Ejercicios de Python (HPBBC)

Sandra Mingo Ramírez

2024/25

Ejercicio 1: **Tabla de multiplicar.** Escribe un programa que muestre la tabla de multiplicar de un número que introduzca el usuario.

```
#Pide al usuario que introduzca un número por teclado
comprobacion = False
while comprobacion == False:
   try:
      num_usuario = int(input("Introduzca un número entero: "))
   except ValueError:
      print("El valor introducido no es un número entero. Pruebe otra vez.").
   else:
      comprobacion = True
#Utiliza un bucle for para mostrar la tabla de multiplicar
for i in range(1, 11):
   print(f"{num_usuario} x {i} = {num_usuario * i}")

#Bonus: haz el mismo ejercicio usando un bucle while
num_multiplicacion = 1

while num_multiplicacion <= 10:
   print(f"{num_usuario} x {i} = {num_usuario * i}")
   num_multiplicacion += 1
```

Ejercicio 2: **Suma de los primeros N números.** Escribe un programa que pida al usuario un número entero positivo nullN y luego calcule la suma de todos los números desde 1 hasta nullN.

```
#Pide al usuario que introduzca un número entero positivo
comprobacion = False
while comprobacion == False:
   try:
      num_usuario = int(input("Introduzca un número entero positivo: "))
   except ValueError:
      print("El valor introducido no es un número entero. Pruebe otra vez.").
      num_usuario = int(input("Introduzca un número entero: "))
   else:
      if num_usuario < 0:
         print("El número no es positivo.")
      else:
         comprobacion = True
resultado = 0
for i in range(1, num_usuario +1):
   resultado += 1
#Muestra el resultado de la suma
print(f"El resultado de la suma es {resultado}.")
```

Ejercicio 3: **Contar nucleótidos en una secuencia de ADN.** Escribe un programa que pida al usuario una secuencia de ADN y cuente cuántas veces aparece cada uno de los nucleótidos (A, T, C, G).

```
#Pide al usuario que ingrese una secuencia de ADN
secuencia_usuario = input("Ingrese una secuencia de ADN: ")
secuencia = secuencia_usuario.upper()
#Utiliza un bucle para recorrer la secuencia y contar la cantidad de veces que aparece cada
    nucleótido.
```

```
num_A = 0
num_T = 0
num_C = 0
num_G = 0
for base in secuencia:
   if base == "A":
      num_A += 1
   elif base == "T":
      num_T += 1
   elif base == "C":
      num_C += 1
   elif base == "G":
      num_G += 1
   else:
      print("Nucleótido no reconocido")
      break
#Muestra el conteo de cada nucleótido
print(f"Conteo de nucleótidos: \n A: {num_A} \n T: {num_T} \n C: {num_C} \n G: {num_G}")
```

Ejercicio 4: **Transcripción de ADN a ARN** Escribe un programa que realice la transcripción de una secuencia de ADN a ARN. En una secuencia de ARN, la base nitrogenada Timina (T) se reemplaza por Uracilo (U).

```
#Pide al usuario que ingrese una secuencia de ADN
secuencia_adn = input("Ingrese una secuencia de ADN: ")
#Utiliza un bucle para recorrer la secuencia y reemplazar todas las apariciones de T por U
bases_adn = "ATCG"
secuencia_arn = ""
for base in secuencia_adn:
   if base not in bases_adn:
      print("Secuencia no válida.")
      break
   if base == "T":
      secuencia_arn += "U"
   else:
      secuencia_arn += base
#Muestra la secuencia transcrita de ARN
print(f"Secuencia de ARN transcrita: {secuencia_arn}")
```

Ejercicio 5: **Devolución de cambio** Realiza un programa que proporcione el desglose en billetes y monedas de una cantidad entera de euros. Recuerda que hay billetes de 500, 200, 100, 50, 20, 10 y 5 € y monedas de 2 y 1 €.

```
euros_posibles = [500, 200, 100, 50, 20, 10, 5, 2, 1]
cambio = []
cantidad_devolver = int(input("Proporcione la cantidad a devolver: "))
idx = 0
while cantidad_devolver > 0:
   if cantidad_devolver - euros_posibles[idx] >= 0:
      cambio.append(euros_posibles[idx])
      cantidad_devolver -= euros_posibles[idx]
   else:
      idx += 1
print(cambio)
```

Ejercicio 6: **Intersecciones de listas** Diseña una función que reciba dos listas y devuelva los elementos comunes a ambas, sin repetir ninguno (intersección de conjuntos). Ejemplo: si recibe las listas

[1, 2, 1] y [2, 3, 2, 4], debe devolver la lista [2]. Escribe, también, otra función que reciba dos listas y devuelva los elementos que pertenecen a una o a otra, pero sin repetir ninguno (unión de conjuntos). Ejemplo: si recibe las listas [1, 2, 1] y [2, 3, 2, 4], devolverá [1, 2, 3, 4].

```
def elementos_comunes_listas(lista1, lista2):
   elementos_comunes = []
      for elemento in lista1:
         if elemento in lista2:
            elementos_comunes.append(elemento)
   return elementos_comunes

def union_elementos_listas(lista1, lista2): #No es eficiente al realizar dos bucles for.
    elementos_unidos = []
    for elemento in lista1:
      if elemento not in elementos_unidos:
         elementos_unidos.append(elemento)
    for elemento in lista2:
      if elemento not in elementos_unidos:
         elementos_unidos.append(elemento)
    return elementos_unidos

def union_elementos_listas_eficiente(lista1, lista2):
    return list(set(lista1) | set(lista2))
```