# clase1-1

April 23, 2024

**# Clase 1 - 20/04/2024: Introducción**

## 1 Parte 1: Generalidades

- 1.1) Comentar código
- 1.2) Introducción a la función `print()`
- 1.3) Variables: Definición y nomenclatura.
- 1.4) Asignacion Simple/Aumentada/Multiple.
- 1.5) Constantes
- 1.6) Keywords

### 1.1) Comentar código

[ ]:
```
# Este es un comentario en 1 línea
"""Este es un comentario
en varias líneas
"""
```

[ ]:
```
'Este es un comentario\nen varias líneas\n'
```

[ ]:
```
'Este es un comentario con "otro comentario" dentro del mismo mezclando␣
 ↪comillas simples y dobles'
```

[ ]:
```
'Este es un comentario con "otro comentario" dentro del mismo mezclando comillas
simples y dobles'
```

### 1.2) Introducción a la función `print()`

La función `print()` en Python es una función integrada (built-in function) que se utiliza para imprimir texto y otros tipos de datos en la consola o en otro flujo de salida.

[ ]:
```
print("Hola Level Up")
```

[ ]:
```
print(1,2,3)
```

```
1 2 3
```

[ ]:
```
# Imprimir con saltos de línea
print(1)
```

```
print(2)
print(3)
```

```
1
2
3
```

[ ]:
```
# Imprimir sin saltos de línea, especificando el final
print(1, end = ',')  # Se modifica el /n por defecto.
print(2, end = ',')
print(3, end = ',')
```

```
1, 2, 3,
```

[ ]:
```
# Especificando separador
print('Primer Variable ', 1, sep = ':', end = '; ')
print('Segunda Variable ', 5, sep = ':', end = '; ')
print('Tercer Variable ', 8, sep = ':', end = '; ')
```

```
Primer Variable :1; Segunda Variable :5; Tercer Variable :8;
```

Formas de imprimir variables dentro de la función `print()`

[ ]:
```
# Imprimiendo independientemente el texto de la variable
print("Mi variable a es ", 1)
print("Mi variable b es ", 3.14)
print("Mi variable c es ", True)
```

```
Mi variable a es  1
Mi variable b es  3.14
Mi variable c es  True
```

[ ]:
```
# Método format
print('Mis valores son {}, {}, {}'.format(1, 3.14, True))
```

```
Mis valores son 1, 3.14, True
```

[ ]:
```
# Método f-string
a = 1
b = 3.14
c = 'Manzana'
print(f'Mis valores son {a} {b} {c} ')
```

```
Mis valores son 1 3.14 Manzana
```

### 1.3) Variables

Las variables son *contenedores* o *cajas* que reservan un espacio de la memoria de la máquina y *hacen referencia* a un objeto: - int - string - bool - date

En Python toda entidad (representación de la realidad) es un objeto, que posee atributos (propiedades) y métodos (acciones que puede realizar)

Definir una variable: - Solo se aceptan `mayus`, `minus`, `digitos`, `guión bajo`. - Case Sensitive - No se pueden usar `keywords` - Convención: snake_case. (existen también camelCase y PascalCase)

[ ]:
```
mi_variable_entera = 2
print(mi_variable_entera)
```

```
2
```

[ ]:
```
mi_variable_float = 3.14
print(mi_variable_float)
```

```
3.14
```

[ ]:
```
mi_variable_string = 'Hello LVL UP'
print(mi_variable_string)
```

```
Hello LVL UP
```

### 1.4) Asignación Simple/Aumentada/Múltiple

Definir y Asignar: Son conceptos similares con la diferencia de que la definición es por primera vez y la asignación se va dando a medida que se actualiza el valor de una variable.

Asignación Simple

[ ]:
```
mi_variable_entera = 2
print(mi_variable_entera)
```

```
2
```

[ ]:
```
mi_variable_float = 3.4
mi_variable_bool = True
print(mi_variable_float)
print(mi_variable_bool)
```

```
3.4
True
```

Asignación Aumentada. Suma/Multiplicación/División

[ ]:
```
# Primero defino la variable
a = 10.5
# Realizo la asignación aumentada
a += 1
print(a)
```

```
11.5
```

[ ]:
```
# Asignación Aumentada. Multiplicación
b = 10
b *= 2
print(b)
```

```
20
```

[ ]:
```
# Asignación Aumentada. División
c = 10
c /= 3
print(c)
```

```
3.3333333333333335
```

Asignación Múltiple

[ ]:
```
a = b = c = 3
print(a,b,c)
```

```
3 3 3
```

**1.5) Constantes** > Las constantes se declaran en MAYUSCULA.

[ ]:
```
VELOCIDAD_LUZ = 300_000 # Se puede usar _ para facilidad de lectura
PRESION_ATM = 1.013
PI = 3.1416
print(VELOCIDAD_LUZ)
```

```
300000
```

**1.6) Keywords**

[ ]:
```
help('keywords')
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

# Ejercicios Parte 1

**Parte 1: Ejercicio de clase**

- Definir 3 variables (`producto1`, `producto2`, `producto3`) que representen los distintos productos o SKU de un ecommerce y los precios de los mismos (`precio1`, `precio2`, `precio3`)
- Imprimir en pantalla un mensaje con un texto que indique la promoción/oferta del produtco3 ante la compra del 1 y 2. Utilice notación `f-string`

[ ]:
```
# Definir variables
producto1 = "Microprocesador Intel I5"
producto2 = "Placa de video Nvidia"
producto3 = "Juego de teclado y mouse"
precio1 = 1000
precio2 = 1500
precio3 = 200

# Imprimir mensaje
print(f"Con la compra del {producto1} - valor {precio1} y el {producto2} -␣
 ↪valor {precio2} se obtiene de regalo un juego de {producto3} - valor␣
 ↪{precio3}- ")
```

```
Con la compra del Microprocesador Intel I5 - valor 1000 y el Placa de video
Nvidia - valor 1500 se obtiene de regalo un juego de Juego de teclado y mouse -
valor 200-
```

**Parte 1: Ejercio tarea 1**

- Define una variable `a` numérica igual a 5 y una `b` que sea float igual a 1.61 (número Aureo).

- Aumenta en 3 el valor de `a` asignación aumentada.
- Potencia en 2 el valor de `b`.
- Imprimir el resultado final de `a` en pantalla y compartirlo con la comunidad.

[ ]:
```
# Asignación simple
a = 5
b = 1.61
# Asignación aumentada
a += 3
b *= 2
print(a)
print(b)
```

```
8
3.22
```

**Parte 1: Ejercicio de tarea 2**

- Define una constante llamada `GRAVEDAD`
- Define una variable `b` y asígnale el valor de la constante `GRAVEDAD`.
- Imprimir el valor de `b` en pantalla

[ ]:
```
GRAVEDAD =  10
b = GRAVEDAD
print(b)
```

```
10
```

# 2 Parte 2: Más sobre variables. Tipos de datós básicos. Introducción a Funciones.

- 2.1) Ubicación en memoria de las variables
- 2.2) Tipos de datos básicos
- 2.3) Método `type()`. Castear una variable
- 2.4) Funciones
- 2.5) Documentación

### 2.1) Ubicación en memoria de las variables

Utilización de la función `id()`. Me da un identificador único por cada ubicación.

[ ]:
```
a = 2
b = 'Hello'
c = True
print(f'El espacio en memoria de mi variable a es {id(a)}')
print(f'El espacio en memoria de mi variable b es {id(b)}')
print(f'El espacio en memoria de mi variable c es {id(c)}')
```

```
El espacio en memoria de mi variable a es 133883252932880
El espacio en memoria de mi variable b es 133881972917744
El espacio en memoria de mi variable c es 98036334060576
```

### 2.2) Tipos de datos básicos

**2.2.1) Strings (cadena de caracteres)** - Inmutables - Métodos más utilizados: - `capitalize()`: Convetir primer caracter en UPPER Case. - `count()`: Cuenta la cantidad de ocurrencias de un strung dentro del string general. - `startswith()` y `endswith()`: Retorna True si la cadena empieza o termina con un valor especificado. - `len()`: Devuelve la cantidad de letras del string. - `find()`: Devuelve la posición del string en caso de encontrarlo.

Más métodos: https://www.w3schools.com/python/python_strings_methods.asp

[ ]:
```
mi_string = "hola buen día, es un buen día para cursar python"
```

[ ]:
```
print('Capitalizo mi_string', mi_string.capitalize())
print('Cuento la cantidad de ocurrencia de la palabra "buen"', mi_string.
 ↪count("buen"))
print('Mi string comienza con "hola"?', mi_string.startswith("hola"))
print('Mi string termina con "python"?: ', mi_string.endswith("python"))
print('La longitud de mi string es: ',len(mi_string))
print('La posición del elemento "hola" es', mi_string.find("hola"))
```

```
Capitalizo mi_string Hola buen día, es un buen día para cursar python
Cuento la cantidad de ocurrencia de la palabra "buen" 2
Mi string comienza con "hola"? True
Mi string termina con "python"?:  True
La longitud de mi string es:  48
La posición del elemento "hola" es 0
```

Un string es **iterable**. Es decir, podemos acceder a sus elementos

Accediendo al 1er elemento

[ ]:
```
elemento_0 = mi_string[0]
elemento_0
```

[ ]:
```
'h'
```

Concatenar strings

[ ]:
```
string1 = 'Hello'
string2 =  'LVL UP'
concatenado = string1+ ' ' +string2
print(concatenado)
```

```
Hello LVL UP
```

Convertir a Lista (Estructura de datos que veremos más adelante)

[ ]:
```
string_a_lista = list(string1)
print(string_a_lista)
```

```
['H', 'e', 'l', 'l', 'o']
```

Los strings son INMUTABLES. No se pueden alterar.

[ ]:
```
string1 = 'Hello' # string original
string1[0] = 'J' # Intento reemplazar H por J pero da ERROR
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-61-f30083fad467> in <cell line: 2>()
      1 string1 = 'Hello' # string original
----> 2 string1[0] = 'J' # Intento reemplazar H por J pero da ERROR

TypeError: 'str' object does not support item assignment
```

Ninguna de las operaciones vistas modifica el string original, sino que devuelve un **nuevo** objeto string.

[ ]:
```
string = 'Hello LVL UP'
trozo1 = string[5:]
```

```
print(id(string))
print(id(trozo1))
```

```
137725319778160
137725319777584
```

2.2.2) **Integers & Floats** - Inmutables - Operaciones más comunes: - Suma/Resta/Multiplicación - División Flotante/Entera/Resto - Potencia/Raíz - Incremento y Decremento

Suma. Resta. Multiplicación.

[ ]:
```
a = 4
b = 2
c = 2.5
suma = a + b + c
suma
```

[ ]:
```
8.5
```

[ ]:
```
resta = a - b
resta
```

[ ]:
```
2
```

[ ]:
```
multiplicacion = a * b
multiplicacion
```

[ ]:
```
8
```

División entera/flotante/resto.

[ ]:
```
division_flotante = a/b
division_flotante
```

[ ]:
```
2.0
```

[ ]:
```
division_entera = a//b
division_entera
```

[ ]:
```
2
```

[ ]:
```
resto = a%b      # modulo
resto
```

[ ]:
```
0
```

Potencia y Raíz

[ ]:
```
potencia = a**b
potencia
```

[ ]: 16

[ ]:
```
# Para la raíz debemos importar el módulo math
from math import sqrt
raiz = sqrt(c)
raiz
```

[ ]: 1.5811388300841898

**2.2.3) BOOL** - Inmutables - Operaciones - Comparación - Negación - Conjunción - Disyunción - Casteo a int

Comparación

[ ]:
```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

```
True
False
False
```

Negación

[ ]:
```
x = True
y = not x  # y será False
y
```

[ ]: False

Conjunción

[ ]:
```
x = True
y = False
z = x and y  # z será False
z
```

[ ]: False

Disyunción

[ ]:
```
x = True
y = False
z = x or y  # z será True
z
```

[ ]:
```
True
```

**2.2.4) DATE** - Día/mes/año - Día de la semana - Comparar Fechas - Sumar y Restar días - Formatear como String

Importamos el módulo standard `datetime`

[ ]:
```
from datetime import date, timedelta
```

[ ]:
```
# Definición
fecha = date(2024, 4, 16)
```

Obtener día/mes/año

[ ]:
```
dia = fecha.day
mes = fecha.month
year = fecha.year
print(dia, mes, year)
```

```
16 4 2024
```

Obtener dia de semana

[ ]:
```
dia_semana = fecha.weekday()
dia_semana
```

[ ]:
```
1
```

Comparar fechas

[ ]:
```
otra_fecha = date(2024, 4, 20)
es_anterior = fecha < otra_fecha
es_anterior
```

[ ]:
```
True
```

[ ]:
```
es_posterior = fecha > otra_fecha
```

Sumar o restar días

[ ]:
```
nueva_fecha = fecha + timedelta(days=7)
nueva_fecha
```

[ ]:
```
datetime.date(2024, 4, 23)
```

Calcular la diferencia en días

[ ]:
```
diferencia = otra_fecha - fecha
diferencia
```

[ ]:
```
datetime.timedelta(days=4)
```

Formatear la Fecha como cadena. Método `strftime()`

[ ]:
```
fecha_formateada = fecha.strftime("%Y-%m-%d")
fecha_formateada
```

[ ]:
```
'2024-04-16'
```

### 2.3) Método `type()`. Castear una variable

¿Cómo conocer que tipo de dato corresponde a mi variable?

[81]:
```
mi_variable1 = 5
mi_variable2 = "Hello"
print('Mi variable1 es del tipo',type(mi_variable1))
print('Mi variable2 es del tipo',type(mi_variable2))
```

```
Mi variable1 es del tipo <class 'int'>
Mi variable2 es del tipo <class 'str'>
```

¿Cómo cambiar el tipo de dato de mi variable?

[85]:
```
mi_variable = 55
casteo = str(mi_variable)
print(casteo)
print(type(casteo))
```

```
55
<class 'str'>
```

### 2.4) Funciones

- Built-in functions: Estas funciones predefinidas en Python que están disponibles en cualquier momento. Ejemplos incluyen `print()`, `input()`, `len()`, `range()`, entre otras.
- User Defined functions: Estas son funciones creadas por el programador para realizar tareas específicas. Son definidas utilizando la palabra clave `def`.

[ ]:
```
def suma(a, b):
    """Función recibe 2 valores y devuelve la suma
    Parametros: 2 valores int o float
    Resultado: Suma de los valores.
    """
    return a + b

resultado = suma(3, 5)
print(resultado)
```

```
8
```

**Función `input()`**

En Python, `input()` es una función integrada (built-in function).Se utiliza para solicitar al usuario que ingrese datos desde el teclado y devuelve la entrada del usuario como una cadena de texto (`string`).

[ ]:
```
# Ejemplo
nombre = input("Por favor, ingresa tu nombre:")
curso = input("Por favor, ingresa el curso que estás realizando: ")

# Concatenar el nombre y la edad utilizando format()
mensaje = f"Hola, {nombre}. Estás antado en el curso: {curso}"

print(mensaje)
```

```
Por favor, ingresa tu nombre:Lucho
Por favor, ingresa el curso que estás realizando: Python
Hola, Lucho. Estás antado en el curso: Python
```

### 2.5) Documentación

Obtenemos la documentación de la función con el método `help()`

[ ]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(…)
    print(value, …, sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

También podemos verlo con el método `__doc__`

[ ]:
```
print(print.__doc__)
```

```
print(value, …, sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
```

# Ejercicios Parte 2

Parte 2 - Ejercicio de clase:

- Define una función llamada `calcular_promedio` que tome tres parámetros (notas) y devuelva el promedio de esos valores.
- Define tres variables `nota1`, `nota2` y `nota3`, y asígnales valores numéricos. Llama a la función `calcular_promedio` con las tres variables `nota1`, `nota2` y `nota3` como argumentos y almacena el resultado en una nueva variable llamada promedio.
- Imprime el valor del promedio calculado.

[ ]:
```python
# Definir una función
def calcular_promedio(a, b, c):
    return (a + b + c) / 3

# Definir variables
nota1 = 7
nota2 = 8
nota3 = 9

# Calcular el promedio utilizando la función
promedio = calcular_promedio(nota1, nota2, nota3)
print("El promedio de las notas es:", promedio)
```

Parte 2 - Ejercicio de tarea 1:

- Define una función llamada `mostrar_id` que tome un argumento variable. Dentro de la función, imprime el ID de la variable utilizando la función `id()`.
- Llama a la función con diferentes tipos de variables (al menos 3) y mostrar cómo cambian sus IDs. ¿Está bien que no coincidan?

[ ]:
```python
def mostrar_id(variable):
    print("El ID de la variable es:", id(variable))

# Llamar a la función con diferentes variables
mostrar_id(5)
mostrar_id("Hola")
mostrar_id([1, 2, 3])
```

```
El ID de la variable es: 138869421310320
El ID de la variable es: 138868992276336
El ID de la variable es: 138868991769088
```

Está bien que no coincidan ya que son diferentes variables que tienen diferentes ubicaciones en memoria.

Parte 2 - Ejercicio de tarea 2

- Escribe un programa que solicite al usuario su `nombre` y su `edad` utilizando la función `input()`.
- Utiliza la notación `f-string` para concatenar el nombre y la edad en una cadena de texto que diga “Hola, [nombre]. Tienes [edad] años”.

[ ]:
```python
# Solicitar al usuario su nombre y edad
nombre = input("Por favor, ingresa tu nombre: ")
edad = input("Por favor, ingresa tu edad: ")

# Concatenar el nombre y la edad utilizando format()
mensaje = f"Hola, {nombre}. Tienes {edad} años."

# Imprimir el mensaje
print(mensaje)
```

```
Por favor, ingresa tu nombre: Luciano
Por favor, ingresa tu edad: 32
Hola, Luciano. Tienes 32 años.
```

Parte 2 - Ejercicio de tarea 3

- Define diferentes variables con los tipos de datos básicos de Python: entero, flotante, cadena de texto y booleano.
- Imprime el tipo de cada variable.

[ ]:
```python
# Definir variables con diferentes tipos de datos
entero = 5
flotante = 3.14
cadena = "Hola"
booleano = True

# Imprimir tipos de datos
print("Tipo de entero:", type(entero))
print("Tipo de flotante:", type(flotante))
print("Tipo de cadena:", type(cadena))
print("Tipo de booleano:", type(booleano))
```

Parte 2 - Ejercicio de tarea 4

- Escribe un programa que solicite al usuario dos números enteros.
- Suma los dos números y muestra el resultado en la pantalla.

[ ]:
```python
# Solicitar al usuario dos números enteros
numero1 = int(input("Por favor, ingresa el primer número entero: "))
numero2 = int(input("Por favor, ingresa el segundo número entero: "))

# Sumar los dos números
resultado = numero1 + numero2

# Mostrar el resultado
print("La suma de", numero1, "y", numero2, "es:", resultado)
```

```
Por favor, ingresa el primer número entero: 2
Por favor, ingresa el segundo número entero: 3
La suma de 2 y 3 es: 5
```

# 3 Parte 3: Condicionales

- if/else/elif
- match/case

Condicional Simple

[ ]:
```
a = 1
b  = 20

if a < b:
  print(f'a es mayor a b')
else:
  print(f'a es menor a b')
```

```
a es mayor a b
```

Condicional anidado

[ ]:
```
a = 1
b = 2
if (a == 1 or b == 2 or a < b):
    print ("Condicionales anidados")
```

```
Condicionales anidados
```

Varias condiciones. Uso de `elif`

[ ]:
```
# Con varias condiciones.Uso de Elif
if a+b == 1:
  print('La suma da 1')
elif a+b == 2:
  print('La suma da 2')
elif a+b == 3:
  print('La suma da 3')
elif a+b == 4:
  print('La suma da 4')
elif a+b == 5:
  print('La suma da 5')
else:
  print('Condicion no contemplada')
```

```
La suma da 3
```

Match y case

El uso es similar a `if/else` pero la notacion queda más limpia para casos que no tengo muchos condicionales complejos.

[87]:
```
x = 10
match x:
```

```
    case 1:
        print('x es 1')
    case 5:
        print('x es 5')
    case _:
        print('x no esta dentro de mis valores esperados <unknown>')
```

```
x no esta dentro de mis valores esperados <unknown>
```

# Ejercicios parte 3:

Parte 3 - Ejercicio de clase

- Escribe un programa que solicitará al usuario que ingrese su `edad` y su `nombre`.
- Verificará si el usuario tiene al menos 18 años y si su nombre comienza con la letra 'A' o 'a'. Si ambas condiciones se cumplen, el programa imprimirá un mensaje de acceso concedido; de lo contrario, imprimirá un mensaje de acceso denegado.

[ ]:
```
# Solicitar al usuario ingresar su edad y nombre
edad = int(input("Ingresa tu edad: "))
nombre = input("Ingresa tu nombre: ")

# Verificar si el usuario tiene al menos 18 años y su nombre comienza con 'A' o␣
 ↪'a'
acceso_concedido = (edad >= 18) and (nombre[0].lower() == 'a')

# Imprimir el resultado
if acceso_concedido:
    print("Acceso concedido. ¡Bienvenido!")
else:
    print("Acceso denegado. Lo siento, no cumples con los requisitos de acceso.
 ↪")
```

Parte 3 - Ejercicio de tarea 1 - Escribe un programa que solicite al usuario que ingrese dos números. - Luego, el programa debe calcular el resultado al cuadrado sumando el cuadrado de cada número. Si el resultado es mayor a 100, el programa debe imprimir "Supera la superficie máxima". En caso contrario, debe imprimir "Superficie habilitada".

*Tip: Recuerda que cuando utilizas la función* ***`input()`*** *debes asegurarte que el valor a sumar es int/float casteando el dato ingresado con los métodos vistos*

[ ]:
```
# Pedir al usuario que ingrese dos números
numero1 = float(input("Ingrese el primer número: "))
numero2 = float(input("Ingrese el segundo número: "))

# Calcular el resultado al cuadrado
resultado_cuadrado = numero1 ** 2 + numero2 ** 2

# Comprobar si el resultado supera la superficie máxima (100)
```

```
if resultado_cuadrado > 100:
    print("El resultado al cuadrado supera la superficie máxima.")
else:
    print("El resultado al cuadrado está dentro de la superficie habilitada.")
```