TAREA

Arenas Vargas Yoshua

Programación para CD.

IPN

Escuela Superior de Computo

Descripción de códigos en Python.

*Código 1:

```
nombre = "Mario"
edad = 43
pi = 3.1416
vivo = True
frutas = ['Manzana', 'Fresa', 'Papaya', 'Uva']

print("Nombre: ", nombre)
print("Edad: ", edad)
print("PI: ", pi)
print("Vivo: ", vivo)
print("Frutas: ", frutas)
```

Descripción de código:
nombre es una cadena de texto que almacena el nombre "Mario".
edad es un número entero que almacena el valor 43.
pi es un número decimal (float) que almacena el valor 3.1416.
vivo es un valor booleano que almacena True (cierto).
frutas es una lista que almacena cuatro elementos: 'Manzana', 'Fresa', 'Papaya', y 'Uva'.
Al momento de imprimir obtenemos:

```
Nombre:  Mario
Edad:  43
PI:  3.1416
Vivo:  True
Frutas:  ['Manzana', 'Fresa', 'Papaya', 'Uva']
```

*Código 2:

Descripción de código:

```
nombre = "Augusto"
edad = 43
```

Descripción de código:
Dentro del código se obtienen 2 variables el nombre y la edad los cuales serán impresos con un cadena de texto.
Al momento de imprimir obtenemos:

```
Hola, yo me llamo Augusto y tengo 43 años.
```

*Código 3:

Código:

```
print("Esto es una linea \n y esto es otra linea.
```

Descripción de código:

Este código en Python imprime un mensaje en dos líneas utilizando el carácter especial \n para realizar el salto de línea. El texto antes de \n se imprime en la primera línea: "Esto es una linea". Después del salto de línea, se imprime el texto en la segunda línea: " y esto es otra linea."

Al momento de imprimir obtenemos:

Esto es una línea
 y esto es otra línea.

*Código 4:

Código:

```
# Aqui vamos a imprimir una linea
print ("El mejor ron del mundo es Zacapa XO")

# Aqui vamos a substituir el caracter de fin de linea por ::
print ("Sin embargo este ron tamben es muy bueno", end= "::")
print("Diplomatico")
print("Esto es una linea \n y esto es otra linea.
```

Descripción de código:

Este código en Python imprime primero una línea de texto: El mejor ron del mundo es Zacapa XO. Luego, en la segunda parte, se usa el argumento end en la función print para reemplazar el carácter de fin de línea por ::, lo que hace que el siguiente texto se imprima en la misma línea.

Al momento de imprimir obtenemos:

El mejor ron del mundo es Zacapa XO
Sin embargo este ron tamben es muy bueno::Diplomatico

*Código 5:

Descripción de código:
print('Amor' + ' & ' + 'Paz')

Descripción de código:
Este código en Python utiliza el operador de concatenación + para unir tres cadenas de texto. Cada una de las cadenas representa un fragmento del mensaje que se desea imprimir.
Primero, se toma la cadena 'Amor', luego se concatena con la cadena ' & ', que incluye un espacio antes y después del símbolo & para separar las palabras de forma adecuada. Finalmente, se concatena la cadena 'Paz'.

Al momento de imprimir obtenemos:
Amor & Paz

*Código 6:

Código:

```
import math
a,b,=3,4
c = math.sqrt(a**2 + b**2)
print('Cateto a: {} y cateto b: {} igual a hipotenuza c: {}'.format(a,b, c))
```

Descripción de código:
Este código en Python calcula la hipotenusa de un triángulo rectángulo utilizando el teorema de Pitágoras y la función sqrt del módulo math. Se importan las funciones matemáticas con import math, se asignan los valores a=3 y b=4 como los catetos del triángulo, y la hipotenusa c se calcula mediante la fórmula c=a2+b2c = \sqrt{a^2 + b^2}c=a2+b2.

Al momento de imprimir obtenemos:
Cateto a: 3 y cateto b: 4 igual a hipotenuza c: 5.0

*Código 7:

Código:

```
# Leyendo una cadena desde el teclado
# Imprimiendo el dato introducido y su tipo
n = input('Dame un numero: ')
print('El numero tecleado es:', n)
print('y el tipo de dato es: ', type(n))
```

Descripción de código:

Este código en Python solicita al usuario que introduzca un valor desde el teclado utilizando la función input(). El valor ingresado se almacena en la variable n, que siempre será de tipo cadena (string), independientemente de si se introduce un número o texto. Primero, el programa muestra un mensaje en la consola para que el usuario introduzca un número. Luego, imprime el valor ingresado y el tipo de dato de la variable n, que será <class 'str'> (cadena de texto). Esto se realiza mediante la función type() que devuelve el tipo de dato de una variable.

Al momento de imprimir obtenemos:

Dame un numero: 5
El numero tecleado es: 5
y el tipo de dato es: <class 'str'>

*Código 8:

Código:

```
import time
count_seconds = 5
for i in reversed(range(count_seconds + 1)):
    if i > 0:
        #Intenta primero con esta linea
        #print(i, end='>>>')
        #Despues comenta la linea anterior y descomenta la siguiente
        print(i, end='>>>', flush = True)
        time.sleep(3)
    else:
        print('Inicio')
```

Descripción de código:

Este código en Python utiliza un ciclo for para realizar una cuenta regresiva comenzando desde un número específico (en este caso, 5) hasta llegar a 0, y espera 3 segundos entre cada número utilizando la función time.sleep(). La secuencia comienza con el número 5, y por cada iteración, si el número es mayor que 0, se imprime junto con el símbolo >>>. Además, el argumento flush=True asegura que la salida se muestre de inmediato, sin retraso de búfer. Al llegar a 0, en lugar de un número, se imprime "Inicio". La espera de 3 segundos se implementa entre cada número mediante time.sleep(3), lo que retrasa la ejecución del ciclo durante ese tiempo.

Al momento de imprimir obtenemos:

5>>>4>>>3>>>2>>>1>>>Inicio

*Código 9:

```
d=27
m=9
a=2024
print(d,m,a,sep="-")
```

Descripción de código:
Este código en Python imprime tres variables (d, m, a), que representan el día, el mes y el año, respectivamente, utilizando el parámetro sep en la función print(). El parámetro sep especifica el separador entre los elementos, en este caso, un guion "-".
Al momento de imprimir obtenemos:
27-9-2024

*Código 10:

```
print('Bienvenidos a PCD 2024.!!', file=open('pcd.txt', 'w'))
```

Descripción de código:
Este código en Python escribe la cadena 'Bienvenidos a PCD 2024.!!' en un archivo llamado pcd.txt en lugar de imprimirla en la consola.
El uso de file=open('pcd.txt', 'w') dentro de la función print() redirige la salida a ese archivo en lugar de a la consola.
Debido a lo anterior no se hace ninguna acción en la consola/terminal.

*Código 11:

```
val = 'cuentos'
print(f"Cuando cuentes {val}, cuenta cuántos {val} cuentas, porque si no cuentas cuántos {val} cuentas, nunca sabrás cuántos {val} cuentas tú.")
name = 'Mario'
age = 43
print(f"Hello, My name is {name} and I'm {age} years old.")
```

Este código en Python utiliza **f-strings** para insertar variables dentro de una cadena de texto. La variable `val` tiene el valor `'cuentos'`, y se inserta en varios puntos de la cadena utilizando la sintaxis `{val}` para generar una frase repetitiva. Luego, se definen las variables `name`, que contiene el nombre `'Mario'`, y `age`, con el valor `43`. Estas variables se insertan dentro de otra cadena de texto usando el mismo formato de **f-string** para generar una frase personalizada con el nombre y la edad del usuario.

Al momento de imprimir obtenemos:
Cuando cuentes cuentos, cuenta cuántos cuentos cuentas, porque si no cuentas cuántos cuentos cuentas, nunca sabrás cuántos cuentos cuentas tú.
Hello, My name is Mario and I'm 43 years old.

*Código 12:

```
# Como imprimir la fecha de hoy
import datetime
hoy = datetime.datetime.today()
print(f"{hoy: %B %d, %Y}")
print(f"{hoy: %m %d, %Y}")
```

Descripción de código:
Este código en Python importa la librería datetime para obtener la fecha actual utilizando datetime.datetime.today(). Se utilizan **f-strings** para formatear la salida de la fecha en dos formatos diferentes. Primero, la fecha se formatea con el especificador "%B %d, %Y", donde %B es el nombre completo del mes, %d es el día y %Y es el año. En el segundo print, el formato es "%m %d, %Y", donde %m es el número del mes, %d el día y %Y el año. Este formato permite mostrar la fecha de maneras diferentes según las necesidades.

Al momento de imprimir obtenemos:
October 06, 2024
10 06, 2024

*Código 13:

```
print(f"'Imprimir comillas'")

print(f"""Imprime "dobles" comillas""")

print(f"'Imprime comillas 'simples'.'")
```

Descripción de código:
Este código en Python utiliza **f-strings** para imprimir cadenas que incluyen comillas simples y dobles. La función print() se usa para mostrar el resultado en la consola.
Al momento de imprimir obtenemos:
'Imprimir comillas'
Imprime "dobles" comillas
Imprime comillas 'simples'.

*Código 14:

```
examen = 60
libro = 10
practicas = 20
```

Descripción de código:
Este código en Python calcula la calificación total sumando las puntuaciones de examen, libro y practicas. Las variables tienen los siguientes valores: examen es 60, libro es 10 y practicas es 20.
Al momento de imprimir obtenemos:
Calificacion total: 90 de 100

*Código 15:

```
pi = 3.14159265358979323846264338327950288419716939937510
formateado = f"{pi:.4f}"
print(formateado)
```

Descripción de código:
Este código en Python define una variable pi que contiene el valor de π (pi) con una gran cantidad de decimales. Luego, se utiliza una **f-string** para formatear este valor, limitándolo a cuatro decimales con la expresión :.4f. La variable formateado almacenará el resultado formateado.
Al momento de imprimir obtenemos:
3.1416

*Código 16:

```
palabras = ["Hello", "World", "!"]
print(" ".join(palabras))
```

Descripción de código:
Este código en Python define una lista llamada palabras que contiene tres elementos: "Hello", "World" y "!". Luego, utiliza el método join() para concatenar estos elementos en una única cadena, separándolos por un espacio (" ").
Al momento de imprimir obtenemos:
Hello World !

*Código 17:

```
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(*lista)
```

Descripción de código:
Este código en Python define una lista llamada lista que contiene los números del 1 al 10. Luego, utiliza el operador de *desempaquetado* (*) en la función print(), lo que pasa cada elemento de la lista como argumento separado.
Al momento de imprimir obtenemos:
1 2 3 4 5 6 7 8 9 10

*Código 18:

```
print("augusto.ramirez", end='@')
print("gmail.com")
```

Descripción de código:
Este código en Python imprime dos partes de una dirección de correo electrónico en dos llamadas a print(). El parámetro end='@' se utiliza para evitar el salto de línea al final de la primera impresión y, en su lugar, agrega un símbolo @ antes de imprimir la siguiente parte.
Al momento de imprimir obtenemos:
augusto.ramirez@gmail.com

*Código 19:

```
# Experimentos con el fin de linea
print('C','D','M', sep='', end='')
print('X')
# Despues de un print sin el delimitador end, se restablece el fin de linea
print('27','09','2024', sep='-', end='\n')
# Otro fin de linea
print('VERDE','BLANCO','ROJO', sep='|', end='@')
print('mexico')
```

Descripción de código:
Este código realiza varios aplicaciones del parámetro end en la función print() que controla lo que se imprime al final de la cadena en lugar del salto de línea por defecto ('\n').
Al momento de imprimir obtenemos:
CDMX
27-09-2024
VERDE|BLANCO|ROJO@mexico

*Código 20:

```
nombre = "Augusto"
edad = 42
print("Mi nombre es", nombre, "y tengo", edad, "años.", end=" ")
print("Mucho gusto!")
```

Descripción de código:
Este código en Python imprime dos frases consecutivas en la misma línea utilizando el parámetro end=" " en la primera llamada a print(). El valor de end=" " añade un espacio al final de la primera línea en lugar de un salto de línea.
Al momento de imprimir obtenemos:

Mi nombre es Augusto y tengo 42 años. Mucho gusto!

*Código 21:

```
# Formato con el caracter (%)
pi = 3.14159265358979323846264338327950288419716939937510
print("Estudiantes : %2d, Edad promedio : %5.2f" % (35, 019.333))
print("Hombres : %3d, Mujeres : %2d" % (20, 15))
print("Octal: %7.3o" % (25))   # Imprimir en Octal
print("Pi: %10.4E" % (pi))   # Notacion exponencial
```

Descripción de código:
Este código en Python utiliza el operador de formato % para formatear e imprimir cadenas con valores específicos de forma personalizada.
Al momento de imprimir obtenemos:

```
Estudiantes : 35, Edad promedio : 19.33
Hombres :  20, Mujeres : 15
Octal:     031
Pi:  3.1416E+00
```

*Código 22:

```
print('I love {}. "{}!"'.format('this game!', 'Just do it!'))
print('{0} and {1}'.format('I love this game!', 'Just do it!'))
print('{1} and {0}'.format('I love this game!', 'Just do it!'))
print(f"I love {'this game'}! and \"{'Just do it'}!\"")
print(f"{'I love this game!'} and {'Just do it!'}")
```

Descripción de código:
Este código utiliza diferentes métodos para formatear e imprimir cadenas en Python. En la primera línea, se usa el método format() para insertar texto en la cadena. Luego, se muestran varias combinaciones de texto utilizando índices en el método format() para cambiar el orden de las frases. Posteriormente, se emplean **f-strings** para insertar directamente las cadenas dentro de una oración. En resumen, se presenta el amor por un juego y una frase motivacional en diferentes formatos.
Al momento de imprimir obtenemos:
I love this game!. "Just do it!"
I love this game! and Just do it!
Just do it! and I love this game!
I love this game! and "Just do it!"
I love this game! and Just do it!

*Código 23:

```
# argumentos por posicion y por nombre
print('El mejor equipo {0}, el segundo {1}, y el tercero {otro}.'
    .format('CELTICS', 'NUGGETS', otro ='BULLS'))
# con format, posiciones y formato
print("Primera posicion, entero de un digito:>>{0:3d}<<, segunda posicion
flotante:>>{1:8.2f}<<".
     format(12, 00.546))
# Cambiando posiciones
print("segundo argumento flotante:>>{1:8.2f}<< primer argumento entero:>>{0:3d}<<, ".
     format(12, 00.546))
# Argumentos por nombre
print("a: {a:5d},  Portbal: {b:8.2f}".
    format(a = 1234, b = 19.123456789))
```

Descripción de código:
Este código en Python muestra diferentes maneras de usar el método format() para imprimir cadenas con argumentos por posición y por nombre. En la primera línea, imprime el nombre de los equipos de baloncesto en un formato que utiliza índices para posicionar los argumentos y un argumento por nombre. La segunda línea muestra cómo se pueden formatear números enteros y flotantes, utilizando especificadores de formato para definir el ancho y la precisión. La tercera línea cambia el orden de los argumentos y también utiliza formateo para los números. Finalmente, la cuarta línea imprime argumentos utilizando nombres, especificando el ancho y la precisión de cada valor. En conjunto, el código ilustra cómo se pueden utilizar distintas técnicas de formateo para imprimir texto y números en Python.

Al momento de imprimir obtenemos:
El mejor equipo CELTICS, el segundo NUGGETS, y el tercero BULLS.
Primera posicion, entero de un digito:>> 12<<, segunda posicion flotante:>> 0.55<<.
segundo argumento flotante:>> 0.55<< primer argumento entero:>> 12<<.
a: 1234, Portbal: 19.12.

*Código 24:

```
texto = "BOSTON Celtics"
# Centrado
print("Texto centrado y lleno con #: ")
print(texto.center(40, '#'))
# Alineacion a la izquierda
print("Alineado a la izquierda : ")
print(texto.ljust(40, '-'))
# Alineacion a la derecha
print("Alineado a la derecha : ")
print(texto.rjust(40, '*'))
```

Descripción de código:
Este código en Python muestra cómo se puede formatear texto utilizando métodos de alineación. Primero, utiliza el método center() para centrar el texto dentro de un ancho de 40 caracteres, rellenándolo con el carácter # en ambos lados. Luego, usa el método ljust() para alinear el texto a la izquierda dentro de un ancho de 40 caracteres, completando el resto con el carácter -. Finalmente, aplica el método rjust() para alinear el texto a la derecha dentro del mismo ancho, llenando el espacio restante con el carácter *. En conjunto, el código ilustra cómo manipular la alineación del texto en Python.
Al momento de imprimir obtenemos:
Texto centrado y lleno con #:
############BOSTON Celtics############
Alineado a la izquierda :
BOSTON Celtics----------
Alineado a la derecha :
**********BOSTON Celtics