



TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE NUEVO LAREDO

Ingeniería en Sistemas Computacionales

“Con la ciencia por la humanidad”

Unidad 1

Programación Multiparadigma - Python

Ing. Luis Daniel Castillo García

José Eduardo Gómez Zúñiga	18100184
Fernando Ángel López Soto	18100194
Jesus Antonio Villanueva Hidalgo	18100245

Nuevo Laredo, Tamps.

Septiembre 2022

Índice

Ejercicio 1	3
Ejercicio 2	4
Ejercicio 3	6
Ejercicio 4	7
Ejercicio 5	9
Ejercicio 6	10
Ejercicio 7	11
Ejercicio 8	12
Ejercicio 8.1.....	12
Ejercicio 8.2.....	14
Ejercicio 8.3.....	19
Enlace a repositorio por equipo (GitHub)	20
Conclusiones y comentarios.....	20

Ejercicio 1

Título de la práctica y descripción:

1.-Funciones con n parámetros

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

Descripción de la solución:

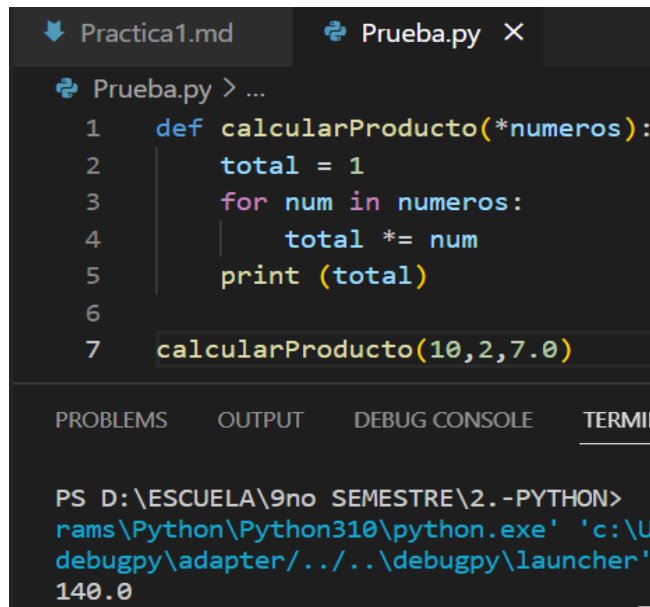
Se define una función que reciba N parámetros anteponiendo un * a la variable del parámetro de la función.

Se inicia un ciclo dónde se recorrerá la cantidad de parámetros recibidos, y dentro de una variable (total) se multiplicará por cada parámetro y se almacenará.

Sentencia de código completas:

```
def calcularProducto(*numeros):  
    total = 1          #Variable que almacenara el  
    producto final  
    for num in numeros: #Recorremos la cantidad de  
        parámetros recibidos  
        total *= num    #Multiplicamos y almacenamos el  
    valor  
    print (total)       #Imprimimos el resultado  
#calcularProducto(10,1,10.0)
```

Resultado:



```
Practica1.md  Prueba.py X
Prueba.py > ...
1  def calcularProducto(*numeros):
2      total = 1
3      for num in numeros:
4          total *= num
5      print (total)
6
7  calcularProducto(10,2,7.0)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\ESCUELA\9no SEMESTRE\2.-PYTHON> python.exe 'c:\Users\...
debugpy\adapter/../../debugpy/launcher'
140.0
```

Explicación del resultado:

Introducimos los parámetros 10,2 y 7.0 a la función, lo que hará es multiplicar el valor de la variable total por los parámetros que enviemos y almacenar el resultado. Así que la primera iteración será: $1 \times 10 = 10$, la segunda: $10 \times 2 = 20$ y, por último: $20 \times 7.0 = 140.0$. Por lo que devuelve el total=140.0

Ejercicio 2

Título de la práctica y descripción:

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 3, y muestre por pantalla la lista resultante.

Descripción de la solución:

Se declara una lista con las letras minúsculas utilizando la clase **string** por comodidad y facilidad.

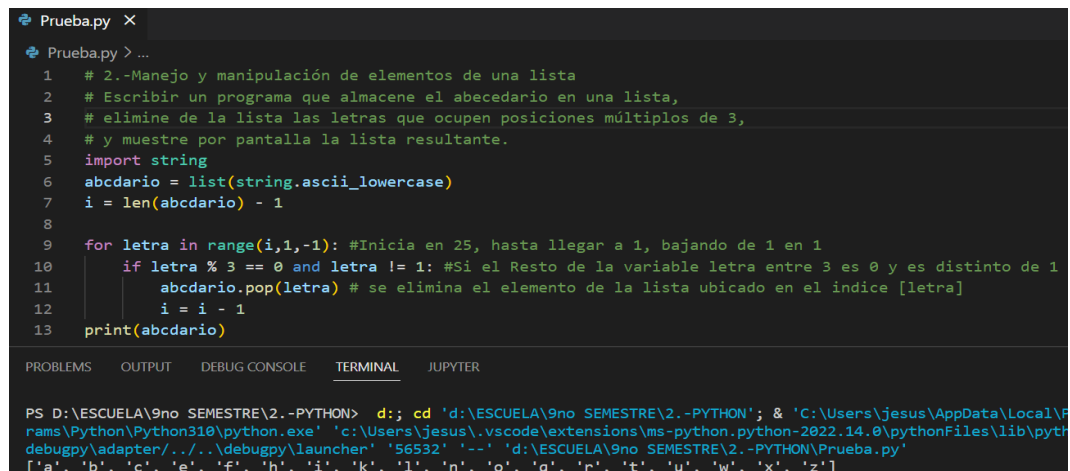
Iniciamos un ciclo en 25 disminuyendo en 1 hasta llegar a 1. Se verifica si el número en el que vamos del rango (letra) entre 3 da como resto 0 y que a su vez (letra) sea distinto de 1.

Si se cumple la condición, se elimina el elemento en la posición en la que se encuentra dentro del ciclo.

Sentencias de código completas:

```
import string
abcdario = list(string.ascii_lowercase)
i = len(abcdario) - 1
for letra in range(i,1,-1): #Inicia en 25, hasta llegar a 1, bajando de 1 en 1
    if letra % 3 == 0 and letra != 1: #Si el Resto de la variable letra entre 3 es 0 y es distinto de 1
        abcdario.pop(letra) # se elimina el elemento de la lista ubicado en el indice [letra]
        i = i - 1
print(abcdario)
```

Comprobación:



```
Prueba.py X
Prueba.py > ...
1 # 2.-Manejo y manipulación de elementos de una lista
2 # Escribir un programa que almacene el abecedario en una lista,
3 # elimine de la lista las letras que ocupen posiciones múltiplos de 3,
4 # y muestre por pantalla la lista resultante.
5 import string
6 abcdario = list(string.ascii_lowercase)
7 i = len(abcdario) - 1
8
9 for letra in range(i,1,-1): #Inicia en 25, hasta llegar a 1, bajando de 1 en 1
10     if letra % 3 == 0 and letra != 1: #Si el Resto de la variable letra entre 3 es 0 y es distinto de 1
11         abcdario.pop(letra) # se elimina el elemento de la lista ubicado en el indice [letra]
12         i = i - 1
13 print(abcdario)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

PS D:\ESCUELA\9no SEMESTRE\2.-PYTHON> d:; cd 'd:\ESCUELA\9no SEMESTRE\2.-PYTHON'; & 'C:\Users\jesus\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\jesus\.vscode\extensions\ms-python.python-2022.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '56532' '--' 'd:\ESCUELA\9no SEMESTRE\2.-PYTHON\Prueba.py'
['a', 'b', 'c', 'e', 'f', 'h', 'i', 'k', 'l', 'n', 'o', 'q', 'r', 't', 'u', 'w', 'x', 'z']
```

Explicación del resultado:

En el resultado se eliminaron las letras de la lista ubicadas en el índice que sea múltiplo de 3, verificándolo con la operación: $\text{letra} \% 3$. Por ejemplo, la letra “d” no se encuentra pues estaba en la posición 3 de la lista.

Ejercicio 3

Título de la práctica y descripción:

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.

Descripción de la solución:

Capturamos la cadena y en otra variable almacenaremos la cadena capturada, pero al revés utilizando “[::-1]”, ya que devuelve los elementos del iterable, comenzando por el último y terminando por el primero, en orden inverso a como estaban.

Sentencias de código completas:

```
nombre = input("Ingrese su nombre: ")
nombreInverso = (nombre[::-1])
for letra in nombreInverso:
    print(letra)
```

Comprobación:

```
1  nombre = input("Ingrese su nombre: ")
2  nombreInverso = (nombre[::-1])
3  for letra in nombreInverso:
4      print(letra)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\ESCUELA\9no SEMESTRE\2.-PYTHON> d:; cd 'd:\ESCUE
l\Programs\Python\Python310\python.exe' 'c:\Users\jesus
\lib\python\debugpy\adapter\..\..\debugpy\launcher' '56
Ingrese su nombre: jesus
s
u
s
e
j
```

Explicación del resultado

Una vez obtenida la cadena al revés, solo la recorremos e imprimimos el elemento en cada iteración.

Ejercicio 4

Titulo y texto de la práctica:

#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.

Descripción de la solución:

Se declara un diccionario donde se almacenarán las materias con sus créditos.

Se declara una función de agregar materia al diccionario que agrega la materia y el número de créditos ingresado por el usuario.

Se utiliza un método while para ingresar todas las materias del semestre y al terminar se recorre el diccionario mediante un ciclo for para imprimir la materia con sus créditos correspondientes y la suma de todos los créditos del semestre.

Sentencias de código completas (no imágenes):

```
reticula = {}
opcion = 1
def agregarMateria():
    nombreMateria = input('Ingrese el nombre de la materia: ')
    cantidadCreditos = int(input('Ingrese la cantidad de créditos: '))
    reticula[nombreMateria] = cantidadCreditos

while (opcion != 0):
```

```

opcion = int(input('\n[0] Salir\n[1] Ingresar Materia\n'))
if(opcion == 1): agregarMateria()

totalCreditos = 0
print('\nResumen:\n-----')
for asignatura, credits in reticula.items():
    print(f'{asignatura} tiene {credits} créditos.')
    totalCreditos += credits
print(f'\nTotal de créditos: {totalCreditos}')

```

Comprobación:

```

>
6  reticula = {}
7  opcion = 1
8
9  def agregarMateria():
10     nombreMateria = input('Ingrese el nombre de la materia: ')
11     cantidadCreditos = int(input('Ingrese la cantidad de créditos: '))
12     reticula[nombreMateria] = cantidadCreditos
13
14  while (opcion != 0):
15     opcion = int(input('\n[0] Salir\n[1] Ingresar Materia\n'))
16     if(opcion == 1): agregarMateria()
17
18  totalCreditos = 0
19  print('\nResumen:\n-----')
20  for asignatura, credits in reticula.items():
21     print(f'{asignatura} tiene {credits} créditos.')
22     totalCreditos += credits
23
24  print(f'\nTotal de créditos: {totalCreditos}')

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES GITLENS

```

[1] Ingresar Materia
1
Ingrese el nombre de la materia: IOS
Ingrese la cantidad de créditos: 5

[0] Salir
[1] Ingresar Materia
1
Ingrese el nombre de la materia: Redes
Ingrese la cantidad de créditos: 4

[0] Salir
[1] Ingresar Materia
0

Resumen:
-----
Android tiene 5 créditos.
IOS tiene 5 créditos.
Redes tiene 4 créditos.

Total de créditos: 14

```


Explicación del resultado:

Se ingresan las 3 materias del semestre 7 de la especialidad de móviles:

Android 5 creditos, IOS 5 con creditos y Redes con 4 creditos.

Al terminar se imprimen las materias con sus creditos y la suma.

Ejercicio 5

Titulo y texto de la práctica:

#5 Manejo de información

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

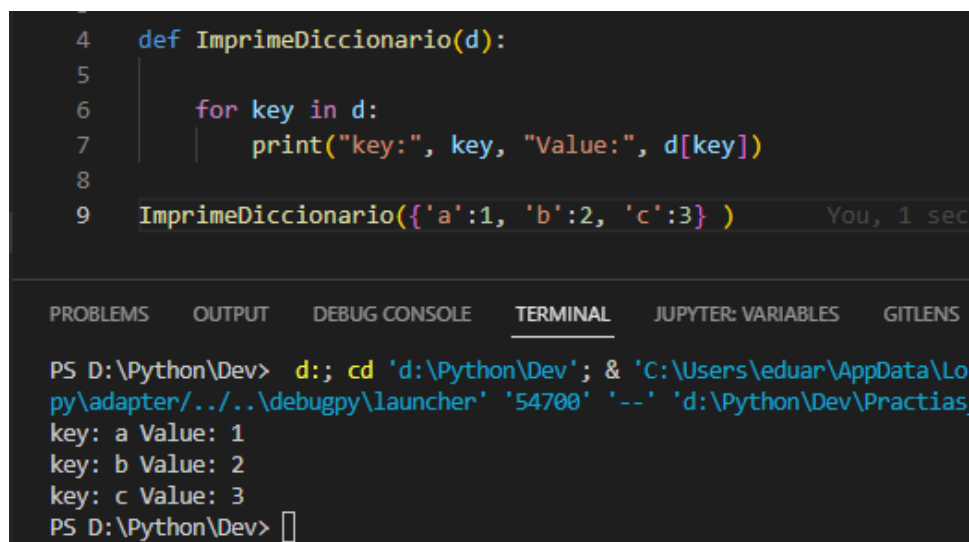
Descripción de la solución:

Se declara la función que recibe un diccionario y se recorre el diccionario imprimiendo la llave y posteriormente el valor accediendo a él.

Sentencias de código completas (no imágenes):

```
def ImprimeDiccionario(d):  
    for key in d:  
        print("key:", key, "Value:", d[key])
```

Comprobación:



```
4 def ImprimeDiccionario(d):  
5  
6     for key in d:  
7         print("key:", key, "Value:", d[key])  
8  
9 ImprimeDiccionario({'a':1, 'b':2, 'c':3} )
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES GITLENS

```
PS D:\Python\Dev> d;; cd 'd:\Python\Dev'; & 'C:\Users\eduar\AppData\Local\Microsoft\Windows\apps\python\adapter/../../debugpy/launcher' '54700' '--' 'd:\Python\Dev\Practias  
key: a Value: 1  
key: b Value: 2  
key: c Value: 3  
PS D:\Python\Dev> 
```

Explicación del resultado:

Al enviarle como parámetro a la función un diccionario, está la recorre e imprime la lleva, valor.

Ejercicio 6

Titulo y texto de la práctica:

#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:

Se declara un diccionario con los números del 1 al 20. Siendo la llave el numero entero y el valor el numero escrito.

Se pide el número entero y se consulta el valor escrito de ese número en el diccionario.

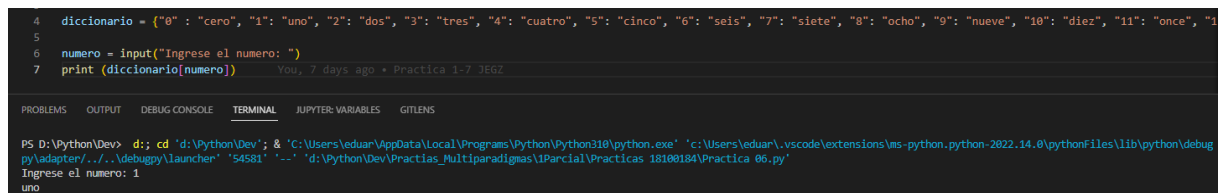
Sentencias de código completas (no imágenes):

```
diccionario = {"0": "cero", "1": "uno", "2": "dos", "3": "tres",  
"4": "cuatro", "5": "cinco", "6": "seis", "7": "siete", "8":  
"ocho", "9": "nueve", "10": "diez", "11": "once", "12": "doce",  
"13": "trece", "14": "catorce", "15": "quince", "16": "dieciseis",  
"17": "diecisiete", "18": "dieciocho", "19": "diecinueve", "20":  
"veinte"}
```

```
numero = input("Ingrese el numero: ")
```

```
print (diccionario[numero])
```

Comprobación:



```
4  diccionario = {"0": "cero", "1": "uno", "2": "dos", "3": "tres", "4": "cuatro", "5": "cinco", "6": "seis", "7": "siete", "8": "ocho", "9": "nueve", "10": "diez", "11": "once", "12": "doce", "13": "trece", "14": "catorce", "15": "quince", "16": "dieciseis", "17": "diecisiete", "18": "dieciocho", "19": "diecinueve", "20": "veinte"}  
5  
6  numero = input("Ingrese el numero: ")  
7  print (diccionario[numero])  
You, 7 days ago • Practica 1-7 JEG2  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES GITLENS  
PS D:\Python\Dev> cd 'd:\Python\Dev'; & 'c:\Users\eduar\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\eduar\.vscode\extensions\ms-python.python-2022.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '54581' '--' 'd:\Python\Dev\Practias_Multiparadigmas\IParcial\Practicas_18100184\Practica_06.py'  
Ingrese el numero: 1  
uno
```

```
3
4 diccionario = {"0": "cero", "1": "uno", "2": "dos", "3": "tres", "4": "cuatro", "5": "cinco", "6": "seis", "7": "siete", "8": "ocho", "9": "nueve", "10": "diez", "11": "once", "12": "doce", "13": "trece", "14": "catorce", "15": "quince", "16": "dieciséis", "17": "diecisiete", "18": "dieciocho", "19": "dieinueve", "20": "veinte"}
5
6 numero = input("Ingrese el número: ")
7 print(diccionario[numero])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER VARIABLES GIT LENS

PS D:\Python\Dev> d:; cd 'd:\Python\Dev'; & "C:\Users\eduar\AppData\Local\Programs\Python\Python310\python.exe" 'c:\Users\eduar\.vscode\extensions\ms-python.python-2022.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '54685' '--' 'd:\Python\Dev\Practias_Multiparadigmas\IParcial\Practicas_18108184\Practica_06.py'
Ingrese el número: 15
quince
PS D:\Python\Dev> []

Explicación del resultado:

Se ingresa el número y se muestra el resultado de consultar al diccionario el valor.

Ejercicio 7

Título y texto de la práctica:

#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:

Para esta solución se utilizó un import ya que de esta manera se puede utilizar el strftime el cual nos sirve para mostrar una fecha en determinado formato.

Sentencias de código completas (no imágenes):

```
from datetime import date

opcion = int(input('1.- Imprimir YYYY/MM/DD\n2.- Imprimir MM/DD/YYYY\n'))

print(date.today().strftime("%Y/%m/%d" if opcion == 1 else "%m/%d/%Y"))
```

Comprobación:

```
7
8  from datetime import date
9  opcion = int(input('1.- Imprimir YYYY/MM/DD\n2.- Imprimir MM/DD/YYYY\n'))
10 print(date.today().strftime("%Y/%m/%d" if opcion == 1 else "%m/%d/%Y"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS

PS D:\Python\Dev> d:; cd 'd:\Python\Dev'; & 'C:\Users\eduar\AppData\Local\Programs\Python\py\adapter\..\..\debugpy\launcher' '54468' '--' 'd:\Python\Dev\Practias_Multiparadigmas\1P

1.- Imprimir YYYY/MM/DD
2.- Imprimir MM/DD/YYYY
1
2022/09/25

```
7
8  from datetime import date
9  opcion = int(input('1.- Imprimir YYYY/MM/DD\n2.- Imprimir MM/DD/YYYY\n'))
10 print(date.today().strftime("%Y/%m/%d" if opcion == 1 else "%m/%d/%Y"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS

PS D:\Python\Dev> d:; cd 'd:\Python\Dev'; & 'C:\Users\eduar\AppData\Local\Programs\Python\py\adapter\..\..\debugpy\launcher' '54484' '--' 'd:\Python\Dev\Practias_Multiparadigmas\1P

1.- Imprimir YYYY/MM/DD
2.- Imprimir MM/DD/YYYY
2
09/25/2022

Explicación del resultado:

Se recibe un input con el valor 1 o 2 dependiendo que formato se quiere imprimir y mediante el método today() de la clase date se obtiene el día actual. Mediante strftime se le da el formato adecuado.

Ejercicio 8

Título y texto de la practica:

8# Resumen y multi-solución

Ejercicio 8.1

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

Usuario, Contraseña, Rol, Nombre, CURP, Ciudad.

Descripción de la solución:

Declarando la clase usuario como es pedido, se crea el método constructor en donde se inicializan las propiedades. De esta forma el código es más directo para su utilización al inicializar las variables.

Sentencias de código completas (no imágenes):

```
class Usuario:

    def __init__(self, usuario, password,rol, nombre, curp, ciudad) -> None:

        self.usuario = usuario

        self.password = password

        self.rol = rol

        self.nombre = nombre

        self.curp = curp

        self.ciudad = ciudad

    def __str__(self) -> str:


        return f'Usuario: {self.usuario}, {self.password}, {self.rol}, {self.nombre}, {self.curp}, {self.ciudad}'

# Comprobacion

miUsuario = Usuario("Jesus32","Michu22","Admin","Jesus","VIHJ9912","NLD")

print(miUsuario.usuario,miUsuario.rol,miUsuario.nombre,miUsuario.ciudad)
```

Comprobación:



```
13 class Usuario:
14     def __init__(self, usuario, password,rol, nombre, curp, ciudad) -> None:
15         self.usuario = usuario
16         self.password = password
17         self.rol = rol
18         self.nombre = nombre
19         self.curp = curp
20         self.ciudad = ciudad
21     def __str__(self) -> str:
22         return f'Usuario: {self.usuario}, {self.password}, {self.rol}, {self.nombre}, {self.curp}, {self.ciudad}'
23
24 # Comprobacion
25 miUsuario = Usuario("Jesus32","Michu22","Admin","Jesus","VIHJ9912","NLD")
26 print(miUsuario.usuario,miUsuario.rol,miUsuario.nombre,miUsuario.ciudad)
27
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS

PS D:\Python\Dev\Practias_Multiparadigmas> d:; cd 'd:\Python\Dev\Practias_Multiparadigmas'; & 'C:\Users\eduar\AppData\Local\Program Files\Python\Python14.0\pythonfiles\lib\python\debugpy\adapter\..\debugpy\launcher' '56126' '--' 'd:\Python\Dev\Practias_Multiparadigmas\1Parcial\Practias_Multiparadigmas.py'
Jesus32 Admin Jesus NLD

Explicación del resultado:

Se creo un usuario de prueba al cual enviaremos los parámetros y luego imprimimos algunas propiedades del objeto para saber que se hayan almacenado correctamente.

Ejercicio 8.2

8.2.- Realizar un programa que contenga el siguiente menú

1.- Registro.

2.- Inicio de sesión.

3.- Salida.

La opción de registro solicitará 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”.

Descripción de la solución:

Se declara un diccionario para almacenar los usuarios.

Se tiene la función de ingresar registro, el cual recibe los datos del usuario y antes de almacenarlos comprueba si el CURP existe con algún otro usuario. De no ser así lo almacena.

Se crea otra función de iniciar sesión, la cual recibe el usuario y la contraseña. Se verifica que existe el usuario y la contraseña. De no ser así indica que el usuario no existe.

En caso de que el usuario sea administrador se muestran todos los usuarios registrados, de caso contrario únicamente se muestra la información del usuario en cuestión.

Se creo una función leerentero() únicamente como validación por si se llega a ingresar una letra o carácter especial el programa no se detenga.

Finalmente se tiene un ciclo while() para el menú y realizar las diferentes acciones de este.

Sentencias de código completas (no imágenes):

```
usuarios = {}

# 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.

miUsuario = Usuario("Eduardo2022","1234","Administrador","Eduardo",
"GOZE000712HTSMXDA1", "Nvo Laredo") #8.3

usuarios["Eduardo2022"] = miUsuario

def Registro():

    print("Ingrese los siguientes datos:")

    usuario = input("\nIngrese su usuario: ")

    password = input("\nIngrese su contraseña: ")

    rol = "usuario"

    nombre = input("\nIngrese su nombre:")

    CURP = input("\nIngrese su CURP:")

    ciudad = input("\nIngrese la ciudad: ")

    insertar = True

    os.system ("cls")

    for curp in usuarios:

        if CURP == curp:

            print("Este CURP ya se uso previamente con otro usuario")

            insertar = False

    if insertar:

        miUsuario = Usuario(usuario,password,rol,nombre,CURP,ciudad)

        usuarios[miUsuario.usuario] = miUsuario

def InicioSesion():

    os.system ("cls")

    print("Ingrese sus credenciales: ")
```

```

username = input('Username: ')
password = input('Password: ')

if username in usuarios.keys():
    usuario = usuarios[username]
    if usuario.password == password:
        if usuario.rol == 'Administrador':
            os.system("cls")
            print('---USUARIOS---')
            print("{:<10} {:<20} {:<20} {:<15}"
{:<25}").format('Usuario', 'Nombre', 'CURP', 'Ciudad', 'Rol'))
            print("\n")
            for user in usuarios.values():
                print("{:<10} {:<20} {:<20} {:<15}"
{:<25}").format(user.usuario, user.nombre, user.curp, user.ciudad, user.rol),
end='\n\n')
        else:
            os.system("cls")
            print('--- USUARIO---')
            print(usuario)
            print("\n")
    else:
        print("Contraseña incorrecta")
else:
    print("El usuario no existe")

def lee_entero() -> int:
    while True:
        entrada = input("Seleccione una opcion: \n1.- Registro\n2.- Inicio de
sesión\n3.- Salida\n")
        try:

```



```

        entrada = int(entrada)

        return entrada

    except ValueError:

        return 9

opcion = 0

while (opcion != 3):

    opcion = lee_entero()

    if( opcion == 1):

        Registro()

        print("\n")

    if (opcion == 2):

        InicioSesion()

        print("\n")

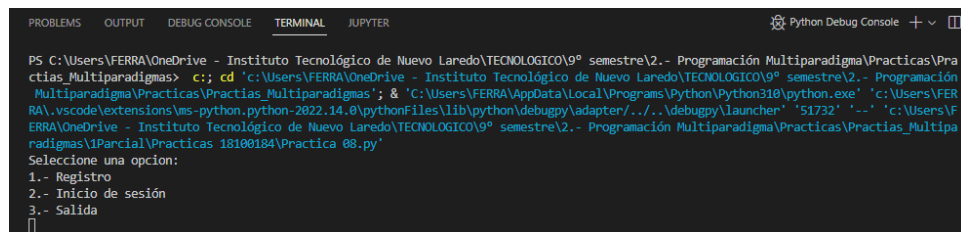
    if (opcion == 3): break

    else:

        print("Ingrese una opcion valida\n")

```

Comprobación:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Python Debug Console + - []

PS C:\Users\FERRA\OneDrive - Instituto Tecnológico de Nuevo Laredo\TECNOLOGICO\9º semestre\2.- Programación Multiparadigma\Practicas\Practias_Multiparadigmas> cd 'c:\Users\FERRA\OneDrive - Instituto Tecnológico de Nuevo Laredo\TECNOLOGICO\9º semestre\2.- Programación Multiparadigma\Practicas\Practias_Multiparadigmas'; & 'C:\Users\FERRA\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\FERRA\.vscode\extensions\ms-python.python-2022.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '51732' '--' 'c:\Users\FERRA\OneDrive - Instituto Tecnológico de Nuevo Laredo\TECNOLOGICO\9º semestre\2.- Programación Multiparadigma\Practicas\Practias_Multiparadigmas\Parcial\Practicas 18100184\Practica 08.py'
Seleccione una opcion:
1.- Registro
2.- Inicio de sesión
3.- Salida
1

```

Ejemplo con CURP repetido

```
1
Ingrese los siguientes datos:

Ingrese su usuario: FerA

Ingrese su contraseña: 123Fer

Ingrese su nombre:Fernando

Ingrese su CURP:Eduardo2022

Ingrese la ciudad: Mexico
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

El usuario ya existe

Ingrese una opcion valida

Seleccione una opcion:
1.- Registro
2.- Inicio de sesión
3.- Salida
```

Ingresar con los datos correctos, al iniciar sesión

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

---USUARIOS---
Usuario  Nombre  CURP  Ciudad  Rol

Eduardo2022  Eduardo  GOZE000712HTSMXDA1  Nvo Laredo  Administrador
```

Iniciando sesión con datos erróneo

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Ingrese sus credenciales:
Username: Fernando
Password: 12
El usuario no existe
```

Explicación del resultado:

En estos ejemplos se muestran como es el menú, el ingreso de datos, las validaciones del CURP repetido, la validación de inicio de sesión erróneo y como es que se muestran los datos para el usuario.

Ejercicio 8.3

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:

Se crea un objeto Usuario con el rol administrador y se ingresa en el diccionario de usuarios.

Sentencias de código completas (no imágenes):

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.

```
miUsuario = Usuario("Eduardo2022", "1234", "Administrador", "Eduardo",  
"GOZE000712HTSMXDA1", "Nvo Laredo") #8.3
```

```
usuarios["Eduardo2022"] = miUsuario
```

Comprobación:

```
40  
41 # 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.  
42 miUsuario = Usuario("Eduardo2022", "1234", "Administrador", "Eduardo", "GOZE000712HTSMXDA1", "Nvo Laredo") #8.3 You, 48 minutes ago • Commit Final  
43 usuarios["Eduardo2022"] = miUsuario  
44  
45 def Registro():  
    PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS  
    Ingrese sus credenciales:  
    Username: Eduardo2022  
    Password: 1234[
```

---USUARIOS---				
Usuario	Nombre	CURP	Ciudad	Rol
Eduardo2022	Eduardo	GOZE000712HTSMXDA1	Nvo Laredo	Administrador
Jose	Jose Garcia	JOGAR1080	Mexico	usuario
Maria10	Maria	MAR10A	Mty	usuario

Explicación del resultado:

Al usar la opción 2 del menú (iniciar sesión) e ingresar las credenciales de administrador.

Se muestran los diferentes usuarios registrados junto con su información.

Enlace a repositorio por equipo (GitHub)

https://github.com/18100194-FernandoALS/Practicas_Multiparadigmas.git

Conclusiones y comentarios

Estas prácticas fueron de utilidad para practicar lo visto durante el parcial, además de servir como reto de aprendizaje de lo visto en clase. La mayoría de las prácticas ya se sabía cómo resolverlas de manera lógica o en otros lenguajes y lo que faltaba era utilizar el lenguaje Python.

Este lenguaje cuenta con múltiples ventajas como lo son los módulos, la fácil declaración y utilización de variables y que todo es un objeto dentro de este lenguaje.