ITNL

Instituto Tecnológico de
Nuevo Laredo

materia:

PROGRAMACIÓN MULTIPARADIGMA

Unidad 1 - Práctica #1

Profesor:
Ing. Luis Daniel Castillo García

Alumnos:
Claudio Jose de la Rosa Torres #19100165
Myriam Yazmin Patiño Segura #19100231
Aldo Ezequiel Rodríguez Méndez 19100243

Especialidad:
Sistemas Computacionales

Fecha:
25 de septiembre del 2022

Nuevo laredo, Tamaulipas, México.

Prácticas

#1 Funciones con n parámetros

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

1. Descripción de la solución.
   Para la solución de este ejercicio se implemento el uso de 2 variables en las cuales se guardarán los 2 valores necesarios para poder realizar el producto de estos 2 de forma correcta. También, se implemento el uso del método int para convertir los valores ingresados (strings) por el usuario a un tipo de dato adecuado para poder realizar la operación (int). Por otra parte implementamos una función llamada “Calcular()” el cual recibe como parámetros las 2 variables antes mencionadas y retorna la multiplicación de estas. Por último, se manda a llamar dentro de un método “print()” la función creada para mostrar el resultado pero antes convirtiendo el valor retornado por la función en un tipo de dato string para poder usarlo en el método “print()”.

2. Sentencias de código completas (no imágenes)

```python
valor1 = int(input("Ingrese el valor del número 1: "))
valor2 = int(input("Ingrese el valor del número 2: "))

def Calcular(valor1, valor2):
    return (valor1 * valor2)

print("El producto es " + str(Calcular(valor1, valor2)))
```

3. Comprobación.

```
PROBLEMS  OUTPUT  TERMINAL  ···                                  Python

PS C:\Users\megaa\Documents\ITNL\Semestre 7\Programacion multiparadigma\Uni
dad 1\Practicas> & "C:/Program Files/Python310/python.exe" "c:/Users/megaa/
Documents/ITNL/Semestre 7/Programacion multiparadigma/Unidad 1/Practicas/1-
Funciones con N parametros.py"
Ingrese el valor del numero 1: 2
Ingrese el valor del numero 2: 2
El producto es 4
```

4. Explicación del resultado.
   Se puede observar que la multiplicación de los valores ingresados por el usuario son para la variable valor1 = 2 y valor2 = 2, por lo que el resultado de la multiplicación entre ambas variables es el valor 4.

#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.

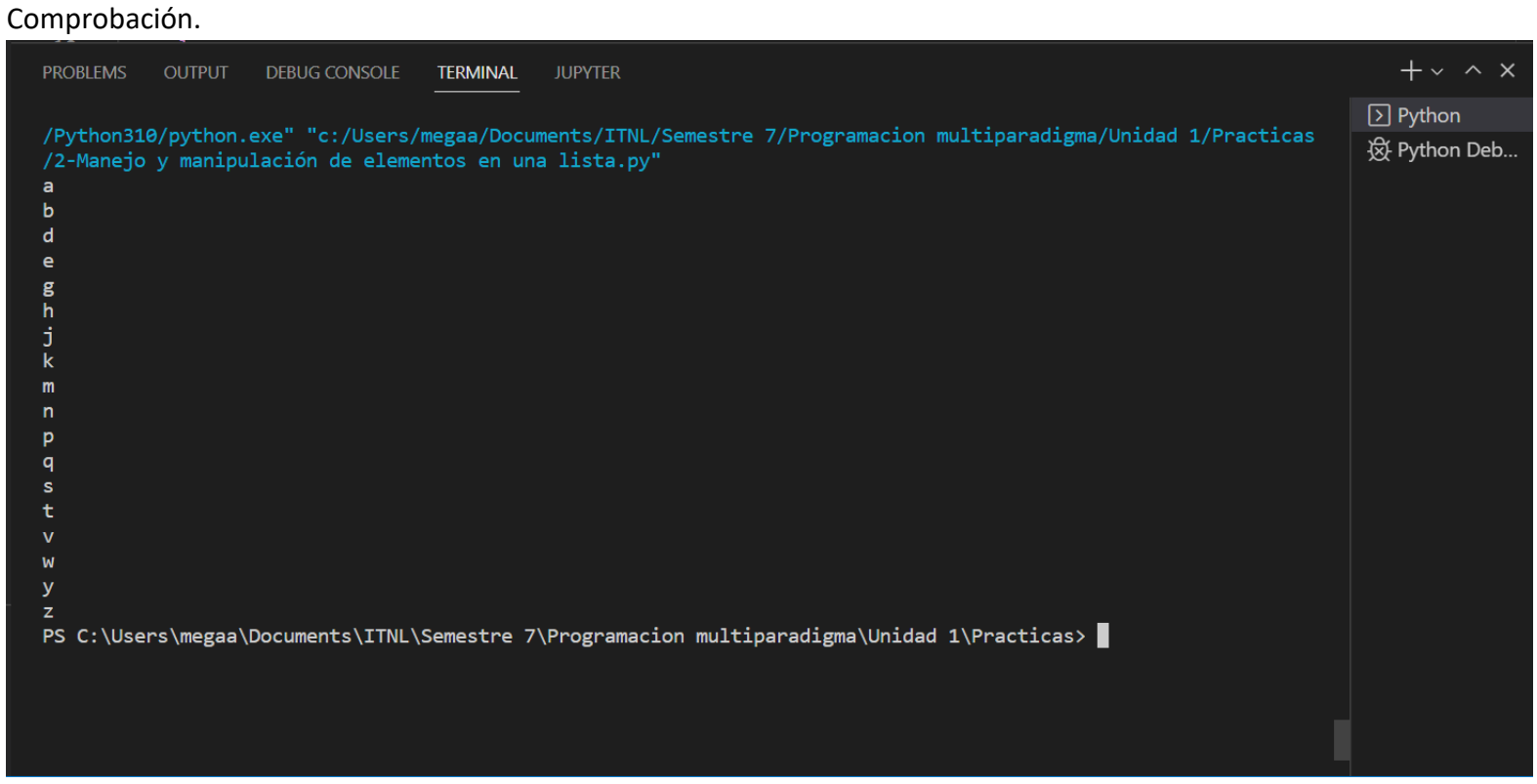
1. Descripción de la solución.
   Para resolver el ejercicio se implemento el uso de una variable auto incrementable (multiploDe3) la cual nos ayudara a eliminar por el número de índice a los elementos de la lista que son múltiplos de 3. Se implemento un ciclo for el cual nos ayudara a recorrer la lista y el primer paso a realizar es incrementar nuestra variable “multiploDe3” usando “ +=2 ”, se incrementa en 2 debido a que los índices en cualquier lista siempre empiezan por el número 0 y no por el número 1, de esta forma podemos eliminar de manera correcta los elementos que son múltiplos de 3.
   Después, se verifica que la variable (multiploDe3) no sobre pase a la cantidad de elementos que existen en la lista para que el programa sepa cuando ya se hayan eliminado todos los elementos que son múltiplos de 3. En caso de que la condición anterior se cumpla, se procederá a salir del ciclo y a mostrar los elementos que quedaron en la lista para verificar si se borraron exitosamente los elementos que eran múltiplos de 3 y en caso de que la condición anterior no se cumpla se procede a eliminar el elemento que es múltiplo de 3 de la lista y de esta forma también decrementamos en 1 la variable que nos ayuda a saber la cantidad de elementos que quedan en la lista.

# INDICE EN LISTA	0	1	**2**	3	4	**5**	6	7	**8**	9	10	**11**	12	13	**14**	15	16	**17**	18	19	**20**	21	22	**23**	24	25
ABECEDARIO	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
MULTIPLO DE 3	1	2	**3**	4	5	**6**	7	8	**9**	10	11	**12**	13	14	**15**	16	17	**18**	19	20	**21**	22	23	**24**	25	26

2. Sentencias de código completas (no imágenes)

```
lista = ['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']

multiploDe3 = 0
cantidadElementos = 26

for elemento in lista:
    multiploDe3 += 2
    if multiploDe3 > cantidadElementos:
        break
    else:
        cantidadElementos -= 1
        del lista[multiploDe3]

for elemento in lista:
    print(elemento)
```

3. Comprobación.



4. Explicación del resultado.

Como se puede observar, mediante un ciclo for se imprimieron los elementos que no eran múltiplos de 3 en la lista.

ABECEDARIO	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
MULTIPLO DE 3	1	2	**3**	4	5	**6**	7	8	**9**	10	11	**12**	13	14	**15**	16	17	**18**	19	20	**21**	22	23	**24**	25	26

#3 Entrada de datos y manipulación.

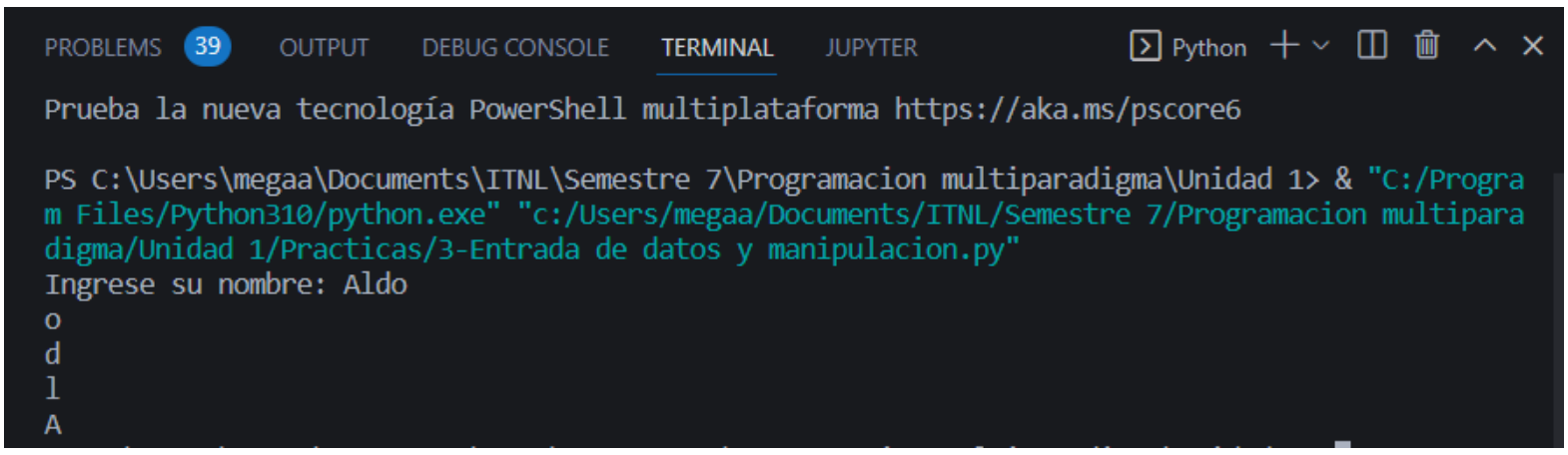
Escribir un programa que permita al usuario capturar su nombre completo e imprima su nombre de manera inversa letra por letra

1. Descripción de la solución.
   La solucion implementada en este ejercicio es definir una variable “nombre” para guardar la cadena ingresada por el usuario; también se creo una lista vacia la cual nos ayudara a almacenar en forma de lista cada carácter en forma individual. El primer paso a realizar transformar en lista la variable nombre usando el método list() para posteriormente definir un método en el cual se imprimían los caracteres guardados en la lista empezando por el elemento que tiene el índice mayor deforma hasta llegar al menor; para esto se uso una variable que ayuda a almacenar la longitud del arreglo que nos servirá como un índice auxiliar para poder acceder a los elementos de forma descendente con la ayuda de un ciclo while.

2. Sentencias de código completas (no imágenes)

```
nombre = input("Ingrese su nombre: ")
lista = []
lista = list(nombre)

def imprimirNombreInverso():
    longitudArreglo = len(lista)
    while longitudArreglo > 0:
        longitudArreglo -= 1
        print(lista[longitudArreglo])

imprimirNombreInverso()
```

3. Comprobación.



4. Explicación del resultado.

   Como se puede observar se ingreso como cadena de entrada “Aldo” la cual se procedió a guardar en una lista carácter por carácter para posteriormente a imprimirlo de manera inversa. Como se puede observar el resultado que arroja es “odlA” el cual es el resultado correcto y esperado.

#4 Entrada de datos y estructuración.

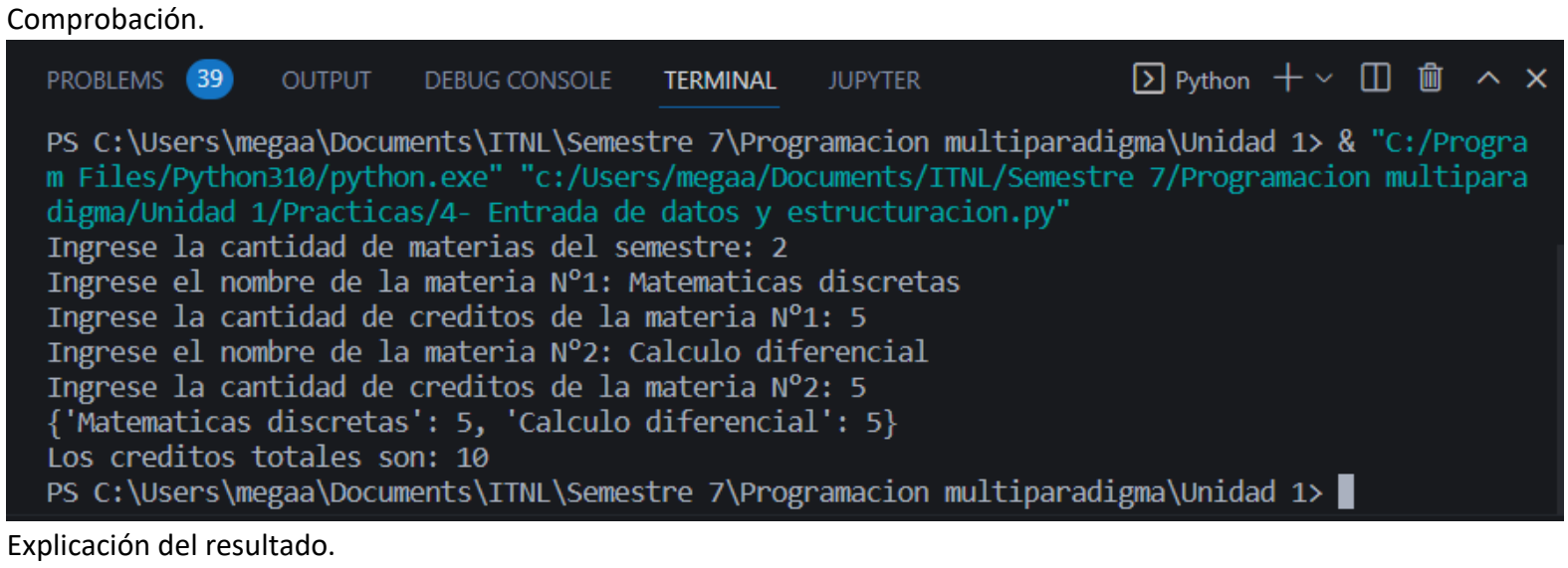
Revisar su retícula para escribir un programa que cree un diccionario vacío para que el usuario capture las materias y créditos de su semestre preferido (inferior a 8vo) al final imprimir en el formato “{asignatura}” tiene “{créditos}” créditos. Y la suma de todos los créditos del semestre

1. Descripción de la solución.
   Para la solución al problema se crearon 2 variables las cuales son cantidadDeAsignaturas la cual ayuda a poder guardar y saber cuantas asignaturas va a ingresar el usuario para poder capturarlas 1 por 1 en las variables auxiliares nombreAsginatura y créditos. Para poder crear un diccionario con los datos de cada uno de las asignaturas se implemento un ciclo while el cual iterara hasta la cantidad de asignaturas registradas. Una vez que se capturan los datos se crea una función la cual ayudara a mostrar la cantidad total de créditos acumulados por todas las asignaturas usando un ciclo for para poder recorrer el diccionario y de esta forma sumar 1 por 1 los créditos de cada asignatura registrada y por ultimo imprimir en pantalla el arreglo y también los créditos totales sumados.

2. Sentencias de código completas (no imágenes)

```python
cantidadAsignaturas = int(input("Ingrese la cantidad de materias del semestre: "))
diccionario = {}
i = 1

while cantidadAsignaturas > 0:
    cantidadAsignaturas -= 1
    nombreAsignatura = input("Ingrese el nombre de la materia N°" + str(i) + ": ")
    creditos = int(input("Ingrese la cantidad de creditos de la materia N°" + str(i) + ": "))
    diccionario.update({nombreAsignatura:creditos})

    i += 1

def imprimirCreditosTotales():
    totalCreditos = 0
    for a in diccionario:
        totalCreditos = totalCreditos + int(diccionario[a])
    print(diccionario)
    print("Los creditos totales son: " + str(totalCreditos))

imprimirCreditosTotales()
```

3. Comprobación.



4. Explicación del resultado.

Como se puede observar se ingreso primeramente la cantidad de materias del semestre para poder capturarlas y de esta manera se ingresa 1 por 1 los datos de las asignaturas. Una vez que se hace esto se nos muestra en pantalla el diccionario con sus valores registrados los cuales son correctos y ya finalmente se muestra la cantidad total de créditos acumulados.

#5 Manejo de información

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

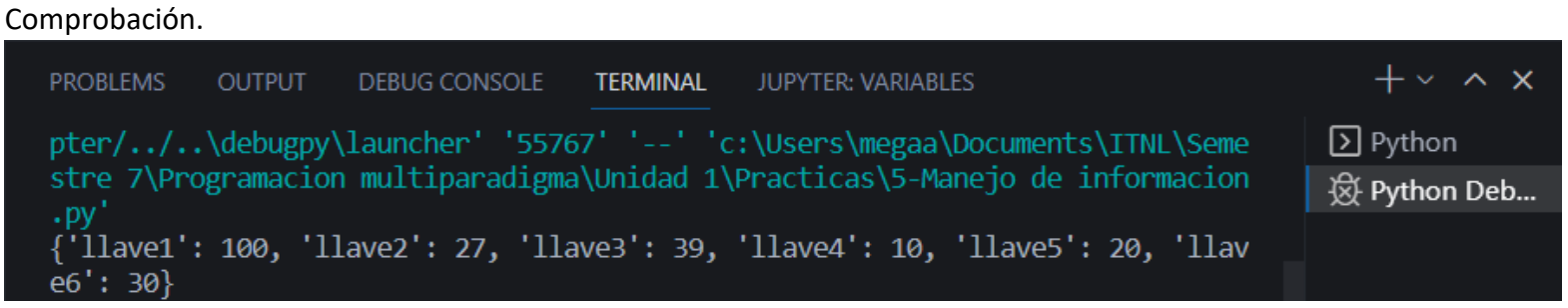
"{llave}": "{Valor}"

1. Descripción de la solución.
   Para resolver este problema se implementó una función la cual recibe N parámetros en la cual se crea un diccionario vacio que nos ayudara a almacenar las llaves con sus respectivos valores y para realizar esto se implemento un ciclo for en donde se iterara la cantidad de veces en relación a la cantidad de parámetros enviados a la función, guardando una llave y su respectivo valor de uno en uno de acuerdo a la iteración en el ciclo for. Una vez terminado el ciclo se imprime el diccionario con los datos actualizados para poder comprobar si se efectuó bien la inserción de las llaves y sus valores.

2. Sentencias de código completas (no imágenes)

```
def imprimirInformacion(**nParametros):
    diccionario = {}
    for llave in nParametros:
       diccionario.update({llave:nParametros[llave]})
    print(diccionario)

imprimirInformacion(llave1 = 100, llave2 = 27, llave3 = 39,
                    llave4 = 10, llave5 = 20, llave6 = 30)
```

3. Comprobación.



4. Explicación del resultado.

Como se puede observar se guardaron de manera correcta los parámetros enviados a la función en el diccionario auxiliar creado.

#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

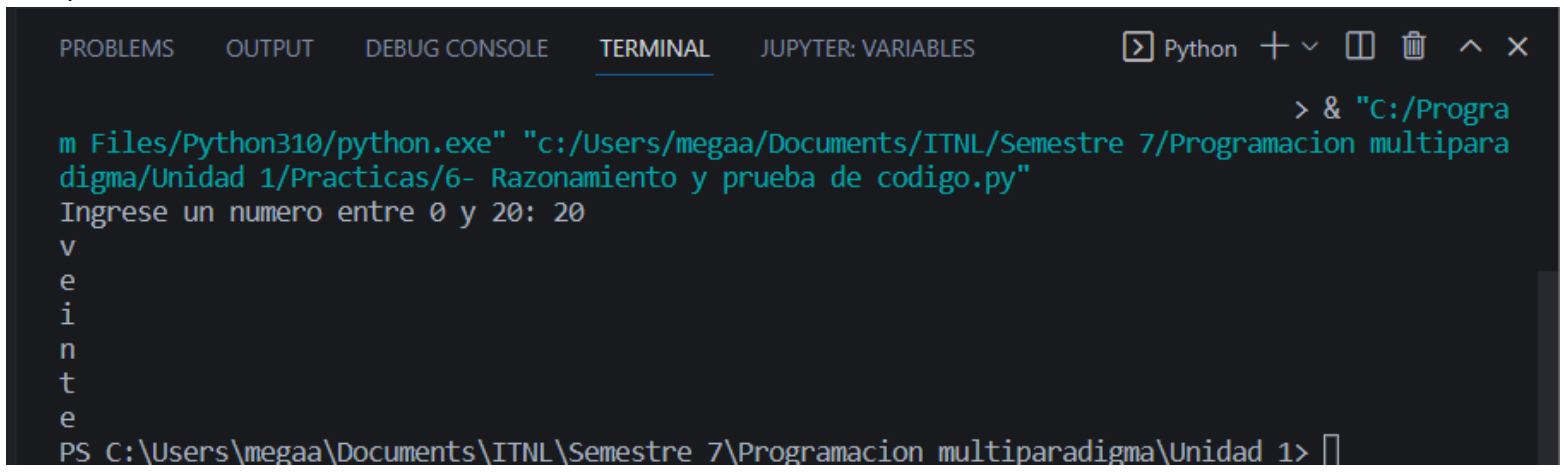
condicionales, máximo 5 líneas de código.

1. Descripción de la solución.
   Como solución al problema se implemento un diccionario en el cual se guardará como llave el numero seleccionado y como valor el numero en letra con el cual nos apoyaremos para imprimir dicho numero seleccionado. Para imprimir el numero seleccionado se implemento el uso de un ciclo for en donde se usa el método get para obtener el numero en letra del numero ingresado por el usuario, en caso de que no exista el numero seleccionado se envia un mensaje que dice "No existe" para confirmar que no se encuentra en el diccionario dicho número, en caso de que si se encuentre el número se empieza a imprmir letra por letra.

2. 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"}
numeroSeleccionado = int(input("Ingrese un numero entre 0 y 20: "))
for n in diccionario.get(numeroSeleccionado, "NO EXISTE!!"):
    print(n)
```

3. Comprobación.



4. Explicación del resultado.
   Como se puede observar, se ingreso 20 como numero a imprimir letra por letra y como se puede observar si se imprime correctamente dicho valor.

#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.

1. Descripción de la solución.
   Para la solución al problema se implemento una variable auxiliar que ayudara a verificar que formato selecciono el usuario, también se creo otra variable llamada “fechaActual” la cual almacenara la fecha actual mediante el uso de la importación de una librería llamada datetime que nos ayudara a realizar dicha consulta a la fecha actual. Se define una función que dará formato en base a la selección que eligio el usuario, para dar formato yyyy/mm/dd se usa la función de la librería strftime() con los parametros adecuados para realizar dicho formateo. Por ultimo se muestra la fecha con el formato actualizado para verificar los cambios.
2. Sentencias de código completas (no imágenes)

```
import datetime

print("-----------MENU------------- \n" +
      " 1. Imprimir YYYY/MM/DD \n" +
      " 2. Imprimir  MM/DD/YYYY \n" +
      "----------------------------")

opcionElegida = input("Ingrese la opcion a elegir: ")
fechaActual = datetime.date.today()

def darFormato():
    if opcionElegida == "1":
        print("\n La fecha actual con formato YYYY/MM/DD es " +
fechaActual.strftime("%Y-%m-%d"))
    else:
        print("\n La fecha actual con formato MM/DD/YYYY es " +
fechaActual.strftime("%m-%d-%Y"))

darFormato()
```

3. Comprobación.

4. Explicación del resultado.

Como se puede observar, para realizar el formateo se seleccionó la opción 1 y para poder comprobar si se realizó la operación correctamente se puede apreciar en un mensaje que la fecha se convirtió en lo esperado lo cual es YYYY/MM/DD

#8 Resumen y multi-solución

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

Usuario

Contraseña

Rol

Nombre

CURP

Ciudad

1. Descripción de la solución.
   Para crear la clase usuario se implemento el uso de “class” la cual nos permite definir clases junto a sus atributos, para efectos de comprobación defini valores en los atributos de la clase.

2. Sentencias de código completas (no imágenes)

```
class Usuario:
    def __init__(self,usuario,contraseña,rol,Nombre,CURP,ciudad) -> None:
        self._usuario = usuario
        self._contraseña = contraseña
        self._rol = rol
        self._Nombre = Nombre
        self._CURP = CURP
        self._ciudad = ciudad
```

3. Comprobación.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER: VARIABLES

/Semestre 7/Programacion multiparadigma/Unidad 1/Practicas/8- Resumen y multi
-solucion.py"
AldoG@outlook.com
contraseña123
Gerente
Aldo Rodriguez
ROMA0923121HTDSNLA8
Nuevo Laredo

PS C:\Users\megaa\Documents\ITNL\Semestre 7\Programacion multiparadigma\Unida
d 1>

Python
Python Deb...
```

4. Explicación del resultado.
   Como se puede apreciar en la consola, se muestra en pantalla los valores que se definieron a los atributos de la clase Usuario como puebra del funcionamiento de dicha clase.

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.

1. Descripción de la solución.
   Para realizar la solución a este problema se implementó de la clase usuario la cual nos ayudara a definir los datos que se ingresen, también el uso un diccionario el cual ayuda a guardar objetos de tipo usuario y de esta manera realizar las operaciones correspondientes en cuanto a inicio de sesión y registro de usuario nos refiramos. Para implementar el menú se hizo la simulación de un “Switch Case” ya que en si, en Python no existe la manera de implementar esto. Una vez que se realizar el menú se integra una variable auxiliar la cual servirá para almacenar el tipo de acción o instrucción que se requiera. En el caso de la operación de registro de un nuevo usuario se crean variables auxiliares que nos permitirán guardar los valores ingresados para después mediante la ayuda del constructor de la clase crear un objeto nuevo mandándole como parámetros dichas variables auxiliares y asi crear un usuario nuevo. Para verificar si un usuario ya existe antes de poder guardarlo se crea un usuario auxiliar el cual contiene los datos ingresados a guardar del nuevo usuario y se implementa una condición en la cual mediante el uso de un ciclo se empieza a recorrer el diccionario de usuarios para verificar si ya existe un usuario con el mismo CURP. Para el caso de inicio de sesión se implemento un objeto el cual nos ayuda a verificar si se ingresa un usuario con rol administrador o no, para esto se creo con anterioridad un usuario llamado admin con contraseña admin, el cual será el usuario ejemplo para el rol de administrador, primeramente se ingresan los datos de inicio de sesión y luego de esto se empieza a realizar la implementación de un ciclo para verificar en el diccionario si existe el usuario, si es un administrador o si es un cliente… esto con el fin de que si el usuario no existe en el diccionario, eso quiere decir que debemos de mostrar un mensaje correspondiente de que no existe dicho usuario, para el caso de que si es un administrador, entonces se procede a mostrar los datos de la cuenta de este administrador y con un ciclo se empieza a recorrer el diccionario y a mosrar en pantalla todos los usuarios de dicho diccionario y por último, en el caso de ser un cliente, únicamente se imprimen sus datos de la cuenta de dicho usuario. Por ultimo se implemento el uso de la librería sys para poder acceder al método exit() el cual nos ayuda a finalizar la ejecución del programa si es el caso de que se ingrese como opción del menú el numero 3 el cual es la opción de salida del programa.
2. Sentencias de código completas (no imágenes)

```python
import os
import sys

def limpiarConsola():
    command = 'clear'
    if os.name in ('nt', 'dos'):  # Si la Maquinna esta corriendo en windows, usar cls
        command = 'cls'
    os.system(command)

limpiarConsola()

opcionElegida = 0
usuariosRegistrados = {}

## clase usuario
class Usuario:
    def __init__(self,usuario,contraseña,rol,Nombre,CURP,ciudad) -> None:
        self._usuario = usuario
        self._contraseña = contraseña
        self._rol = rol
        self._Nombre = Nombre
        self._CURP = CURP
        self._ciudad = ciudad

def mostrarMenu():
    print("\n------------MENU------------- \n" +
        "     1. Registro \n" +
        "     2. Inicio de sesión \n" +
        "     3. Salida \n" +
        "------------------------------ \n")
    opcionElegida = int(input("Ingrese la opcion a elegir: "))
    limpiarConsola()
    return opcionElegida
opcionElegida = mostrarMenu()

##CREAMOS EL USUARIO ADMINISTRADOR
administrador = Usuario("admin","admin","administrador","Aldo",'1', "Nuevo Laredo")
usuariosRegistrados.update({administrador._CURP:administrador})
##-------------------------------------------------------------------------------------
```

```
def mostrarOpcion(opcionElegida):
    if opcionElegida == 1:
        print("\n-----------REGISTRO DE USUARIO-----------")
        nuevoUsuario = Usuario(input("Ingrese el correo del usuario nuevo: "),
                               input("Ingrese la contaseña del usuario nuevo: "),
                               "cliente",
                               input("Ingrese el nombre del usuario nuevo: "),
                               input("Ingrese el CURP del usuario nuevo: "),
                               input("Ingrese la ciudad del usuario nuevo: "))

        print("----------------------------------------- \n")

        bandera = 0
        for u in usuariosRegistrados:
            if usuariosRegistrados[u]._CURP == nuevoUsuario._CURP:
                print("El usuario ya existe")
                bandera = 1
        if bandera != 1:
            usuariosRegistrados.update({nuevoUsuario._CURP:nuevoUsuario})
            print("SE HA AGREADO EL NUEVO USUARIO! \n")
            bandera = 0
        mostrarOpcion(mostrarMenu())

    if opcionElegida == 2:
        nusuario = ""
        ncontraseña = ""
        nrol = ""
        nNombre = ""
        nCURP = ""
        nciudad = ""

        print("\n------------INICIO DE SESION-----------")
        nusuario = input("Ingrese el correo del usuario: ")
        ncontraseña = input("Ingrese la contraseña del usuario: ")

        usuario = Usuario(nusuario,ncontraseña,nrol,nNombre,nCURP,nciudad)

        print("---------------------------------------- \n")

        for u in usuariosRegistrados:
            if usuariosRegistrados[u]._usuario == usuario._usuario:
                usuarioContraseña = usuariosRegistrados[u]
                if usuarioContraseña._contraseña == usuario._contraseña:
                    print("------INICIO DE SESION EXITOSO!------ \n" +
                        "Usuario: " + usuariosRegistrados[u]._usuario +  "\n" +
                        "Contraseña: " + usuariosRegistrados[u]._contraseña +  "\n" +
                        "ROL: " + usuariosRegistrados[u]._rol +  "\n" +
                        "Nombre: " + usuariosRegistrados[u]._Nombre +  "\n" +
                        "CURP: " + usuariosRegistrados[u]._CURP +  "\n" +
                        "Ciudad: " + usuariosRegistrados[u]._ciudad +  "\n" +
                        "-----------------------------------\n")
                    if usuarioContraseña._rol == 'administrador':
                        print("--------USUARIOS REGISTRADOS-------")
                        for u in usuariosRegistrados:
                            print("Usuario: " + usuariosRegistrados[u]._usuario +  "\n"
+
                                "Contraseña: " + usuariosRegistrados[u]._contraseña
+  "\n" +
                                "ROL: " + usuariosRegistrados[u]._rol +  "\n" +
                                "Nombre: " + usuariosRegistrados[u]._Nombre +  "\n" +
                                "CURP: " + usuariosRegistrados[u]._CURP +  "\n" +
                                "Ciudad: " + usuariosRegistrados[u]._ciudad +  "\n" +
                                "-----------------------------------")
                else:
                    print("Datos incorrectos :c")
```

```
        mostrarOpcion(mostrarMenu())

    if opcionElegida == 3:
        print("FIN DEL PROGRAMA...")
        sys.exit()

    if opcionElegida != 1 and opcionElegida != 2 and opcionElegida != 3:
        print("La opción elegida es invalida!!! \n")
        mostrarOpcion(opcionElegida)

    mostrarMenu()

mostrarOpcion(opcionElegida)
```

3. Comprobación.

Menu

Registro

Registro de un usuario con CURP ya registrada

Inicio de sesión

- Inicio de sesión con el usuario administrador

- Inicio de sesión con un usuario cliente

Salida

4. Explicación del resultado.

Como se puede observar, se cumplieron con los resultados esperados del ejercicio a realizar.