0 -> Paramètre optionnel car on a rentré une valeur par défaut : ici, 0.  
 print() # Permet de créer une ligne d'espace  
 print("Vous vous appelez " + nom + ", vous avez " + str(age) + " ans.")  
 print("L'an prochain vous aurez " + str(age + 1) + " ans.")

*"""  
 3 façons de concaténer des chaines de caractères (chaines formatées) :  
 1) print(f"Vous vous appelez {nom}, vous avez {age} ans.") -> f -> formater + {} qui prennent le nom de la variable  
 2) print("L'an prochain vous aurez %s ans." % (age + 1)) -> %s + donner 1 valeur de %s  
 3) print("Vous vous appelez %s, vous avez %s ans." (nom, age)) -> Remplacer plusieurs valeurs dans %s  
"""*

--------------------------------------------------------  
CREATION DE CONDITIONS :

--------------------------------------------------------  
 # condition = age >= 18 # Création d'une condition avec résultat de type booléen -> True / False)  
 # print(condition) -> Afficher les conditions  
 if age == 1 or age == 2: # if -> si (+ condition). Placé toujours en 1er dans les conditions. / or -> ou  
 print("Vous êtes un bébé.")  
 elif age < 10:  
 print("Vous êtes enfant.")  
 elif age == 1 or age == 2: # or -> ou  
 print("Vous êtes un bébé.")  
 elif 12 <= age < 18: # Autre possibilité : elif >= 12 and age < 18:  
 print ("Vous êtes adolescent.")  
 elif age == 17: # elif -> sinon si (+ condition). Toujours placé entre "if" et "else".  
 print("Vous êtes presque majeur.")  
 elif age == 18:  
 print("Tout juste majeur : Félicitations !")  
 elif age > 60:  
 print("Vous êtes sénior.") # A mettre avant age >= 18 car cas particulier : 60 > 18 !  
 elif age >= 18:  
 print("Vous êtes majeur.")  
 else: # else -> sinon. Placé toujours en bas dans les conditions. Si aucune condition ci-dessus n'est respectée, c'est celle-ci qui s'affiche.  
 print("Vous êtes mineur.")

*"""  
Attention à l'ordre dans la création des conditions vu que le code est testé de haut en bas !  
Une condition valable en haut pourrait bloquer une de celles en bas !  
Conseil : privilégier d'abord les cas particuliers avant de taper les cas généraux.  
"""*

------------------------------------------------------------  
AFFICHER LA TAILLE :

------------------------------------------------------------  
 # taille = 1.75 # Variable de type "nombre à virgule" (float) mais tout le monde ne mesure pas 1.15m.  
 if not taille == 0: # si la taille n'est PAS 0 (valeur rentrée par défaut dans def afficher\_informations\_personne)  
 print("Votre taille est : " + str(taille) + " m.")

# Demander le nom  
# Ce sont des appels à la fonction demander\_nom  
nom1 = demander\_nom()  
nom2 = demander\_nom() # Création de la variable nom pour récupérer un résultat, comme pour input  
# Appel de la fonction créée ci-dessus  
  
# Demander l'âge  
#Ce sont des appels à la fonction demander\_age  
age1 = demander\_age(nom1) # nom1 -> argument  
age2 = demander\_age(nom2) # Création de la variable nom pour récupérer un résultat, comme pour input  
# Appel de la fonction créée ci-dessus  
#Ne pas oublier de lier nom1 -> age1 et nom2 -> age2

# Afficher les résultats :  
afficher\_informations\_personne(nom1, age1) # Attention ! Ordre des paramètres important  
afficher\_informations\_personne(nom2, age2)  
  
NB\_PERSONNES = 1

*"""*

*Variable constante -> permet de mieux comprendre le code.  
Les constantes n'existent pas vraiment dans Python, c'est plus une convention pour mieux reconfigurer le code.  
"""*

-----------------------------------------------------------

LA BOUCLE FOR

-----------------------------------------------------------  
  
# Permet de boucler un certain nombre de fois / d'éléments concernant les listes  
for i in range(0, NB\_PERSONNES): # i -> nom de la variable -> déclare la variable -> index / Range -> dans l'ensemble.  
# for i in range -> pour i dans l'ensemble  
 nom = "personne" + str(i+1)  
 age = demander\_age(nom) # Attention ! On ne peut pas concaténer des noms de variable !  
 afficher\_informations\_personne(nom, age) # Ici, les arguments nom et age se rapportent aux deux lignes ci-dessus

*"""*

*Comment faire des print sur plusieurs lignes :  
Même principe que les commentaires sur plusieurs lignes.  
print("""  
 Mettre  
 ce que  
 l'on veut  
""")*

*"""*

-----------------------------------------------------------  
AJOUTER D’AUTRES ARGUMENTS A LA FONCTION PRINT :

-----------------------------------------------------------  
print("toto", 20, "ans", "taille:", 1.70) # -> affiche les uns après les autres séparés par un espace.  
# Ici, on n’a pas converti les str et int car on n’a pas de "+" mais des ",". On n’a pas concaténé.