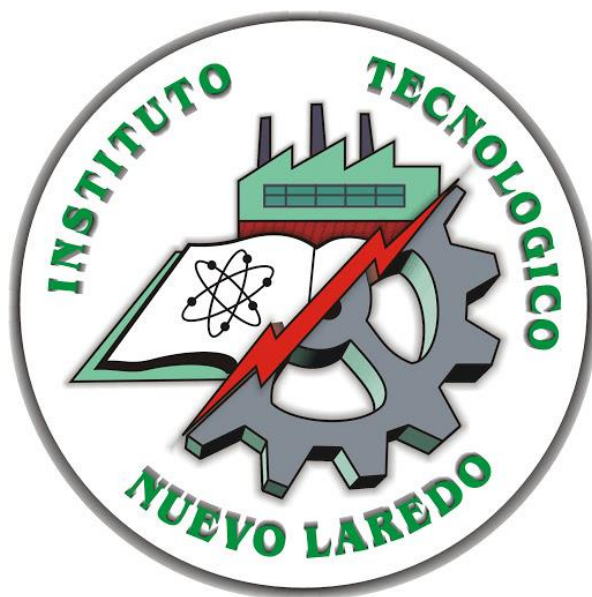


INSTITUTO TECNOLÓGICO NUEVO LAREDO

INGENIERÍA EN SISTEMAS COMPUTACIONALES



Programación Multiparadigma

Unidad 1 - Practicas

Jonathan Alonso Lara	#19100141
Bryant Fernando Gómez Domínguez	#19100196
Felipe De Jesús Martínez Salinas	#19100211

Catedrático: Ing. Luis Daniel Castillo García

Ing. Sistemas Computacionales

Nuevo Laredo, Tamaulipas, México

29/Septiembre/2023

Practicas

#1 Funciones con n parámetros

Escribir un programa que contenga una función que reciba n parámetros de tipo numérico y calcule el producto total y su suma total.

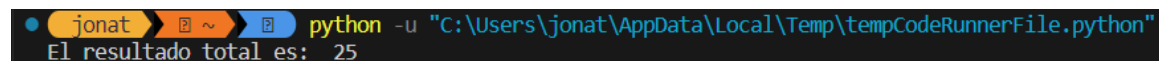
Descripción de la solución.

1. Definición de la Función: Se define una función llamada addProducts que acepta un número variable de argumentos. Los argumentos se almacenan en args.
2. Inicialización de la Variable Total: Se inicializa una variable total en 0 para almacenar la suma de los argumentos.
3. Iteración a través de los Argumentos: Se utiliza un bucle for para recorrer todos los argumentos que se pasaron a la función.
4. Suma de los Argumentos: En cada iteración, el valor del argumento actual se suma al total.
5. Impresión del Resultado: Después de sumar todos los argumentos, se imprime el resultado total de la suma.

Sentencias de código

```
def addProducts(*args):  
    total = 0  
    for arg in args:  
        total += arg  
    print('El resultado total es: ', total)  
  
addProducts(5,5,5,5,5)
```

Comprobación.



```
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.python"  
El resultado total es: 25
```

Explicación del resultado.

Esto se debe a que la función recibe cinco argumentos numéricos iguales (5) y los suma para obtener 25 como resultado total.

#2 Manejo y manipulación de elementos de una lista

Escribir un programa que almacene el abecedario en una lista, elimine de la lista las letras que ocupen posiciones múltiplos de 2, y muestre por pantalla la lista resultante.

Descripción de la solución.

El programa comienza con una lista llamada `abcList` que contiene todas las letras del alfabeto. Luego, hay una función llamada `deleteAbcPairs` que toma esta lista como argumento. Dentro de la función, se crea una nueva lista llamada `newAbcList`.

A continuación, la función itera sobre `abcList` usando un bucle `for` y la función `enumerate()`, que proporciona tanto el índice como el valor del elemento actual. Dentro del bucle, verifica si el índice es múltiplo de 2 usando la condición `index % 2 != 0`. Si el índice no es múltiplo de 2, la letra correspondiente se agrega a la lista `newAbcList`.

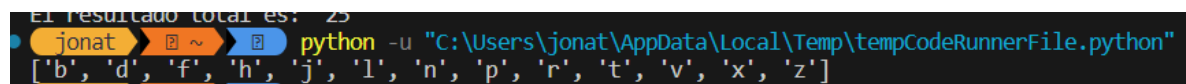
Finalmente, la función imprime `newAbcList`, que contiene las letras del alfabeto en posiciones impares.

Sentencias de código

```
abcList = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def deleteAbcPairs(abcList):
    newAbcList = []
    for index, letter in enumerate(abcList):
        if index % 2 != 0:
            newAbcList.append(letter)
    print(newAbcList)
deleteAbcPairs(abcList)
```

Comprobación.



```
El resultado total es: 25
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.py"
['b', 'd', 'f', 'h', 'j', 'l', 'n', 'p', 'r', 't', 'v', 'x', 'z']
```

Explicación del resultado.

La función `deleteAbcPairs` elimina las letras del alfabeto que están en posiciones múltiplos de 2. En este caso, dado que estamos eliminando las letras en posiciones pares (empezando desde el índice 0), las letras que permanecerán en la lista son aquellas en posiciones impares.

Cuando se ejecuta la función `deleteAbcPairs(abcList)`, se imprime la lista resultante que contiene las letras del alfabeto en posiciones impares.

#3 Entrada de datos y manipulación.

Escribir un programa que permita al usuario capturar su nombre completo e imprima su nombre de manera inversa letra por letra intercalando una letra minúscula a una mayúscula ejemplo Luis : L u l s

Descripción de la solución.

El programa solicita al usuario que ingrese su nombre. Luego, invierte el nombre ingresado y crea un nuevo nombre (newName) siguiendo una regla específica: cada letra en las posiciones pares (contando desde 0) se convierte en mayúscula, mientras que las letras en las posiciones impares permanecen en minúscula. Finalmente, imprime el nuevo nombre.

Sentencias de código

```
def reverseName():
    name = input('Ingrese su nombre: ')
    reversedName = ''.join(reversed(name))
    newName = ''
    for index, letter in enumerate(reversedName):
        if index % 2 == 0:
            newName += letter.upper()
        else:
            newName += letter
    print(newName)

reverseName()
```

Comprobación.



```
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.python"
Ingrese su nombre: Jonathann
NnAhTaNoJ
```

Explicación del resultado.

1. Entrada del Usuario
2. Inversión del Nombre: El programa invierte el nombre ingresado
3. Creación del Nuevo Nombre:

La lógica dentro del bucle for itera sobre cada letra. Dado que la posición de cada letra es:

par, se convierte a Mayuscula

'impar, se mantiene como minuscula

4. Salida: El programa imprime el resultado final.

#4 Entrada de datos y estructuración.

Revisar su retícula para escribir un programa que cree un diccionario vacío para que el usuario capture las materias y créditos de su semestre preferido (inferior a 8vo) al final imprimir en el formato "{asignatura}" tiene "{créditos}" créditos. Y la suma de todos los créditos del semestre y una LISTA de todas las materias

Descripción de la solución.

Este programa permite al usuario ingresar las materias y sus créditos para un semestre preferido (que debe ser inferior al 8vo). Utiliza un diccionario llamado `dicc_reticula` para almacenar las asignaturas como llaves y el número de créditos como valores. El programa continúa solicitando asignaturas y créditos hasta que el usuario decide detenerse. Luego imprime las asignaturas con sus respectivos créditos, una lista de todas las materias y la suma de los créditos del semestre.

Sentencias de código

```
dicc_reticula = dict([])

while True:
    asignatura = input("Digite la materia: ")
    creditos = int(input("Digite el numero de creditos de la materia: "))

    #add the vars to a dict
    dicc_reticula[asignatura] = creditos

    done = input("Desea continuar? (s/n)")
    if done == "n":
        break

for asignatura, creditos in dicc_reticula.items():
    print(f"{asignatura} tiene {creditos} creditos.")

materias = list(dicc_reticula.keys())
print(f"Las materias de este semestre son las siguientes: {asignatura}")

sum_creds = sum(dicc_reticula.values())
print(f"La suma de los creditos del semestre es de : {sum_creds}")
```

Comprobación.

```
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.python"
Digite la materia: Multiparadigma
Digite el numero de creditos de la materia: 5
Desea continuar? (s/n)s
Digite la materia: Conmutacion
Digite el numero de creditos de la materia: 5
Desea continuar? (s/n)n
Multiparadigma tiene 5 creditos.
Conmutacion tiene 5 creditos.
Las materias de este semestre son las siguientes: Conmutacion
La suma de los creditos del semestre es de : 10
```

Explicación del resultado.

1. Entrada de Datos:

El programa solicita al usuario ingresar una asignatura y el número de créditos para esa asignatura. Estos datos se almacenan como una llave-valor en el diccionario `dicc_reticula`.

2. Continuación del Proceso:

Después de ingresar una asignatura y créditos, el programa pregunta al usuario si desea continuar (`done = input("Desea continuar? (s/n)")`).

Si el usuario ingresa "s", el bucle `while` continúa y solicita más asignaturas y créditos. Si ingresa "n", el bucle se detiene (`break`).

3. Impresión de Resultados:

Después de que el usuario decide no continuar, el programa utiliza un bucle `for` para iterar sobre el diccionario `dicc_reticula`.n cada iteración, imprime la asignatura y el número de créditos en el formato "{asignatura} tiene {creditos} créditos."

Luego, crea una lista `materias` con las llaves del diccionario (`list(dicc_reticula.keys())`). Imprime la lista de materias y la suma de los créditos del semestre.

#5 Manejo de información

Escribir una función que reciba n parámetros de llave valor e imprima la información en formato "{valor}": "{llave}"

Descripción de la solución.

El programa permite al usuario ingresar pares de llave-valor y almacena estos valores en un diccionario (kwargs). Utiliza un bucle while True para continuar solicitando pares de llave-valor hasta que el usuario decida detenerse. Después de ingresar los datos, el programa imprime los valores junto con sus respectivas llaves en el formato "{valor}: {llave}".

Sentencias de código

```
kwargs = dict([])

while True:
    k = input("Escriba la llave: ")
    v = input("Escriba el valor: ")

    #Add to the dict
    kwargs[k] = v

    done = input("Desea continuar? (s/n)")
    if done == "n":
        break

for key, value in kwargs.items():
    print(value, " : ",key)
```

Comprobación.

```
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.python"
Escriba la llave: carro
Escriba el valor: 5669
Desea continuar? (s/n)s
Escriba la llave: casa
Escriba el valor: 7898
Desea continuar? (s/n)n
5669 : carro
7898 : casa
```

Explicación del resultado.

1. Entrada de Datos:

El programa solicita al usuario ingresar la llave y el valor para el diccionario.

Estos pares de llave-valor se almacenan en el diccionario kwargs.

2. Continuación del Proceso:

Después de ingresar un par de llave-valor, el programa pregunta al usuario si desea continuar (done = input("Desea continuar? (s/n)").

Si el usuario ingresa "s", el bucle while continúa y solicita más pares de llave-valor. Si ingresa "n", el bucle se detiene (break).

3. Impresión del Diccionario:

Después de que el usuario decide no continuar, el programa utiliza un bucle for para iterar sobre el diccionario kwargs.

En cada iteración, imprime el valor y la llave en el formato "{valor}: {llave}".

#6 Razonamiento y prueba de código

Escribir un programa que reciba un numero entre 0 y 20 e imprimir el numero en letra, no utilizar condicionales, máximo 5 líneas de código.

Descripción de la solución.

La función transformNumber toma un número como entrada y utiliza la función num2words del módulo word2number para convertir ese número en palabras en español. Luego imprime el resultado.

Sentencias de código

```
def transformNumber(number):  
    newNumber = num2words(number, lang='es')  
    print(newNumber)  
  
transformNumber(11)
```

Comprobación.

A screenshot of a terminal window with a dark background. The word "once" is printed in a light blue font. Below it, a light blue prompt character ">" is visible.

Explicación del resultado.

En el código, la función transformNumber(11) se llama con el número 11 como argumento.

num2words(11, lang='es') convierte el número 11 en palabras en español, que es "once".

La función imprime "once".

#7 Formateo y conversiones

Escribir un programa que muestre un menú con 2 opciones la primera opción "1.- Imprimir YYYY/MM/DD" la segunda "2.- Imprimir MM/DD/YYYY" una vez seleccionada la opción imprimir la fecha del día de hoy en el formato seleccionado.

Descripción de la solución.

El programa presenta al usuario un menú con dos opciones: imprimir la fecha en el formato "YYYY/MM/DD" o en el formato "MM/DD/YYYY". Utiliza la estructura de control match para manejar las opciones seleccionadas por el usuario. Si el usuario elige la opción 1, imprime la fecha actual en el formato "YYYY/MM/DD"; si elige la opción 2, imprime la fecha en el formato "MM/DD/YYYY". Si el usuario ingresa una opción diferente, muestra un mensaje de error.

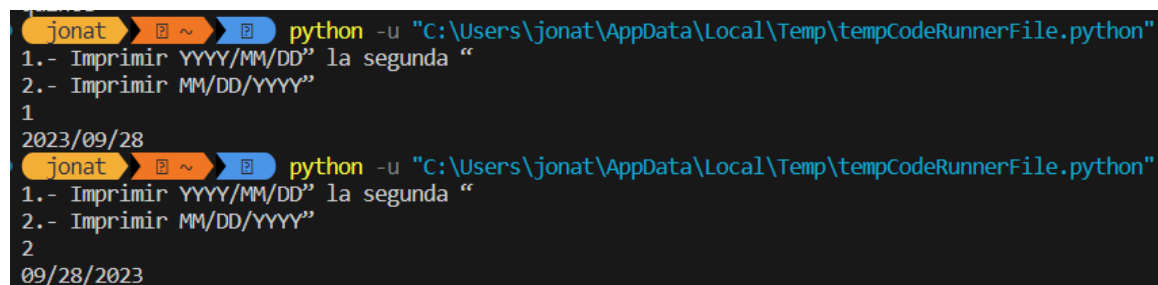
Sentencias de código

```
from datetime import date

today = date.today()

x = int(input('1.- Imprimir YYYY/MM/DD" la segunda "\n2.- Imprimir MM/DD/YYYY"\n'))
match x:
    case 1:
        d1 = today.strftime("%Y/%m/%d")
        print(d1)
    case 2:
        d2 = today.strftime("%m/%d/%Y")
        print(d2)
    case _:
        print("Error")
```

Comprobación.



```
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.py"
1.- Imprimir YYYY/MM/DD" la segunda "
2.- Imprimir MM/DD/YYYY"
1
2023/09/28
jonat ~ python -u "C:\Users\jonat\AppData\Local\Temp\tempCodeRunnerFile.py"
1.- Imprimir YYYY/MM/DD" la segunda "
2.- Imprimir MM/DD/YYYY"
2
09/28/2023
```

Explicación del resultado.

1. El programa obtiene la fecha actual utilizando el módulo date del paquete datetime.
2. Luego, el programa solicita al usuario que elija entre las opciones 1 y 2.
3. Si el usuario elige la opción 1, `today.strftime("%Y/%m/%d")` formatea la fecha actual en "YYYY/MM/DD" y la imprime.
4. Si el usuario elige la opción 2, `today.strftime("%m/%d/%Y")` formatea la fecha actual en "MM/DD/YYYY" y la imprime.
5. Si el usuario ingresa una opción diferente a 1 o 2, imprime "Error".

#8 Resumen y multi-solución

8.1.-Definir una clase usuario que contenga como atributos:

Usuario

Contraseña

Rol

Nombre

CURP

Ciudad

8.2.-Realizar un programa que contenga el siguiente menú

1.- Registro

2.- Inicio de sesión

3.- Salida

La opción de registro solicitara al usuario registrarse solicitando la información de los atributos la clase exceptuando el atributo Rol que por defecto será rol cliente, no se permitirán usuarios con CURP repetido en caso de mostrar mensaje de "El usuario ya existe"

La opción de inicio de sesión permitirá al usuario introducir sus credenciales al ser correctas desplegar en pantalla la información del usuario de lo contrario mostrar mensaje de "datos incorrectos"

8.3.- Declarar un usuario con rol "Administrador" el cual al momento de iniciar sesión despliegue la información de todos los usuarios registrados al momento.

Descripción de la solución.

Clase Usuario:

Tiene atributos como Usuario, Contraseña, Nombre, CURP, Ciudad y Rol (por defecto, "cliente").

Menú de Opciones:

El programa presenta un menú con tres opciones: Registro, Inicio de Sesión y Salida.

Registro (opción 1): Permite al usuario registrar un nuevo usuario, solicitando información como Usuario, Contraseña, Nombre, CURP y Ciudad. No se permite la duplicación de CURP.

Inicio de Sesión (opción 2): Permite a los usuarios iniciar sesión introduciendo su nombre de usuario y contraseña. Si las credenciales son correctas, se muestra la información del usuario. Para el usuario administrador, muestra la información de todos los usuarios registrados hasta el momento.

Salida (opción 3): Permite salir del programa.

Rol de Administrador:

Se crea un usuario administrador al principio del programa. El usuario administrador tiene privilegios especiales para ver la información de todos los usuarios registrados.

Sentencias de código

```
class Usuario:
    def __init__(self, usuario, contraseña, nombre, curp, ciudad,
rol="cliente"):
        self.usuario = usuario
        self.contraseña = contraseña
        self.nombre = nombre
        self.curp = curp
        self.ciudad = ciudad
        self.rol = rol

usuarios_registrados = []

def registrar_usuario():
    usuario = input("Ingrese nombre de usuario: ")
    contraseña = input("Ingrese contraseña: ")
    nombre = input("Ingrese su nombre: ")
    curp = input("Ingrese su CURP: ")
    ciudad = input("Ingrese su ciudad: ")

    # Verificar si el usuario ya existe
    for u in usuarios_registrados:
        if u.curp == curp:
            print("El usuario ya existe.")
            return

    nuevo_usuario = Usuario(usuario, contraseña, nombre, curp, ciudad)
    usuarios_registrados.append(nuevo_usuario)
    print("Usuario registrado con éxito.")

def iniciar_sesion():
    usuario = input("Ingrese nombre de usuario: ")
```

```
contraseña = input("Ingrese contraseña: ")

for u in usuarios_registrados:
    if u.usuario == usuario and u.contraseña == contraseña:
        print(f"Información del usuario: \nUsuario: {u.usuario}\nNombre: {u.nombre}\nCURP: {u.curp}\nCiudad: {u.ciudad}\nRol: {u.rol}")
        return

print("Datos incorrectos.")

# Usuario administrador
admin = Usuario("admin", "admin123", "Admin Name", "ADMIN123456789", "Admin City", "Administrador")
usuarios_registrados.append(admin)

while True:
    print("Menú:")
    print("1.- Registro")
    print("2.- Inicio de sesión")
    print("3.- Salida")
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        registrar_usuario()
    elif opcion == "2":
        iniciar_sesion()
    elif opcion == "3":
        break
    else:
        print("Opción inválida.")
```

Comprobación.

```
jonat PythonSchoolPractices main ?4 -2 & C:/Users/jonat/AppData/Local/Programs/Python/Python311/python.exe c:/Users/jo
nat/Desktop/MULTIPARADIGMA/PythonSchoolPractices/FirstExercices/ejercicio8.py
Menú:
1.- Registro
2.- Inicio de sesión
3.- Salida
Seleccione una opción: 1
Ingrese nombre de usuario: Jonathan
Ingrese contraseña: 5669
Ingrese su nombre: Jonathan Alonso Lara
Ingrese su CURP: AOSA25J
Ingrese su ciudad: Nuevo Laredo
Usuario registrado con éxito.
Menú:
1.- Registro
2.- Inicio de sesión
3.- Salida
Seleccione una opción: 1
Ingrese nombre de usuario: Jonnhy
Ingrese contraseña: 7899
Ingrese su nombre: Jonathan Alonso Lara
Ingrese su CURP: AOSA25J
Ingrese su ciudad: Nuevo Laredo
El usuario ya existe.
Menú:
1.- Registro
2.- Inicio de sesión
3.- Salida
Seleccione una opción: 1
Ingrese nombre de usuario: Fernando
Ingrese contraseña: 1456
Ingrese su nombre: Fernando Gomez Dominguez
Ingrese su CURP: GDF78AD
Ingrese su ciudad: Nuevo Laredo
Usuario registrado con éxito.
Menú:
1.- Registro
2.- Inicio de sesión
3.- Salida
Seleccione una opción: 2
Ingrese nombre de usuario: Jonathan
Ingrese contraseña: 5669
Información del usuario:
Usuario: Jonathan
Nombre: Jonathan Alonso Lara
CURP: AOSA25J
Ciudad: Nuevo Laredo
Rol: cliente
Menú:
1.- Registro
2.- Inicio de sesión
3.- Salida
Seleccione una opción: 2
Ingrese nombre de usuario: admin
Ingrese contraseña: admin123
Información del usuario:
Usuario: admin
Nombre: Admin
CURP: ADMIN123456789
Ciudad: Nuevo Laredo
Rol: Administrador
Menú:
1.- Registro
2.- Inicio de sesión
3.- Salida
Seleccione una opción: 3
jonat PythonSchoolPractices main ?4 -2
```

Explicación del resultado.

Al ejecutar el programa, el usuario puede elegir entre registrar un nuevo usuario, iniciar sesión o salir del programa.

Si elige registrar un usuario, puede ingresar sus datos. Si intenta registrar un usuario con una CURP que ya existe, el programa mostrará un mensaje de "El usuario ya existe".

Si elige iniciar sesión, debe ingresar su nombre de usuario y contraseña. Si las credenciales son correctas, el programa muestra la información del usuario.

Si inicia sesión como administrador, el programa muestra la información de todos los usuarios registrados hasta ese momento.

Al salir del programa, el bucle se detiene y el programa finaliza su ejecución.

3.- Enlace a repositorio por equipo (GitHub)

<https://github.com/FernandoGmz2001/PythonSchoolPractices/tree/main/Unidad%201%20-%20Practicas>

4.- Conclusiones y comentarios

A lo largo de estos ejercicios, hemos explorado una amplia gama de temas en Python, desde las operaciones básicas hasta tareas más complejas. Hemos reforzado nuestros conocimientos en operaciones fundamentales como las matemáticas, la manipulación de cadenas y las estructuras de control, incluyendo bucles y condicionales.

En cuanto a las estructuras de datos, hemos trabajado con listas, diccionarios y conjuntos, lo que nos ha proporcionado una comprensión profunda de cómo manejar datos de diversas maneras en Python. Además, hemos aprendido técnicas fundamentales para leer y escribir archivos, especialmente en formato CSV, habilidad esencial para el manejo de datos en aplicaciones del mundo real.

En el ámbito de la programación orientada a objetos, hemos adquirido conocimientos sobre cómo utilizar clases y objetos para modelar y organizar datos de manera más efectiva. Esta comprensión ha sido fundamental para abordar problemas más complejos y estructurar nuestro código de manera más eficiente.

Además, hemos creado programas interactivos que solicitan y procesan la entrada del usuario, mejorando así la experiencia del usuario y la interactividad de las aplicaciones. Esta habilidad para interactuar de manera efectiva con el usuario final ha agregado una dimensión práctica y dinámica a nuestros proyectos.

En resumen, estos ejercicios nos han proporcionado una sólida comprensión de Python y nos han equipado con habilidades esenciales para abordar problemas de programación de diversos niveles de complejidad.