08 de mayo de 2024

INGENIERIA EN DESARROLLO DE SOFTWARE

Materia: FUNDAMENTOS DE PROGRAMACION II

1.2 Actividad. Tipos, variables, bloques y operadores

Elaborado por: Jesús Angel Hernández Martínez

Lenguaje de Programación Python

Concepto	Definición / Características	Aplicación
Python	Es un lenguaje de programación de alto nivel, interpretado y multipropósito. (Fernandez Montoro, 2013). • Los programas son compactos. • Legible y más sencilla su sintaxis. • Ofrece entorno interactivo. • Detecta errores de programación. • Puede ser procedimental u orientado a objetos. • Posee estructuras de datos de fácil manipulación. (Marzal Varo, Garcia Sevilla, & Gracia Luengo, 2016)	• Ciencia de datos. • Aprendizaje automático (machine learning). • Desarrollo web. • Enseñanza de computación y programación. • Visión por computadora y procesamiento de imágenes. • Desarrollo de videojuegos. • Medicina y farmacología. • Biología y bioinformática. • Neurociencia y psicología. • Astronomía. • Otras áreas tales como robótica, vehículos autónomos, negocios, meteorología y desarrollo de interfaz gráfica de usuario. (Cassingena Navone, 2022)
Variables	Existe el tipado estático y el dinámico, Python utiliza el tipado dinámico, esto quiere decir que una variable puede cambiar su tipo en tiempo de ejecución, es decir durante el desarrollo del programa. (Fernandez Montoro, 2013).	Podemos asignar en el mismo bloque de código, diferentes tipos de datos a la misma variable.
Tipos de datos	• Números (enteros, float, complejos) • Strings (texto) • Boolean (true, false) • Colecciones(Tuplas, Listas, Diccionarios)	Desarrollo de programas en Python
Bloques	Los programas se construyen a partir de bloques de código, un block es una parte del texto del programa que se ejecuta como una unidad.	Módulos, Funciones y clases

Operadores Aritméticos			
Tipo	**Operación**	**Operador**	**Aplicación**
Aritméticos	Suma	+	Sumar
	Resta	-	Restar
	Multiplicación	*	Multiplicar
	División con decimales	/	División real
	División entera	//	Dividir con resultado entero
	Modulo	%	Modulo o residuo
	Potencia o exponente	**	Elevar un valor a una exponente

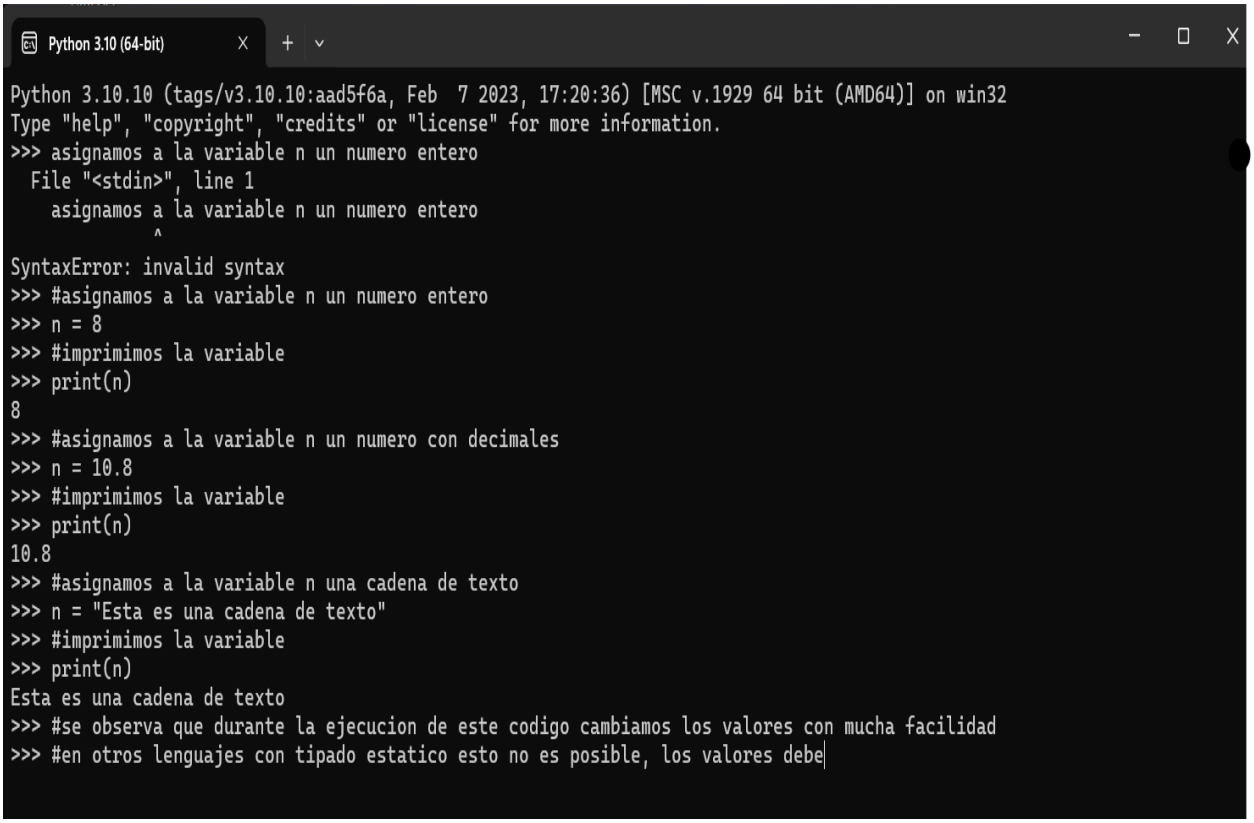
Operadores de Asignación			
Operador	**Definición**	**Uso**	**Resultado ejemplo**
=	Asignación	a = 5, asigna el valor de 5 a la variable a	5
+=	Agrega valor a un acumulado	a+=5, suma 5 al valor previo de la variable	10
-=	Disminuye valor a un acumulado	a-=3, resta 3 al valor previo de la variable	7
***=**	Multiplica el valor acumulado por un valor dado	a*=2, multiplica el valor acumulado de a por 2.	14
****=**	Eleva el valor acumulado a una potencia	a**=2, eleva al exponente 2 el valor acumulado en la variable	196
/=	División real del acumulado entre un valor dado	a/= 14, divide el valor acumulado entre 14	14.0
//=	División entera del acumulado entre un valor dado	a//=2, divide el valor acumulado entre 2	7.0
%=	Retorna el residuo de la división del acumulado entre un valor dado.	a%=2	1.0

Operadores Relacionales		
Operador	**Significado**	**Uso**
==	Igual a	Validar si x es igual a y
>	Mayor a	Validar si x es mayor que y
<	Menor a	Validar si x es menor que y
>=	Mayor o igual a	Validar si x es igual o mayor que y
<=	Menor o igual a	Validar si x es menor o igual que y
¡=	Diferente a	Validar si x es diferente a y

Operadores Lógicos	Significado	Uso
not	Negación	Validar que **x** es distinto a **y** ¿x **not** y?
and	Y	Validar que dos o más condiciones son verdaderas ¿x == y **and** x == z?
or	O	Validar que al menos una condición es verdadera ¿x == y **or** x ==z?

Ejemplos

- **Variables en Python**



- **Tipos de Datos**

In [12]:
```
#Ejemplo de numero entero (int)
n = 4
type(n)
```

Out[12]: int

In [4]:
```
#Ejemplo de numero flotante o real (float)
n = 4.5
type(n)
```

Out[4]: float

In [5]:
```
#Ejemplo de numero complejo
n = complex(5,6)
print(n)
type(n)
```

```
(5+6j)
```

Out[5]: complex

In [6]:
```
#Ejemplo de tipo string
n = "Este es un ejemplo de string"
type(n)
```

Out[6]: str

In [7]:
```
#Ejemplo de tipo boolean
n = True
type(n)
```

Out[7]: bool

In [8]:
```
#Ejemplo de tupla
n = (5,3)
type(n)
```

Out[8]: tuple

In [9]:
```
#Ejemplo de lista
n = [5,"a",True]
print(n)
type(n)
```

```
[5, 'a', True]
```

Out[9]: list

In [11]:
```
#Ejemplo de diccionario
calificaciones = {
    'Jesus' : [10,10,10],
    'Angel': (10,9,8),
    'Godinez': "Sin calificacion"
}
print(calificaciones)
type(calificaciones)
```

```
{'Jesus': [10, 10, 10], 'Angel': (10, 9, 8), 'Godinez': 'Sin calificacion'}
```

Out[11]: dict

- **Bloques**

Clases

```
In [1]: class Employee:
          new_id =1
          def __init__(self):
            self.id = Employee.new_id
            Employee.new_id+=1
          def say_id(self):
            print(f"My id is {self.id}")

        e1 = Employee()
        e1.say_id()
```

```
My id is 1
```

Funciones

```
In [2]: def print_greather(a,b):
            if(a > b):
                print(a,' is greather than ',b)
            else:
                print(b,' is greather than ',a)


        print_greather(5,8)
```

```
8  is greather than  5
```

Módulos

Ejemplo del modulo sqlite3, un modulo puedo constar de distintas clases, métodos, contenedores, funciones, variables.

```
>>> print(dir('sqlite3'))
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnew
args__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__red
uce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center',
 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifie
r', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'r
emovesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 't
ranslate', 'upper', 'zfill']
>>>
```

- **Operadores**

```
In [3]: #Todos los operadores incluidos
        n = 5
        print(n)
```
```
5
```

```
In [4]: n+=5
        print(n)
```
```
10
```

```
In [5]: n-=3
        print(n)
```
```
7
```

```
In [6]: n*=2
        print(n)
```
```
14
```

```
In [7]: n**=2
        print(n)
```
```
196
```

```
In [8]: n/=14
        print(n)
```
```
14.0
```

```
In [9]: n//=2
        print(n)
```
```
7.0
```

```
In [10]: n%=2
         print(n)
```
```
1.0
```

```
In [11]: r = 10
         print(n==r)
```
```
False
```

```
In [12]: print(n>r)
```
```
False
```

```
In [13]: print(n<r)
```
```
True
```

```
In [14]: print(n>=r)
```
```
False
```

```
In [15]: print(n<=r)
```
```
True
```

```
In [16]: print(n==r)
```
```
True
```

```
In [24]: n_es_mayor = n == r
         print(n_es_mayor)
         print(not n_es_mayor) # no es verdad que n es mayor que r?
```
```
False
True
```

```
In [25]: n = 10
         r = 15
         x = 12
         print(n > x and r >x) # se cumple que n y r son mayores que x?
```
```
False
```

```
In [26]: print(n > x or r > x)# se cumple que n o r sean mayores que x?
```
```
True
```

Bibliografia

Cassingena Navone, E. (22 de 03 de 2022). *¿Para qué se usa Python? 10+ usos del lenguaje de programación Python*. Obtenido de freecodecamp: https://www.freecodecamp.org/espanol/news/para-que-se-usa-python-10-usos-del-lenguaje-de-programacion-python/

Fernandez Montoro, A. (2013). *Python 3 al descubierto.* Mexico D.F: Alfaomega Grupo Editor S.A de C.V .

Lujan Castillo, J. D. (2020). *Aprende a programar con Python.* Ciudad de Mexico: Alfaomega Grupo Editor S.A de C.V.

Marzal Varo, Garcia Sevilla, A., & Gracia Luengo. (2016). *Introduccion a la programacion con python 3.* Universitat Jaume I. Servei de Comunicació i Publicacions.

Python Software Foundation. (07 de 05 de 2024). *https://docs.python.org/es/3/reference/executionmodel.html*. Obtenido de https://www.python.org/: https://docs.python.org/es/3/reference/executionmodel.html