PYTHON

Instructivo #2

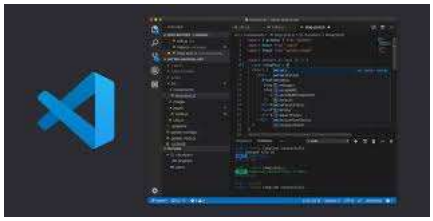
Programas secuenciales (asignación, lectura y escritura)

Objetivo: Realizar programas en Python, donde se ilustren los tipos, estructuras de datos, variables y constantes.

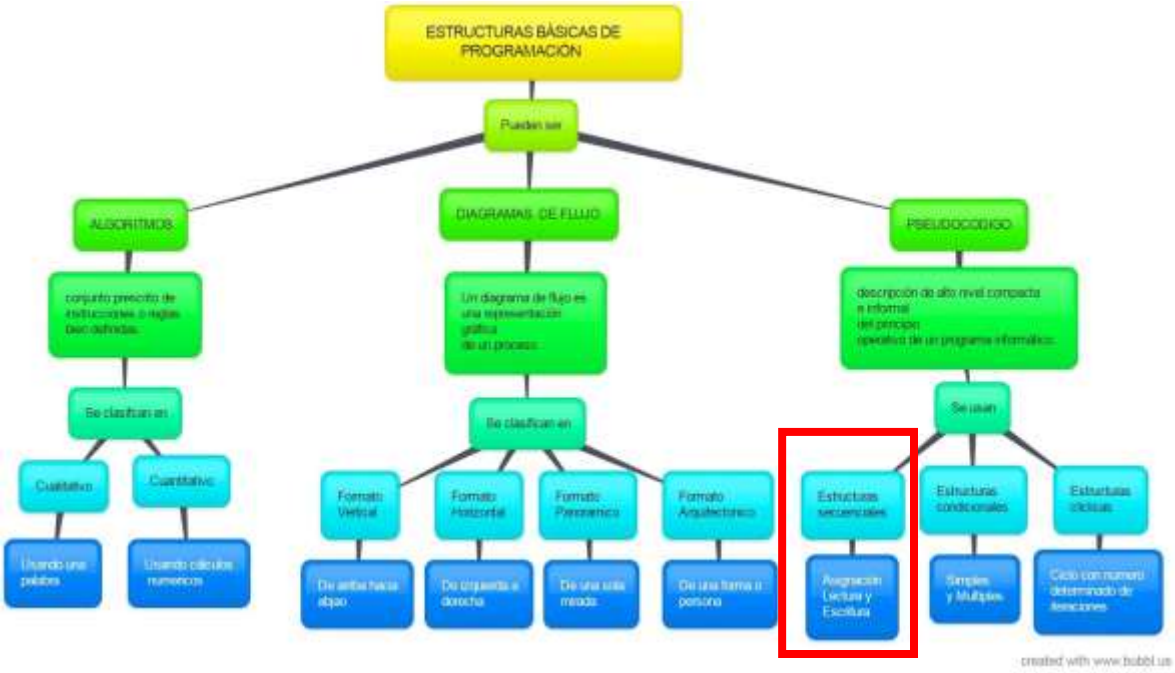
Herramienta de trabajo: Software

Python 3.9


MSDOS


Editor Visual Studio Code (Terminal)

Ing. Carolina Zamaida González P.
SENA


ESTRUCTURAS BÀSICAS DE PROGRAMACION
Pueden ser
ALGORITMOS
DIAGRAMAS DE FLUJO
PSEUDOCODIGO
conjunto preciso de instrucciones o reglas bien definidas
Un diagrama de flujo es una representacion gráfica de un proceso
descripción de alto nivel compacta e informal del principio operativo de un programa informático
Se clasifican en
Se clasifican en
Se usan
Cualitativo
Cuantitativo
Formato Vertical
Formato Horizontal
Formato Panorámico
Formato Arquitectónico
Estructuras secuenciales
Estructuras condicionales
Estructuras cíclicas
Usando una palabra
Usando cálculos numéricos
De arriba hacia abajo
De izquierda a derecha
De una sola manera
De una forma o persona
Asignación Lectura y Escritura
Simples y Multiples
Ciclo con numero determinado de iteraciones
created with www.bubbl.us

Tipos de datos en Python

Dentro de los tipos básicos o primitivos, encontramos:

Tipos numéricos: **int, float**

```
>>> type(42)
Salida
<class 'int'>
```

```
>>> type(-139)
<class 'int'>
```

```
>>> type(1.25)
Salida
<class 'float'>
```

```
>>> type(-3.0)
<class 'float'>
```

Tipos de Texto: **str** representa secuencias o cadenas (string) de caracteres.

```
>>> type("Hola mundo")
Salida
<class 'str'>
```

```
>>> type(4)
Salida
<class 'int'>
```

```
>>> type("4")
<class 'str'>
```

Tipo lógicos (booleanos): **bool** representa valores booleanos o de lógica binaria: **Verdadero o Falso.**

```
>>> type(True)
```

Salida

```
<class 'bool'>
```

```
>>> type(False)
```

Salida

```
<class 'bool'>
```

Mensajes de error: (“False”) ; (false)

```
>>> type(false)
```

Salida

```
NameError: name 'false' is not defined
```

Ejercicio 2:

ejercicio2.py

```python
# Comentario : Ejercicio 2 -
# Tipo de Datos:int, float, str (cadena), booleano

type(42)
type(-139)
type(1.25)
type(-3.0)
type("Hola Mundo")
type(4)
type("4")
type(True)
type(False)


# Comentario : Se requiere imprimir los anteriores tipos de datos...
print(type(2))
print(type(-139))
print(type(1.25))

```

Ejercicio 3: Tipo de Datos

Realizar un programa donde se visualice un String (cadena) utilizando las funciones **print, type,**. Resultados observados desde la Terminal del editor Visual Studio Code.

Funcion print()

```
Python2021 > ejercicio3.py
1   # Comentario : Ejercicio 3 - Tipo de Datos: String (cadena)
2
3   # Comillas dobles: codigo ascii : alt 34
4   print("Hola mundo: comillas doble")
5   print("""Hola mundo: triple comillas """)
6
7
8   # Comillas simples: codigo ascii : alt 39
9   print('Hola mundo: comillas simple')
10  print('''Hola mundo: tripe comillas simple''')
11
```

Terminal

Funcion Type

```
12  # Validar el tipo de datos : funcion type
13  print(type("Hola mundo"))
14
```

Terminal

```
Hola mundo: comillas doble
Hola mundo: triple comillas
Hola mundo: comillas simple
Hola mundo: tripe comillas simple
<class 'str'>
```

Concatenar texto (cadena)

```
15   # Concatenar texto
16   print("SENA" + " " + "Servicio Nacional de Aprendizaje")
17   print("Nombre del aprendiz" + " " + "Ficha No. " + " " + "ADSI")
18
```

Terminal

```
PROBLEMS   OUTPUT   CONSOLA DE DEPURACIÓN   TERMINA
PS C:\0_SENA_Armenia_2021\3_Guias_2021\1_Registr
Hola mundo: comillas doble
Hola mundo: triple comillas
Hola mundo: comillas simple
Hola mundo: tripe comillas simple
<class 'str'>
SENA Servicio Nacional de Aprendizaje
Nombre del aprendiz Ficha No.  ADSI
PS C:\0_SENA_Armenia_2021\3_Guias_2021\1_Registr
```

Tipo de dato: **int** , representa números enteros

```
# Tipo de dato: Intero
print("Numero entero:")
print (30)
```

Tipo de dato: **float** . representa números con punto decimal.

```
# Tipo de dato: Flotante
print("Numero flotante:")
print (45.36)
```

Tipo de dato: Lógico (Boolean)

```
27  # Tipo de Dato: Logico (Boolean)
28  print(type(True))
29  print(type(False))
30
```

Tipo de dato: **LISTAS**

```
# Tipo de dato: Listas
[10, 20, 30, 40, 50]
["Lunes", "Martes", "Miercoles", "Jueves]
[10, "Lunes", True, 10.8]
```

```
# Para imprimir la lista se requiere crear una variable y llamar la funcion print()
lista1= [10, 20, 30, 40, 50]
print(lista1)

lista2= ["Lunes", "Martes", "Miercoles", "Jueves"]
print(lista2)

lista3= [10, "Lunes", True, 10.8]
print(lista3)
```

TUPLAS: Son registro de dato ubicados en una fila

```
#Tuples: Son registro de dato ubicados en una fila
(10, 20, 30, 55)
()
```

```
# Para imprimir las TUPLAS se requiere crear una variable y llamar la funcion print()
#Tuples: Son registro de dato ubicados en una fila
tupla1= (10, 20, 30, 55)
print(tupla1)

tupla2= ()
print(tupla2)
```

Directorio: Está compuesto por archivos.

Estructura: **clave: valor**

Nota: Tener presente la **identacion** (tabulación) de las líneas de código.

Indentación: Es la forma que usa Python para agrupar declaraciones. En el intérprete interactivo debes teclear un tabulador o espacio(s) para cada línea indentada.

Ejercicio 4: Calculadora Python

Calculando valores, operadores y expresiones bajo en el lenguaje de Python.

Figure 1. Operadores y expresiones

Operadores de datos en Python:

- Aritméticos: `+ - * / // %`
- Comparación: `< <= > >= == !=`
- Lógicos: `not and or`
- Texto: `+ *`

- Reglas para evaluar expresiones que usan múltiples operadores

Conversión de tipos en Python:

- Operadores requieren valores del mismo tipo de datos

- Convertir entre tipos de datos

- `int()`, `float()`, `bool()`, `str()`

1. Operadores aritméticos

En cuanto a los operadores aritméticos, estos permiten realizar las diferentes operaciones aritméticas del álgebra: suma, resta, producto, división, … Estos operadores Python son de los más utilizados. El listado completo es el siguiente:

Operador	Descripción
+	Suma dos operandos.
–	Resta al operando de la izquierda el valor del operando de la derecha. Utilizado sobre un único operando, le cambia el signo.
*	Producto/Multiplicación de dos operandos.
/	Divide el operando de la izquierda por el de la derecha (el resultado siempre es un float).
%	Operador módulo. Obtiene el resto de dividir el operando de la izquierda por el de la derecha.
//	Obtiene el cociente entero de dividir el operando de la izquierda por el de la derecha.
**	Potencia. El resultado es el operando de la izquierda elevado a la potencia del operando de la derecha.

El intérprete puede utilizarse como una simple calculadora; puedes introducir una expresión y este escribirá los valores.

Precedencia y asociatividad:

```
>>> (3+5//4-2)-2**4+3*(7-2)
```

Salida

```
1
```

Dentro de cada paréntesis se evalúa:

Operador	Preced.	Asociatividad	Ejemplo	Resultado
**	1	Derecha a izquierda	2**3**2	512
+, - (unarios)	2		-2**2	-4
*, /, //, %	3	Izquierda a derecha	15/3*2	10
+, - (binarios)	4	Izquierda a derecha	3-4+5	4

2. Operadores de comparación:

Los operadores de comparación se utilizan, como su nombre indica, para comparar dos o más valores. El resultado de estos operadores siempre es True o False.

Operador	Descripción
>	**Mayor que.** True si el operando de la izquierda es estrictamente mayor que el de la derecha; False en caso contrario.
>=	**Mayor o igual que.** True si el operando de la izquierda es mayor o igual que el de la derecha; False en caso contrario.
<	**Menor que.** True si el operando de la izquierda es estrictamente menor que el de la derecha; False en caso contrario.
<=	**Menor o igual que.** True si el operando de la izquierda es menor o igual que el de la derecha; False en caso contrario.
==	**Igual.** True si el operando de la izquierda es igual que el de la derecha; False en caso contrario.
!=	**Distinto.** True si los operando son distintos; False en caso contrario.

- Se aplican a **int** o **float**

  `<` `<=` `>` `>=` `!=` `==`

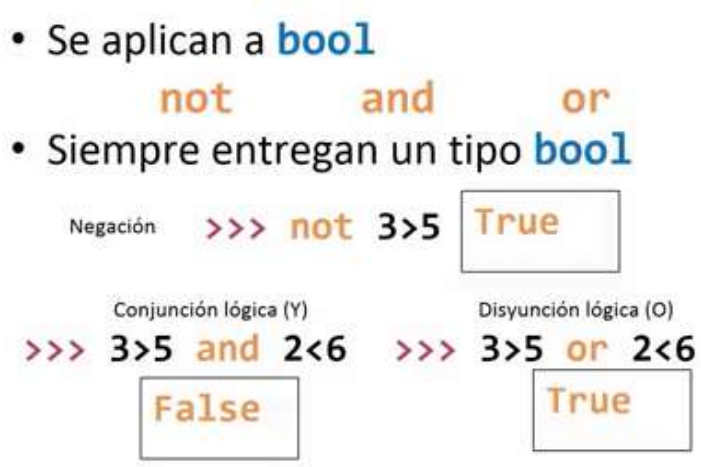
- Siempre entregan un tipo **bool**

Menor	Mayor o igual	Distinto	Igualdad
`>>> 5<5.1`	`>>> 3>=5`	`>>> 3!=5`	`>>> 6==9`
Salida			
`True`	`False`	`True`	`False`

3. Operadores lógicos o boleanos:

A la hora de operar con valores booleanos, tenemos a nuestra disposición los operadores and, or y not.

Operación	Resultado	Descripción
a **or** b	Si a se evalúa a falso, entonces devuelve b, si no devuelve a	Solo se evalúa el segundo operando si el primero es falso
a **and** b	Si a se evalúa a falso, entonces devuelve a, si no devuelve b	Solo se evalúa el segundo operando si el primero es verdadero
not a	Si a se evalúa a falso, entonces devuelve True, si no devuelve False	Tiene menos prioridad que otros operadores no booleanos

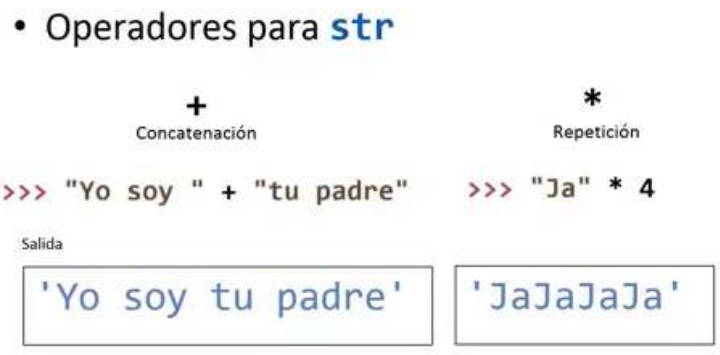


4. Procedencias actualizadas:

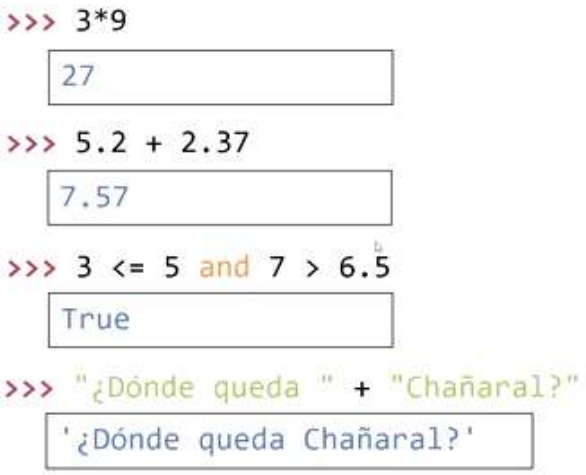
```
>>> 5//4 > 3 or 2<5**2
True
```

Operador	Asociatividad	Ejemplo	Resultado
**	Derecha a izquierda	2**3**2	512
+, - (unarios)		-2**2	-4
*, /, //, %	Izquierda a derecha	15/3*2	10
+, - (binarios)	Izquierda a derecha	3-4+5	4
<. <=, >, >=, !=, ==	Izquierda a derecha	3<4<=4<5	True
not		not not 5>2	True
and	Izquierda a derecha	not True and False	False
or	Izquierda a derecha	True or True and False	True

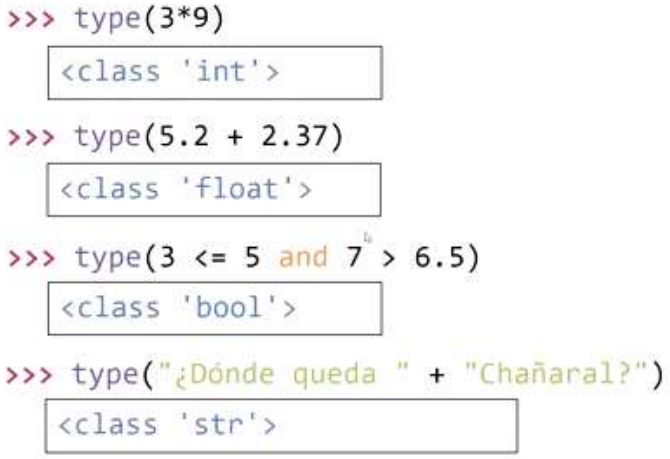
5. **Operadores para tipos de texto:** Sumando y multiplicando str



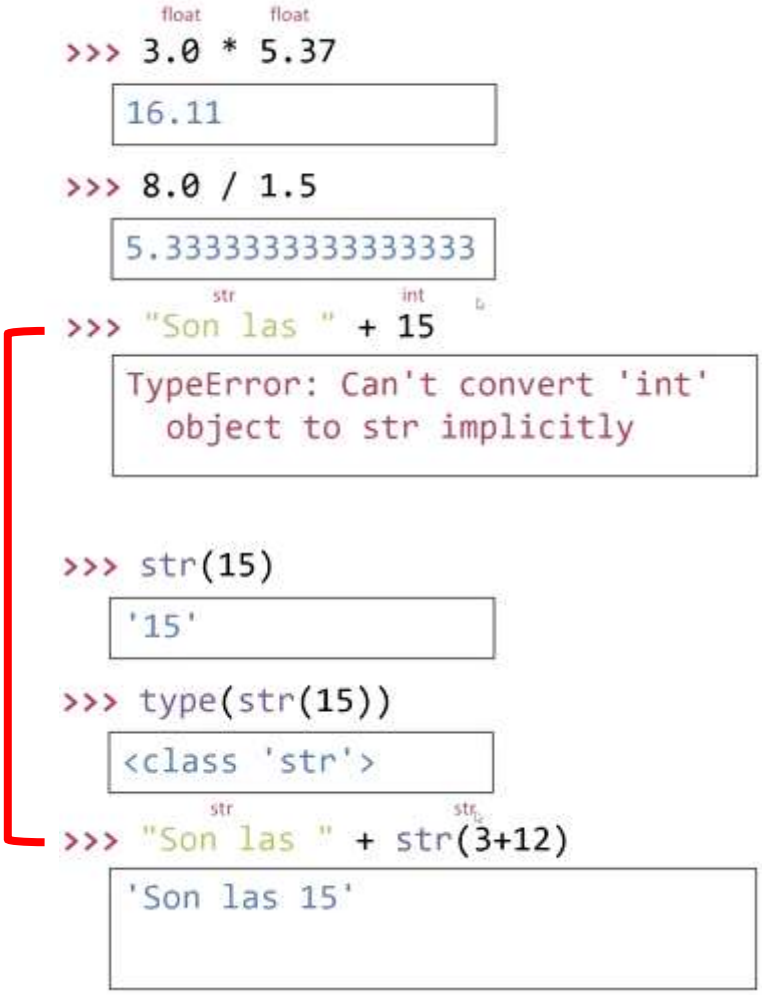
6. Operando con valores: mismo tipo



7. Operando con valores: Tipo de expresiones

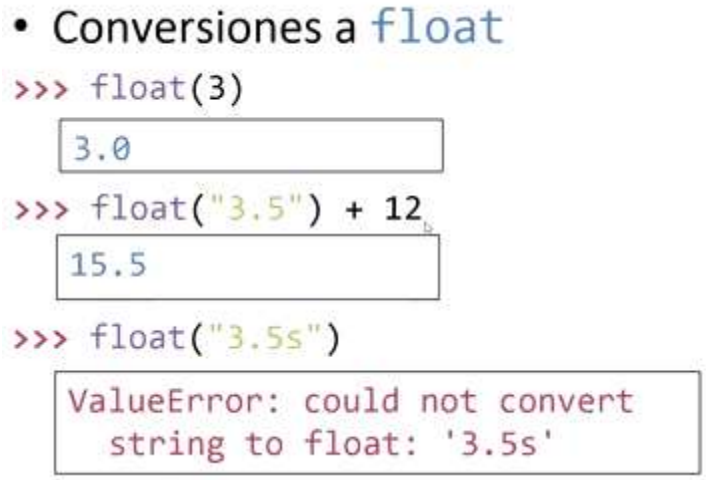


8. Operando con valores: Valores de distinto tipo

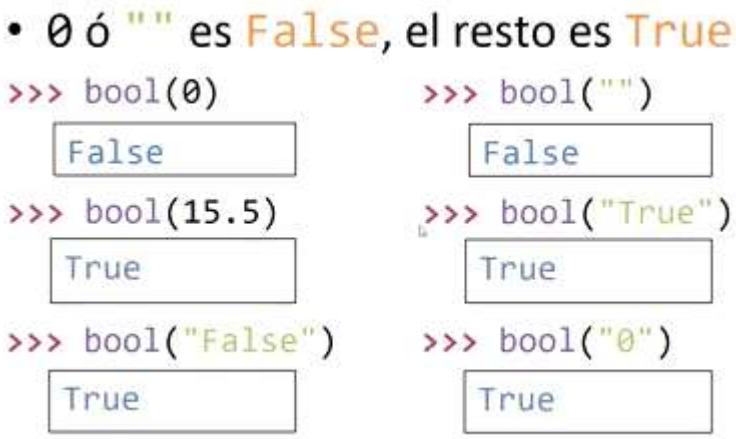


9. Conversión de tipo: int ()

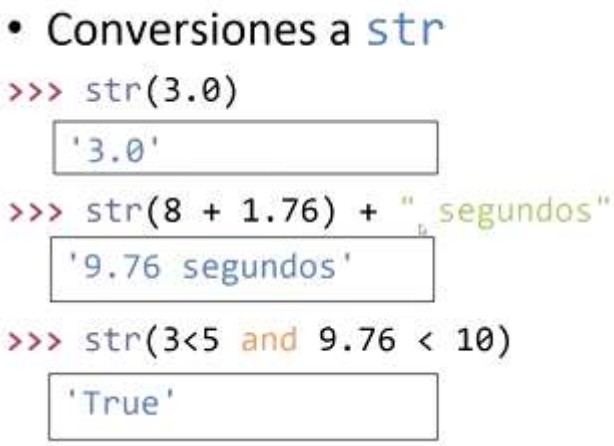
10. Conversión de tipos: float ()



11. Conversión de tipos: bool ()



12. Conversiones de tipos: str()



EDITOR VISUAL STUDIO CODE

Realizar las siguientes operaciones matemáticas, manejo de cadenas con el editor de Visual Studio Code.

Ejercicio4: Realizar las siguientes operaciones matemáticas y manejo de cadenas

NUMEROS

1. Operaciones matemáticas

```python
# Ejercicio 4:

# NUMEROS --------------------------------

print("Operaciones Matematicas:")
print()  # una nueva linea
print("Suma, resta, multiplicacion y division:")
print (2 + 2)
print (50 - 5*6)
print ((50 - 5*6) / 4)
print (8 / 5)

```

Terminal

```
PS C:\0_SENA_Armenia_2021\3_Guias_2021\1_Registros_Guias_2021\GUIAS_2021_Armenia\1_Introd_a la Prog con Phyton\Python2021> python ejercicio4.py
Operaciones basicas:

Suma, resta, multiplicacion y division:
4
20
5.0
1.6
```

Nota: Utilizar la función print() cada vez que finalice un tema.

```python
print()  # una nueva linea
```

Ing. Carolina Zamaida González P.
SENA

2. La **división** (/) siempre retorna un punto flotante. Para hacer floor division y obtener un resultado entero (descartando cualquier resultado fraccionario) puede usarse el operador //; para calcular el resto puedes usar %:

```
# Divisiones:

print("Division:")
print (17 / 3)  # Division: Decimales(float)
print (17 // 3)  # Division: Fraccionario
print (17 % 3)  # Residuo de la division
print (5 * 3 + 2)  # Operacion de validacion
```

3. Potencias: Se utiliza el operador **

```
# Exponencial:

print("Exponencial:")
print (5 ** 2)
print (2 ** 7)
```

4. El signo igual (=) se usa para asignar un valor a una variable.

```
# Asignación de variables (igual):

print("Asignacion de variables = ")
A = 20  #Variable A
B = 5 * 9  #Variable B

print(A)
print(B)
```

CADENAS

5. Las cadenas se pueden concatenar (pegar juntas) con el operador `+` y se pueden repetir con `*`:

```
# CADENAS --------------------------------

# Repeticion de Cadenas
print("Repeticion de Cadenas:")
print(10 * 'Aprendiz')
print(3 * 'Aprendiz' + 'SENA')
```

6. Las cadenas de texto se pueden *indexar* (subíndices), el primer carácter de la cadena tiene el índice 0. No hay un tipo de dato diferente para los caracteres; un carácter es simplemente una cadena de longitud uno:

```
# Repeticion de Indices
print("Indices:")
word = 'SENA'
print(word[0])  # character in position 0
print(word[1])  # character in position 1
print(word[2])  # character in position 2
print(word[3])  # character in position 3
```

7. Los índices quizás sean números negativos, para empezar a contar desde la derecha:

```
# Repeticion de Indices (-)
print("Indices - :")
word = 'SENA'
print(word[0])  # character in position 0
print(word[-1])  # character in position 1
print(word[-2])  # character in position 2
print(word[-3])  # character in position 3
```

8. Sustracción de caracteres: obtener partes de las cadenas de texto

```
 +---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

```
# Sustracion de caracteres
print("Sustraccion de caracteres:")
word = 'Python'
print (word[0:2])  # characters from position 0 (included) to 2 (excluded)
print (word[2:6]) # characters from position 2 (included) to 5 (excluded)
```

9. La función incorporada `len()` retorna la longitud de una cadena:

```
# Longitud
print("Cantidad de caracteres:")
cadena = 'Servicio Nacional de Aprendizajes'
len(cadena)

print(cadena)
print(len(cadena))

print()  # una nueva linea
```

10. Imprimir el valor de una variable

```
# Variable
print("Valor variable:")
i = 256*256
print('El valor de la variable i, es:', i)
```

Primeros pasos hacia la programación

¿Por qué es tan famosa la secuencia FIBONACCI?

Articulo: https://www.muyinteresante.com.mx/preguntas-y-respuestas/secuencia-fibonacci-matematicas-aplicadas/

Ing. Carolina Zamaida González P.
SENA

Realizar una comparación entre los ejercicios 5 y 6. Serie de Finonacci y **socializar su punto de vista con el grupo.**

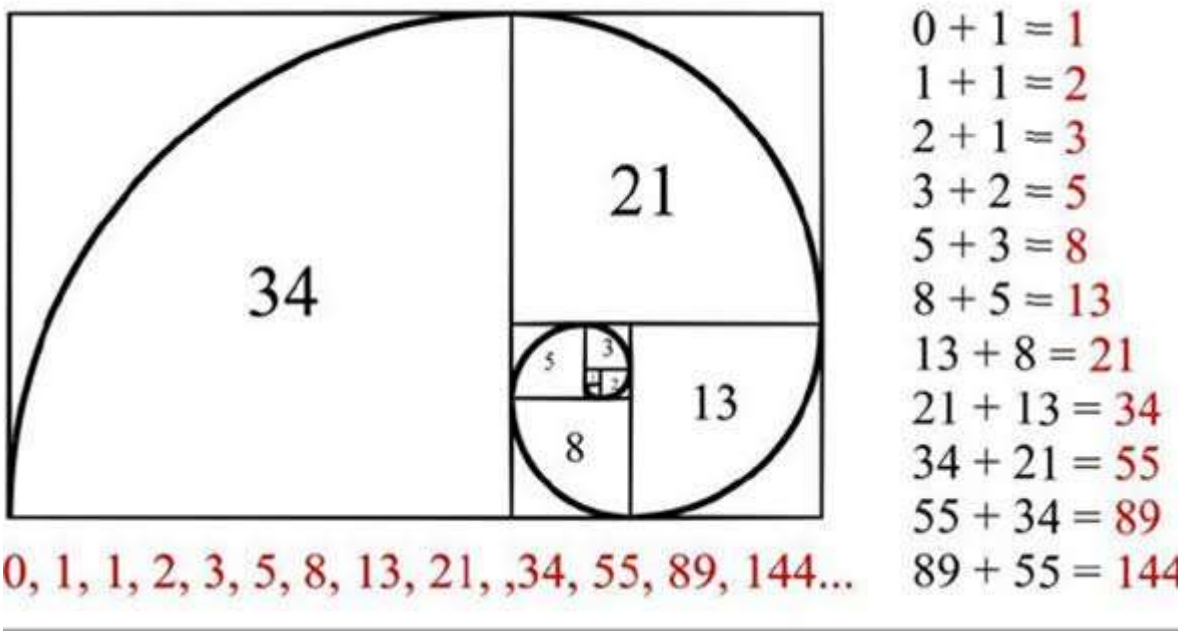
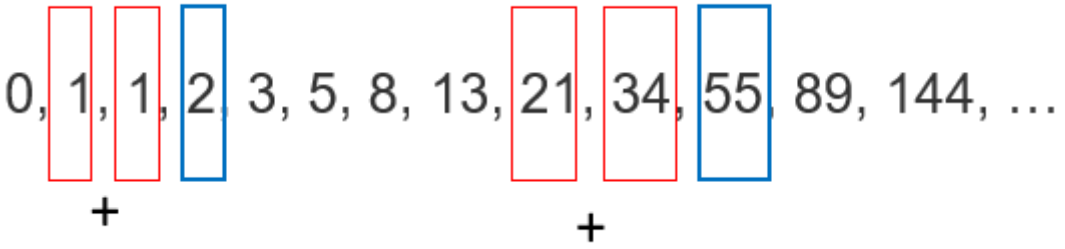


Figure 2. Representación gráfica de la sucesión de Fibonacci

En matemáticas, la sucesión o serie de Fibonacci hace referencia a la secuencia ordenada de números descrita por Leonardo de Pisa, matemático italiano del siglo XIII:



A cada uno de los elementos de la serie se le conoce con el nombre de número de Fibonacci.

Ejercicio 5. Serie de Fibonacci

Python2021 > ejercicio5.py > ...

```
1   # Serie de Fibonacci
2   # La suma de dos elementos define el siguiente
3
4   # Definicion de variable
5   a=0
6   b=1
7   c=0
8
9   print(a)
10  print(b)
11
12  #Ciclo: Mientras
13  while c < 10:
14      c=a+b
15      print(c)
16      a=b
17      b=c
```

Ejercicio 6. Serie de Fibonacci bajo la estructura que ofrece Python

ejercicio4.py | ejercicio5.py | ejercicio6.py

Python2021 > ejercicio6.py > ...

```
1   # Serie de Fibonacci
2   # La suma de dos elementos define el siguiente
3
4   # Definicion de variable
5   a, b = 0, 1
6
7   #Ciclo: Mientras
8   while a < 10:
9       print(a)
10      a, b = b, a+b
```

Ing. Carolina Zamaida González P.
SENA

Este ejemplo introduce varias características nuevas.

- La primera línea contiene una ***asignación múltiple***: las variables `a` y `b` obtienen simultáneamente los nuevos valores 0 y 1. En la última línea esto se usa nuevamente, demostrando que las expresiones de la derecha son evaluadas primero antes de que se realice cualquiera de las asignaciones. Las expresiones del lado derecho se evalúan de izquierda a derecha.
- El bucle `while` se ejecuta mientras la condición (aquí: `a < 10`) sea verdadera. En Python, como en C, cualquier valor entero que no sea cero es verdadero; cero es falso. La condición también puede ser una cadena de texto o una lista, de hecho, cualquier secuencia; cualquier cosa con una longitud distinta de cero es verdadera, las secuencias vacías son falsas. La prueba utilizada en el ejemplo es una comparación simple. Los operadores de comparación estándar se escriben igual que en C: `<` (menor que), `>` (mayor que), `==` (igual a), `<=` (menor que o igual a), `>=` (mayor que o igual a) y `!=` (distinto a).

- **El cuerpo del bucle está *indentado***: la indentación es la forma que usa Python para agrupar declaraciones. En el intérprete interactivo debes teclear un tabulador o espacio(s) para cada línea indentada.

  En la práctica vas a preparar entradas más complicadas para Python con un editor de texto; todos los editores de texto modernos tienen la facilidad de agregar la indentación automáticamente. Cuando se ingresa una instrucción compuesta de forma interactiva, se debe finalizar con una línea en blanco para indicar que está completa (ya que el analizador no puede adivinar cuando tecleaste la última línea). Nota que cada línea de un bloque básico debe estar sangrada de la misma forma.

FUNCION input()

La función input() permite obtener texto escrito por teclado. Al llegar a la función, el programa se detiene esperando que se escriba algo y se pulse la tecla Intro.

Ejercicio 7. Realizar una entrada de datos y guardar en un variable etiquetada "**nombre**".

```
ejercicio7.py > ...
1    print("¿Cómo se llama?")
2    nombre = input()
3    print(f"Me alegro de conocerlo(a), {nombre}")
4
```

f (f-strings)**:** A partir de la versión 3.6 de Python, se introdujeron las f-strings o literals strings, para resolver un poco estos problemas de legibilidad. Consultar: https://platzi.com/blog/f-strings-en-python/

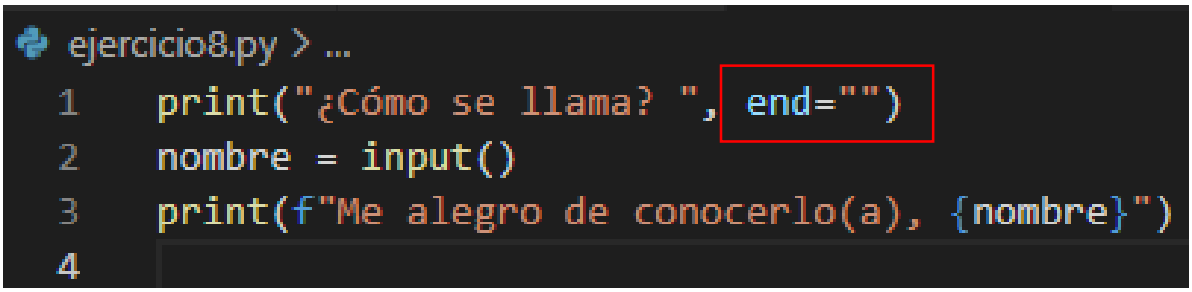
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
PS C:\0_SENA_Armenia_2021\3_Guias_2021\1_Registros_
¿Cómo se llama?
CAROLINA ZAMAIDA
Me alegro de conocerlo(a), CAROLINA ZAMAIDA
```

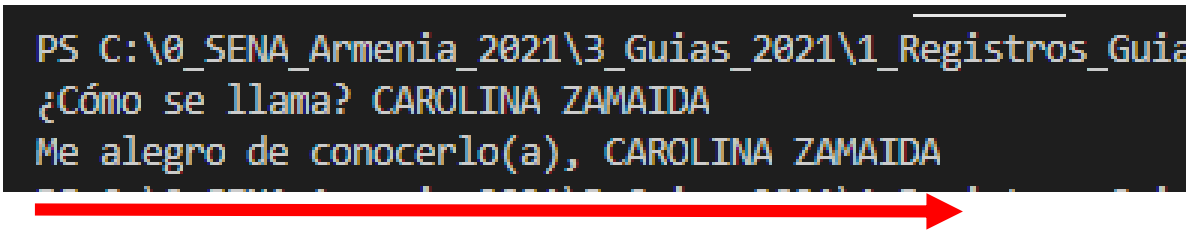
Ejercicio 8.

En el ejercicio anterior, el usuario escribe su respuesta en una línea distinta a la pregunta porque Python añade un salto de línea al final de cada print().

Si requiere que el usuario escriba su respuesta a continuación de la pregunta, se podría utilizar el argumento opcional **end** en la función print(), que indica el carácter o caracteres a utilizar en vez del salto de línea.

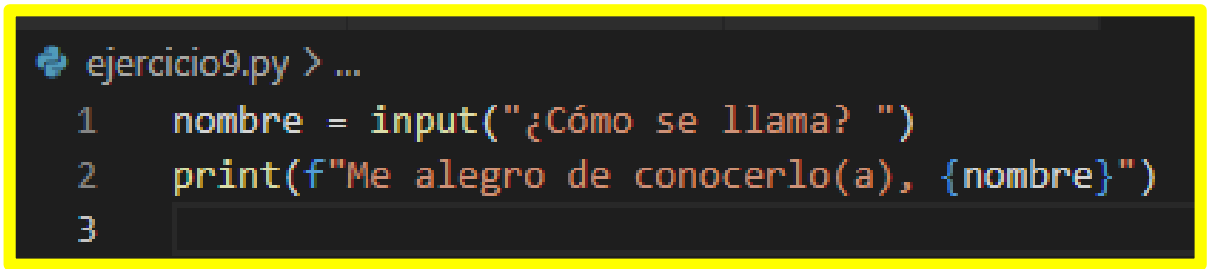
Para separar la respuesta de la pregunta se ha añadido un espacio al final de la pregunta.





Ejercicio 9.

Otra solución, más compacta, es aprovechar que a la función input() se le puede enviar un argumento que se escribe en la pantalla (sin añadir un salto de línea):



Variables como argumento de la función input()

Ejercicio 10.

La función input() sólo puede tener un argumento. En versiones de Python anteriores a la versión 3.6 esto causaba problemas cuando se querían incorporar variables en el argumento de la función input().

```
nombre = input("Digitar su nombre: ")
apellido = input(f"Digitar su apellido, {nombre}: ")
print(f"Me alegro de conocerlo(a), {nombre} {apellido}.")
```

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
PS C:\0_SENA_Armenia_2021\3_Guias_2021\1_Registros_G
Dígame su nombre: CAROLINA
Dígame su apellido, CAROLINA: GONZALEZ
Me alegro de conocerle, CAROLINA GONZALEZ.
```

Ejercicio 11.

ejercicio11.py > ...

```
1
2   numero1 = int(input("Dígame un número: "))
3   numero2 = int(input(f"Dígame un número mayor que {numero1}: "))
4   print(f"La diferencia entre ellos es {numero2 - numero1}.")
5
```

Resultados.

```
Dígame un número: 56
Dígame un número mayor que 56: 60
La diferencia entre ellos es 4.
```

Ejercicios 12 hasta 30 . Programas Secuenciales

12 Lea un número y escriba su cuadrado

13 Determinar el **área** y **volumen** de un cilindro. Referente a sus dimensiones radio y altura se lee desde teclado (usuario).

14 Se desea convertir de metros a centímetros, dados los primeros.

15 Intercambiar los valores de dos variables numéricas.

16 Realice un programa que sume dos números e imprima el resultado.

17 Realice un programa que imprima la tabla de multiplicar de un número n (sin utilizar ciclos de repetición).

18 Lea un número y escriba su raíz cuadrada.

Consultar: Módulos Matemáticos en Python: Math y Cmath.
https://code.tutsplus.com/es/tutorials/mathematical-modules-in-python-math-and-cmath--cms-26913

19 Determinar el área y perímetro de un rectángulo cuyas dimensiones se leen desde teclado.

20 Realizar la conversión de pies a pulgadas, solicitando al usuario pies.

21 Se desea convertir de metros a milímetros, solicitando al usuario los metros.

22 Informar en pantalla el promedio de cinco números.

23 Conociendo el valor de la entrada al cine, calcule el monto a pagar para una delegación de personas. Recuerde que deberá hacer un descuento del 3% por cantidad.

24 Ingresando la cantidad de un producto y su precio unitario, obtener como resultado el precio final que debe pagar el cliente.

25 Suponga que un individuo desea invertir su capital en un banco y desea saber cuánto dinero ganará después de un mes si el banco paga a razón de 2% mensual.

26 Realizar una aplicación que calcule las cuatro operaciones básicas, suma, resta, división, multiplicación, con dos números.

27 Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

Ing. Carolina Zamaida González P.
SENA

28 Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuánto deberá pagar finalmente por su compra.

29 Un alumno desea saber cuál será su calificación final en la materia de español, donde sus calificaciones están relacionadas con los siguientes porcentajes:

- ✓ 50% Promedio de sus (3) tres calificaciones parciales.
- ✓ 30% Examen final.
- ✓ 20% Un trabajo final.

Ejercicios 30. Programas Secuenciales

30. **Salario de un Trabajador:** Se desea obtener el salario neto de un trabajador conociendo el número de horas trabajadas, el salario hora y la tasa de impuestos que se le debe deducir.

Material de referencia:

- Video, Cristian Ruz, Académico Facultad de Educación. Pontificia Universidad Católica de Chile
- Operadores Python, https://j2logo.com/python/tutorial/operadores-en-python/#operadores-comparacion
- Tutorial Python, https://docs.python.org/es/3/tutorial/introduction.html#numbers
- Qué son las f-strings en Python, https://platzi.com/blog/f-strings-en-python/

Ing. Carolina Zamaida González P.
SENA