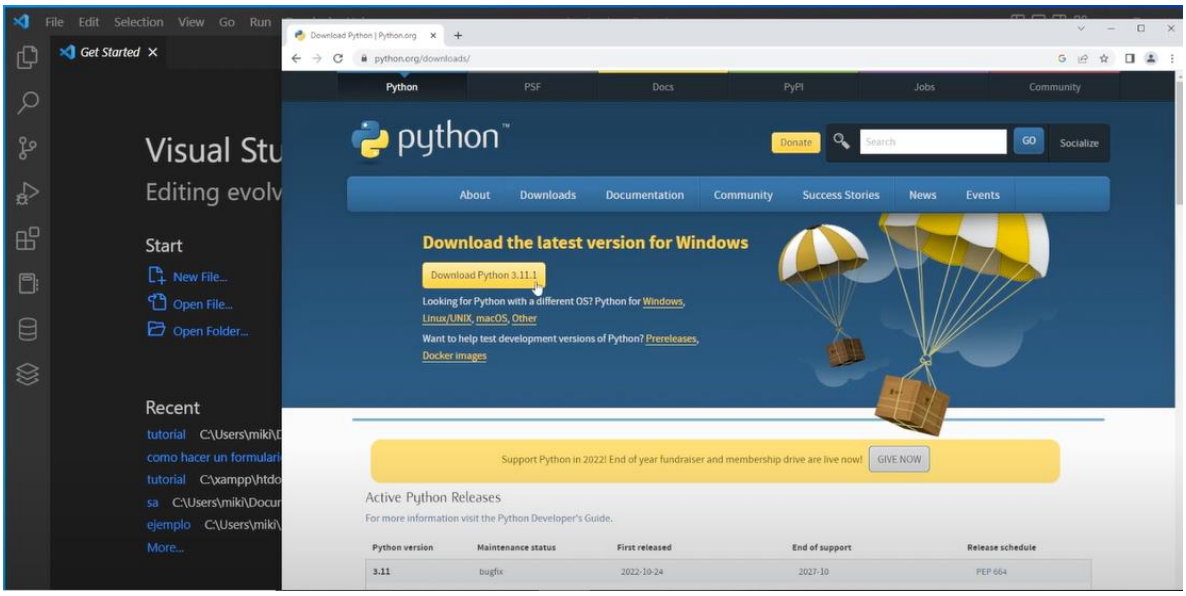
COMO SE UTILIZA PYTHON

1. Debemos de dirigirnos a la página oficial de Python https://www.python.org/downloads/

2. Abre Visual Studio Conde y dirígete a la sección de búsqueda de paquetes. Realiza una búsqueda del paquete de Python.

3. Se descarga la versión más resiente lo cual es:



4. Para poder empezar debes de tener instalado visual studio code.

VOCABULARIO DE PYTHON

print ():

es una función incorporada que se utiliza para mostrar información en la salida estándar, que generalmente es la consola en la que se ejecuta el programa. Esta función es muy útil para mostrar mensajes, resultados de cálculos, variables y cualquier otra información que quieras que sea visible para el usuario o para el desarrollador mientras se ejecuta el programa.

La sintaxis de esta es la siguiente:

```
print(valor)
```

Donde valor puede ser cualquier dato o expresión que deseas mostrar en la consola. Puede ser una cadena de caracteres (string), un número, una variable, una expresión matemática, entre otros.

Aquí hay algunos ejemplos de cómo se puede usar la función print ():

```
mensaje = "¡Hola, mundo!"
numero = 42

print(mensaje)   # Muestra el contenido de la variable mensaje
print("El número es:", numero)  # Muestra un mensaje y el valor de la variab
print(3.14)      # Muestra el valor del número directamente
```

En estos ejemplos, cada vez que se llama a la función print (), el valor proporcionado se mostrará en la consola. Esto es útil para depurar tu código, para mostrar resultados de cálculos, para interactuar con el usuario, entre otros usos.

Float:

Es un tipo de dato que se utiliza para representar números de punto flotante, es decir, números con parte decimal. Los números enteros y los números con decimales se pueden representar usando el tipo de dato float. Por ejemplo:

```
x = 3.14  # Esto es un número de punto flotante
y = 2.0   # Esto también es un número de punto flotante
```

Input:

es una función incorporada que se utiliza para solicitar información al usuario a través de la línea de comandos (consola) y obtener esa información como una cadena de caracteres (texto). La función input muestra un mensaje (prompt) en la consola y espera a que el usuario ingrese una respuesta seguida de la tecla "Enter". Luego, la función devuelve la respuesta del usuario como una cadena.

```
nombre = input("Por favor, ingresa tu nombre: ")
print("Hola,", nombre)
```

EJERCICIOS DE OPERACIONES BASICAS DE PYTHON

1. **crear un programa que, al recibir como datos dos números enteros, calcule suma, resta, multiplicación de dichos números.**

   EJEMPLO:

n1= float (input (“Ingrese el primer número”); **n2= float (input (“Ingrese el segundo número”);**	• **Suma= n1+n2** • **Resta=n1-n2** • **Multiplica= n1*n2** • **División = n1/n2**

   Ahora para imprimir estos datos debemos colocar el siguiente código:

   EJEMPLO:

• **Print (“La suma es”, (nombre de la variable que en este caso es suma);**	• **Print (“La suma es”, (nombre de la variable que en este caso es resta);**
• **Print (“La multiplicación es”, (nombre de la variable que en este caso es multiplicación);**	• **Print (“La multiplicación es”, (nombre de la variable que en este caso es multiplicación);**

VIDEOS RELACIONADOS:

https://www.youtube.com/watch?v=nKPbfIU442g

NOTAS:

Python es un lenguaje de propósito general, de alto nivel (con solo leerlo se puede entender, además de estas dos características Python es de tipado dinámico y es orientado a objetos.

Python es un lenguaje interpretado, lee línea por línea para que se puedan entender.

Ejemplos de los números enteros y números flotantes en Python son:

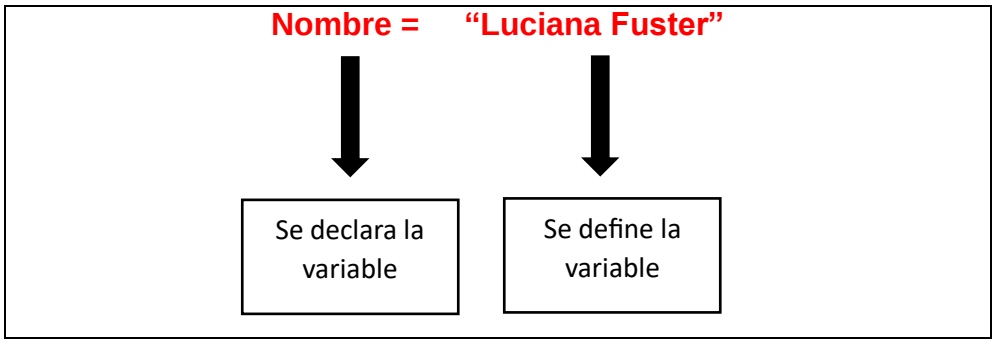
40.5 = número flotante que es **float**	**40 =** número entero que es un **int**
Ahora en Python las pablas deben de iniciar con letra mayúscula, además siguiendo con el contexto de tipo de números y demás Los números boléanos serian:	**False** y **True**

Variables:

Son espacios que se almacenan en la memoria del programa, porque cualquier dato es reutilizable **(TODO PUEDE VARIAR).**

Un ejemplo de ellos es:

nombre = “patricio parodi”; **Print= (nombre)**	**nombre = “48”;** **Print= (nombre)**



Para concatenar haciendo una operación podemos colocar el (+=) , significa que el valor que ya tiene mas lo que este depuse del igual

Ejemplo :

numero += 11

Si imprimo "numero" de esta manera) o si repito el proceso, pero cambiando la variable, obtendré el mismo resultado. Por ejemplo:

```
numero = 12
numero += 7
numero += 3
```

En la consola, el resultado será 22. Ahora, para concatenar cadenas de texto:

```
nombre = "Luciana Fuster"
bienvenido = "Hola buenas tardes " + nombre + ", cómo te va"
print(bienvenido)
```

DIFERENCIA ENTRE LISTA Y TUPLA:

Una distinción fundamental radica en que las tuplas son inmutables, lo que implica que al utilizarlas estaremos trabajando con un conjunto de datos que no podrá ser alterado una vez creado. En contraste, las listas permiten la modificación flexible de sus elementos, posibilitando diversas formas de manipulación de datos.

```
lista = ["lucas daltgo", "soy dalto", True,1.85]   #ESTO SUNA MATRIZ "QUE SON LAS LISTAS"
print(lista (0))


tupla = ("lucas daltgo", "soy dalto", True,1.85)  #LAS TUPLAS SON IGUALES A LAS LISTAS"
print(tupla (0))

```

CONJUNTO:

En un conjunto, se destacan las diferencias notables con las listas y las tuplas en cuanto a la naturaleza desordenada y su susceptibilidad a cambios (sin que estos afecten los resultados). **un ejemplo de ello :**

```
conjunto = {"soy dalto", "lucas daltgo", 1.85, True}
```

A su vez, se distingue de las listas debido a la imposibilidad de acceder mediante índices. **un ejemplo de ello :**

```
conjunto = {"soy dalto", "lucas daltgo", 1.85, True}
print(conjunto [0])
```

Una de las tantas características que tiene el conjunto, no te permite repetir valores , **un ejemplo de ello :**

```
conjunto = {"soy dalto", "lucas daltgo", 1.85, True, "soy dalto "}
print(conjunto [0])
```

DICCIONARIO:

Un diccionario es una estructura de datos en programación que permite almacenar y organizar información en forma de pares clave-valor. Cada elemento en un diccionario consiste en una clave única y un valor asociado a esa clave.

un ejemplo de ello :
la clave en este caso es ‘nombre’ y el valor serio “soy dalto”

```python
#ESTO ES UN JSON EN JAVASCRIP Y PYTHON, ADEMAS TIENE LA ESTRUCTURA DE KEY Y VALUE
diccionario = {
        'nombre ' : "soy dalto",
        'apellido ' : "loreto loreto",
        'les gusta trabajar' : True,
        'altura' : 1.65,
        'dato duplicado ' : "soy dalto"
    }
print (diccionario[2])
```

OPERADORES ARITMETICOS:

Los operadores aritméticos símbolos especiales que se utilizan para realizar operaciones matemáticas básicas entre valores numéricos. Estas operaciones incluyen sumar, restar, multiplicar, dividir y más.

```python
# suma y resta (+ y -)
suma= 12 + 5
resta= 12 - 5

# multiplicacion y división (* y /)
multiplicacion = 15 * 5
division= 15 / 5 #devuelve un dato float

#potenciacion  (exponente) (**)
exponente= 12 ** 5

#division baja (//)
division_baja= 12 // 5  #devuelve entero redondiado hacia bajo

# resto 0 módulo ( porsentajes)
resto= 12 % 5


tipo_datos= type (division_baja) #type(dato) nos devuelve que tipo de dato es
print(tipo_datos) #EN ESTA PARTE SE COLOCA EL NOMBRE DE SUMA,RESTA ETC.....

```

OPERACIONES DE COMPARACION:

Las operaciones de comparación son herramientas que se usan para evaluar y comparar relaciones entre valores, determinando si son iguales, mayores, menores o similares en algún aspecto.

Además, también hay que acotar que las operaciones de comparación nos devuelven **(True y False)**

operadores > comparacion.py > ...

```python
igual_que = 5 == 6

distinto_de = 5 != 6

mayor_que = 5 > 6

menor_que=  5 < 6

mayor_O_igual= 5 >= 6

mayor_O_igual= 5 <= 6

print(igual_que) # AQUI SE COLOCAN TODAS LAS COMPARACIONES

#CALCULOS COMBINADOS
a = 5
b = 10
c = 20

comparaciones = a + b == c
print(comparaciones)

#COMPARACIONES USUARIOS

password= "patricio parodi"
password_escrito= "patricio parodi"
print(password == password_escrito)
```

CODICIONALES:

Las condicionales son instrucciones que permiten al programa tomar decisiones basadas en ciertas condiciones. Esto permite ejecutar diferentes partes de código dependiendo de si una condición es verdadera o falsa. **un ejemplo de ello :**

codicionales > if-else.py > ...

```python
edad = 20 #ESTE  VALOR PUEDE CAMBIAR

if edad >= 19 :
    print("Podes Pasar")
else:
    print("No Podes Pasar")



password = "python"
password_escrita= "python"

if password == password_escrita:
    print("Iniciando Sesion ")
else:
    print("Password Equivocado , Ingrese Nuevamente")

```

NOTA IMPORTANTE:

Elif: es una abreviatura de "else if", y se utiliza dentro de una estructura condicional (**if**) para verificar una condición adicional si la condición principal no se cumple.

Ifs anidados: se refieren a la situación en la que tienes una estructura condicional dentro de otra estructura condicional. Cada bloque de if anidado se verifica solo si la condición del if exterior es verdadera.(es decir es un if dentro de otro if)

Ejemplo de ello:

codicionales > else-if.py > ...

```python
if ingreso_mensual > 1000:
    if ingreso_mensual - gasto_mensual < 0:
        print("Estás en deficit")
    elif ingreso_mensual - gasto_mensual >3000:
        print("Estás bien de dinero")
    else:
        print("Estas gastando en mucha ropa, hay que ver si alnza")

```

OPERADORES DE LOGICOS

Los operadores lógicos son símbolos especiales que permiten combinar y evaluar condiciones, como **"and" (y), "or" (o) y "not" (no**), para tomar decisiones basadas en el valor de verdad de expresiones booleanas.

- El operador **AND** retorna True solamente si ambas expresiones son **True**, en caso contrario, retorna **False**.

- El operador **OR** retorna True si al menos una de las expresiones es **True**, y solo retorna **False** si ambas son **False.**
- El operador **NOT** invierte el valor de verdad. Si la expresión es **True,** not **True** resultará en **False**, y si la expresión es **False**, not **False** resultará en **True.**
- En resumen, el programa realiza operaciones lógicas usando estos operadores y luego imprime el valor de resultado2, que es **False**.

operadores > logicos.py > ...

```
1   #AND
2
3   resultado1 = True  & True   #defvolver true
4   resultado2 = False & True   #devolver false
5   resultado3 = True  & False  #devolver false
6   resultado4 = False & False  #devolver false
7
8   #OR
9
10  resultado5 = True  & True   #defvolver true
11  resultado6 = False & True   #defvolver true
12  resultado7 = True  & False  #defvolver true
13  resultado8 = False & False  #devolver false
14
15
16
17   #NOT
18
19  resultado9 = not True   #defvolver false
20  resultado10 = not False   #defvolver true
21
22
23  print (resultado2)
```

Hacer ejemplos con muchas variables.