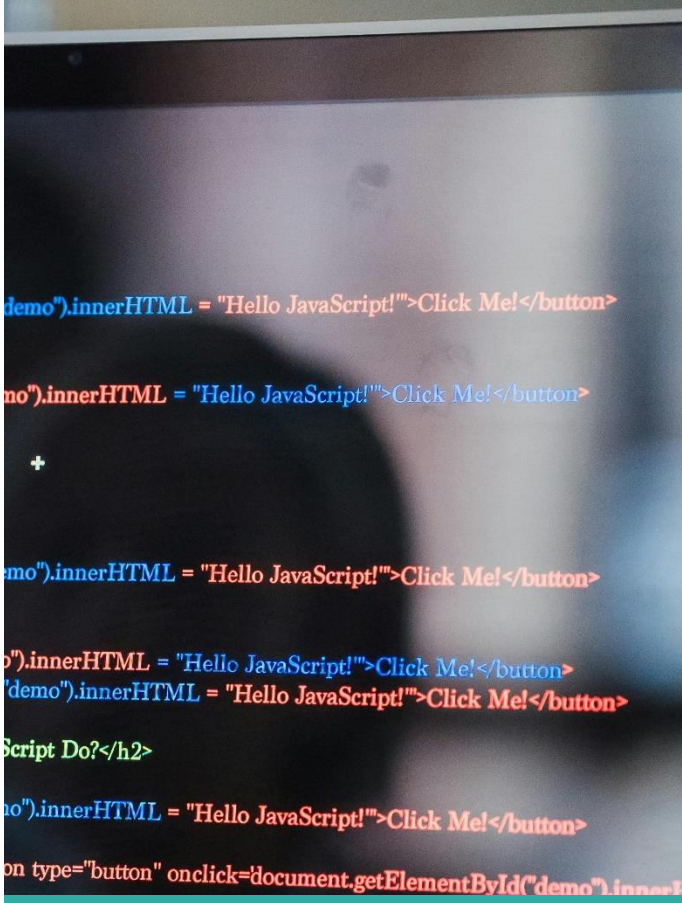
Lección 001



17 OCTUBRE

ROLOTECH

Creado por: Morales Campuzano Jose Rodrigo

Print01.py

Código

```
# Imprimir variables simples, una cadena, un numero entero

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)
```

Explicación

El código imprime los valores de varias variables, junto con sus tipos de datos:

1. Nombre (str): "Mario" → Cadena de texto.
2. Edad (int): 43 → Número entero.
3. PI (float): 3.1416 → Número decimal (de punto flotante).
4. Vivo (bool): True → Valor booleano (verdadero/falso).
5. Frutas (list): ['Manzana', 'Fresa', 'Papaya', 'Uva'] → Lista de cadenas de texto.

Cada valor se imprime precedido por su etiqueta correspondiente:

1. Nombre: "Mario"
2. Edad: 43
3. PI: 3.1416
4. Vivo: True
5. Frutas: ['Manzana', 'Fresa', 'Papaya', 'Uva']

Salída

```
$ python ./parcial1/leccion01/print01.py
Nombre:  Mario
Edad:  43
PI:  3.1416
```

```
Vivo:  True
Frutas:  ['Manzana', 'Fresa', 'Papaya', 'Uva']
```

Print02.py

```
# Intercalando cadenas y variables para imprimir

nombre = "Augusto"
edad = 43

print("Hola, yo me llamo", nombre, "y tengo", edad, "años.")
```

Explicación

1. **nombre** ya es una cadena, por lo que no requiere conversión.
2. **edad**, que es un número entero (int), se convierte automáticamente a cadena (str) para poder ser impreso junto con el texto.

El resultado final es una mezcla de texto estático y variables:

Salída

```
$ python ./parcial1/leccion01/print02.py
Hola, yo me llamo Augusto y tengo 43 años.
```

Print03.py

```
# Separando dos lineas

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

Explicación

El texto "Esto es una linea \n y esto es otra linea." contiene el carácter especial \n, que le indica a Python que debe cambiar a una nueva línea justo donde aparece.

Como resultado, el texto antes de \n se imprime en la primera línea, y el texto después de \n se imprime en la segunda línea.

Salída

```
$ python ./parcial1/leccion01/print03.py
Esto es una linea
 y esto es otra linea.
```

Print04.py

```
# 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")
```

Explicación

El código imprime dos líneas: la primera menciona "El mejor ron del mundo es Zacapa XO", y la segunda imprime "Sin embargo este ron tamben es muy bueno::Diplomatico" sin hacer salto de línea entre los dos mensajes, uniendo ambos con ::

Salída

```
$ python ./parcial1/leccion01/print04.py
El mejor ron del mundo es Zacapa XO
Sin embargo este ron tamben es muy bueno::Diplomatico
```

Print05.py

```
#Concatenando cadenas para imprimir

print('Amor' + ' & ' + 'Paz')
```

Explicación

El código imprime la cadena "Amor & Paz" al concatenar las tres partes utilizando el operador +.

Salída

```
$ python ./parcial1/leccion01/print05.py
Amor & Paz
```

Print06.py

```
# Dando formato a una cadena con variables

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))
```

Explicación

El código calcula la hipotenusa c usando el teorema de Pitágoras con los catetos a y b, y luego imprime la frase: "Cateto a: 3 y cateto b: 4 igual a hipotenusa c: 5.0" utilizando formato de cadenas con format().

Salída

```
$ python ./parcial1/leccion01/print06.py
Cateto a: 3 y cateto b: 4 igual a hipotenuza c: 5.0
```

Print07.py

```
# 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))
```

Explicación

El código solicita al usuario que ingrese un número, luego imprime ese número y muestra su tipo de dato, que será siempre str (cadena de texto) porque input() devuelve un valor de tipo cadena.

Salída

```
$ python ./parcial1/leccion01/print07.py
Dame un numero: 5
El numero tecleado es: 5
y el tipo de dato es:  <class 'str'>
```

Print08.py

```
# Conteo regresivo con y sin flush

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(1)
    else:
        print('Inicio')
```

Explicación

El código realiza una cuenta regresiva desde 5 hasta 0. Utiliza flush=True para forzar la salida inmediata de cada número en la consola sin esperar a que se complete la línea, mientras imprime >>> como separador. Al final de la cuenta, muestra "Inicio".

Salída

```
$ python ./parcial1/leccion01/print08.py
5>>>4>>>3>>>2>>>1>>>Inicio
```

Print09.py

```
# Separando datos con un separador especial
```

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

Explicación

El código imprime la fecha 27-9-2024 utilizando el carácter - como separador entre los valores de día, mes y año.

Salída

```
$ python ./parcial1/leccion01/print09.py
27-9-2024
```

Print10.py

```
# Escribiendo un archivo desde el print

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

Explicación

El código crea un archivo llamado pcd.txt y escribe en él el texto "Bienvenidos a PCD 2024.!!".

Salída

```
$ python ./parcial1/leccion01/print03.py

En un pcd.txt
Bienvenidos a PCD 2024.!!
```

Print11.py

```
# Print con f-string
```

```
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.")
```

Explicación

El código usa f-strings para insertar variables en cadenas de texto. En la primera línea, imprime un juego de palabras con la variable val ("cuentos"). En la segunda, imprime un saludo con el nombre y la edad de una persona (name y age).

Salída

```
$ python ./parcial1/leccion01/print11.py
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.
```

Print12.py

```
# Como imprimir la fecha de hoy

import datetime

hoy = datetime.datetime.today()
print(f"{hoy: %B %d, %Y}")
print(f"{hoy: %m %d, %Y}")
```

Explicación

El código imprime la fecha actual en dos formatos diferentes: el primero muestra el nombre completo del mes, día y año (" September 04, 2024") y el segundo usa números para mes, día y año (" 09 04, 2024").

Salída

```
$ python ./parcial1/leccion01/print12.py
 October 04, 2024
 10 04, 2024
```

Print13.py

```
# Como imprimir comillas

print(f"'Imprimir comillas'")

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

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

Explicación

El código imprime cadenas que contienen comillas simples y dobles dentro de ellas, utilizando diferentes tipos de comillas (', ", ''') para definir la cadena sin problemas.

Salída

```
$ python ./parcial1/leccion01/print13.py
'Imprimir comillas'
Imprime "dobles" comillas
Imprime comillas 'simples'.
```

Print14.py

```
# Operaciones con f-string
examen = 60
libro = 10
practicas = 20

print(f"Calificacion total: {examen + libro + practicas} de 100")
```

Explicación

El código calcula la calificación total sumando las variables examen, libro y practicas, e imprime el resultado como: "Calificacion total: 90 de 100**".**

Salída

```
$ python ./parcial1/leccion01/print14.py
Calificacion total: 90 de 100
```

Print15.py

```
# Dando formato a pi

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

Explicación

El código imprime el valor de pi con solo 4 decimales, mostrando: 3.1416.

Salída

```
$ python ./parcial1/leccion01/print15.py
3.1416
```

Print16.py

```
# Imprimiendo una lista
palabras = ["Hello", "World", "!"]

print(" ".join(palabras))
```

Explicación

El código imprime la lista palabras como una sola cadena con espacios entre cada elemento, resultando en: Hello World !.

Salída

```
python ./parcial1/leccion01/print16.py
```

```
Hello World !
```

Print17.py

```
#Imprimir sololos elementos de una lista

lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(*lista)
```

Explicación

El código imprime los elementos de la lista separados por espacios, mostrando: 1 2 3 4 5 6 7 8 9 10.

Salída

```
$ python ./parcial1/leccion01/print17.py
1 2 3 4 5 6 7 8 9 10
```

Print18.py

```
# Correo electronico

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

Explicación

El código imprime el correo electrónico augusto.ramirez@gmail.com en dos partes, usando end='@' para unirlas sin salto de línea.

Salída

```
$ python ./parcial1/leccion01/print18.py
augusto.ramirez@gmail.com
```

Print19.py

```
# 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')
```

Explicación

El código realiza las siguientes acciones con diferentes configuraciones del parámetro end:

1. Imprime CDMX en una sola línea, ya que el primer print() tiene end='' (sin salto de línea).
2. Imprime 27-09-2024 con el fin de línea estándar (end='\n'), lo que genera un salto de línea después.
3. Imprime VERDE|BLANCO|ROJO@ en una línea y luego mexico inmediatamente después, ya que end='@' no añade un salto de línea entre las dos instrucciones.

Salída

```
$ python ./parcial1/leccion01/print19.py
CDMX
27-09-2024
VERDE|BLANCO|ROJO@mexico
```

Print20.py

```
# Mas complicado

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

```
print("Mucho gusto!")
```

Explicación

El código imprime en la misma línea:

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

Esto sucede porque el primer print() utiliza end=" " (un espacio) al final, lo cual evita el salto de línea y permite que el segundo print() continúe en la misma línea

Salída

```
$ python ./parcial1/leccion01/print20.py
Mi nombre es Augusto y tengo 42 años. Mucho gusto!
```

Print21.py

```
# 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
```

Explicación

El código realiza las siguientes impresiones utilizando el formato printf con %:

1. Estudiantes : 35, Edad promedio : 19.33
    - Muestra el número de estudiantes (35) y la edad promedio (19.33) con formato específico.
2. Hombres : 20, Mujeres : 15
    - Muestra la cantidad de hombres y mujeres, ajustando el espacio según los especificadores.

3. Octal: 031
    - Imprime el número 25 en su representación octal (031).
4. Pi: 3.1416E+00
    - Imprime el valor de pi en notación exponencial con 4 decimales (3.1416E+00).

Salída

```
python ./parcial1/leccion01/print21.py
Estudiantes : 35, Edad promedio : 19.33
Hombres :  20, Mujeres : 15
Octal:  031
Pi: 3.1416E+00
```

Print22.py

```
# Diferentes formas de imprimir con formato

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!'}")
```

Explicación

El código muestra diferentes formas de formatear e imprimir cadenas:

1. Primera línea: I love this game!. "Just do it!"
2. Segunda línea: I love this game! and Just do it!
3. Tercera línea: Just do it! and I love this game! (cambia el orden de los argumentos)
4. Cuarta línea: I love this game! and "Just do it!" (usa f-string)
5. Quinta línea: I love this game! and Just do it! (usa f-string con variables directas)

Cada método muestra la flexibilidad para combinar texto y variables.

Salída

```
$ python ./parcial1/leccion01/print22.py
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!
```

Print23.py

```
# 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))
```

Explicación

El código muestra cómo usar argumentos por posición y por nombre con format() para personalizar la impresión:

1. **Primera línea**:
   El mejor equipo CELTICS, el segundo NUGGETS, y el tercero BULLS.
   Utiliza una combinación de argumentos por posición (0 y 1) y un argumento por nombre (otro).

2. **Segunda línea**:
   Primera posicion, entero de un digito:>> 12<<, segunda posicion flotante:>> 0.55<<
   Usa formato para especificar ancho y decimales (3d para entero y 8.2f para flotante).

3. **Tercera línea**:
   segundo argumento flotante:>> 0.55<< primer argumento entero:>> 12<<,
   Intercambia las posiciones de los argumentos 0 y 1.

4. **Cuarta línea**:
   a: 1234, Portbal: 19.12
   Utiliza argumentos por nombre (a y b) para formatear un número entero (5d) y un número flotante (8.2f).

Salída

```
$ python ./parcial1/leccion01/print23.py
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
```

Print24.py

```
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, '*'))
```

Explicación

El código muestra cómo alinear y formatear texto:

1. **Centrado**:
   Texto centrado y lleno con #:
   ##########BOSTON Celtics##########
   El texto se centra en un campo de 40 caracteres, y el espacio restante se rellena con #.

2. **Alineado a la izquierda**:
   Alineado a la izquierda :
   BOSTON Celtics----------
   El texto se alinea a la izquierda en un campo de 40 caracteres, y el espacio restante se rellena con -.

3. **Alineado a la derecha**:
   Alineado a la derecha :
   **********BOSTON Celtics
   El texto se alinea a la derecha en un campo de 40 caracteres, y el espacio restante se rellena con *.

Salída

```
python ./parcial1/leccion01/print24.py
Texto centrado y lleno con #:
##############BOSTON Celtics##############
Alineado a la izquierda :
BOSTON Celtics----------
Alineado a la derecha :
**********BOSTON Celtics
```