Python es un lenguaje orientado a objetos, de modo quetiene soporte de primer nivel para la creación de clases.Es un lenguaje de programación, interpretado, de tipadodinámico. De sintaxis sencilla y legible.Es un lenguaje de programación multiparadigma ymultiplatafora.
Interpretado:

Se ejecuta de manera sencilla, sin necesidad de ser procesado por un compilador y se detectan los errores en tiempo de ejecución.

Tipado dinámico: Las variables se comprueban en tiempo de ejecución.

Multiplataforma: Está disponible para plataformas: Windows, MAC o Linux.

Multiparadigma: Soporta programación funcional, imperativa, POO.

Variables:

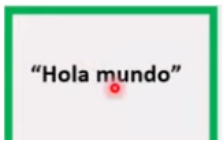
Una variable es un espacio en memoria donde se guardan y recuperan los datos que utiliza un programa.

Son contenedores de información temporal que les permiten guardar datos que pueden ser modificados por acción de un programa.

Enteros: int

Decimales: float

texto: string (str)



Suma de dos números

```
print ("BIENVENIDOS A PYTHON")
print ("Suma de dos números")
numeroUno = 6
numeroDos = 8
suma = numeroUno + numeroDos
print(suma)
```

Cadena de caracteres

La asignación: + =

Consiste en asignar una cadena de caracteres a otra.

```
mensaje = "Hola"
mensaje += " "
mensaje += "clase"
print (mensaje)
```

La concatenación: +

Consiste en unir dos o más cadenas, para formar una cadena de mayo tamaño.

```
mensaje = "Hola"
espacio = " "
nombre  = "clase"
print (mensaje+espacio+nor
```

Cadena de caracteres

Búsqueda: find

Consiste en localizar dentro de una cadena una subcadena mas pequeña a un carácter.

```
mensaje = "Hola clase"
buscarSubcadena = mensaje.find("clase")
print(buscarSubcadena)
```

La extracción: [n : m]

Consiste en sacar fuera de una cadena una porción de la misma según su posición dentro de ella.

```
mensaje = "Hola clase"
extraerSubcadena = mensaje[5:9]
print(extraerSubcadena)
```

Comparación: ==

Consiste en comparar dos cadenas de caracteres.

```
mensajeUno = "Hola clase"
mensajeDos = "Hola clase"
print(mensajeUno == mensajeDos)
```

Operadores lógicos:

Son aquellos que manipulan datos numéricos, tanto enteros como decimales.

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

Los objetos de diferentes tipos, excepto los tipos numéricos, nunca se comparan igual. El operador == siempre está definido, pero para algunos tipos de objetos (por ejemplo, objetos de clase) es equivalente a is. Las instancias no idénticas de una clase normalmente se comparan como no iguales a menos que la clase defina el método __eq__(). Las instancias de una clase no se pueden ordenar con respecto a otras instancias de la misma clase u otros tipos de objeto, a menos que la clase defina los métodos __lt__(), __gt__()

Operadores de asignación:

El operador de asignación se utiliza para asignar un valor a una variable. Como te he mencionado en otras secciones, este operador es el signo =.

Además del operador de asignación, existen otros operadores de asignación compuestos que realizan una operación básica sobre la variable a la que se le asigna el valor.

Por ejemplo, x += 1 es lo mismo que x = x + 1. Los operadores compuestos realizan la operación que hay antes del signo igual, tomando como operandos la propia variable y el valor a la derecha del signo igual.

A continuación, aparece la lista de todos los operadores de asignación compuestos:

+=	x += 2	x = x + 2
-=	x -= 2	x = x – 2
*=	x *= 2	x = x * 2
/=	x /= 2	x = x / 2
%=	x %= 2	x = x % 2
//=	x //= 2	x = x // 2
**=	x **= 2	x = x ** 2
&=	x &= 2	x = x & 2
|=	x |= 2	x = x | 2
^=	x ^= 2	x = x ^ 2
>>=	x >>= 2	x = x >> 2
<<=	x <<= 2	x = x << 2

Tipos de Datos:

Enteros int

Números enteros, positivos y negativos que no tienen decimales.

Largos long

Flotantes o reales float

Tienen decimales tanto positivos como negativos.

Complejos complex

Números que tienen una parte real y una imaginaria.

se puede acceder a la parte real e imagir de los atributos real e imag:

```
>>> real = 1.1 + 2.2  # real es un float
>>> print(real)
3.3000000000000003  # Representación aproximada de 3.3
>>> print(f'{real:.2f}')
3.30  # real mostrando únicamente 2 cifras decimales
```

```
> un_real = 1.1  # El literal debe incluir el carácter .
> otro_real = 1/2  # El resultado de 1/2 es un float
> not_cient = 1.23E3  # float con notación científica (1230.0)
```

Booleanos bool

Tienen dos valores true o false

Por defecto, **cualquier objeto es considerado como verdadero con dos excepciones**:

- Que implemente el método **__bool__()** y este devuelva **False.**
- Que implemente el método **__len__()** y este devuelva **0.**

Cadenas “ texto” str

También conocidas como Strings, texto encerrado entre comillas.

```
>>> caracter_a = 'a'
>>> print(caracter_a)
a
```

```
>>> hola = 'Hola "Pythonista"'
>>> hola_2 = 'Hola \'Pythonista\''
>>> hola_3 = "Hola 'Pythonista'"

>>> print(hola)
Hola "Pythonista"

>>> print(hola_2)
Hola 'Pythonista'

>>> print(hola_3)
Hola 'Pythonista'
```

Conocer el tipo de una variable

`type()`

recibe como parámetro un objeto y devuelve el tipo del mismo.

`isinstance()`:

recibe dos parámetros: un objeto y un tipo. Devuelve **True** si el objeto es del tipo que se pasa como parámetro y **False** en caso contrario.

```
>>> type(3)
<class 'int'>
>>> type(2.78)
<class 'float'>
>>> type('Hola')
<class 'str'>
>>> isinstance(3, float)
False
>>> isinstance(3, int)
True
>>> isinstance(3, bool)
False
>>> isinstance(False, bool)
True
```

```
mensaje = input("Introduce tu nombre: ")
numeroEntero = int(input("Introduce número entero: "))
numeroFlotante = float(input("Introduce número flotante: "))
numeroComplejo = complex(input("Introduce número complejo: "))
print("Bienvenido:“, mensaje)
print("En número entero introducido es : ", numeroEntero)
print("En número flotante introducido es : ", numeroFlotante)
print("En número comlejo introducido es : ", numeroComplejo)
```

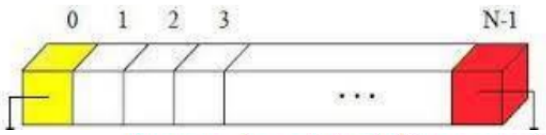
✓ En la práctica, la mayor parte de información útil no aparece aislada en forma de datos simples, sino que lo hace de forma ORGANIZADA y ESTRUCTURADA. Las estructuras de datos es una rama de las ciencias de la computación que estudia y aplica diferentes formas de organizar información dentro de una aplicación para: manipular buscar e insertar

Arreglos

Definición

Un arreglo puede definirse como un grupo o una colección FINITA, **HOMOGÉNEA** y **ORDENADA** de elementos.

- ✓ Un arreglo es un conjunto ordenado de datos por posiciones y todos asociados en una sola variable.
- ✓ Los datos pueden ser de cualquier tipo de dato.
- ✓ Los arreglos empiezan con el subíndice 0.



Arreglos Unidimensionales

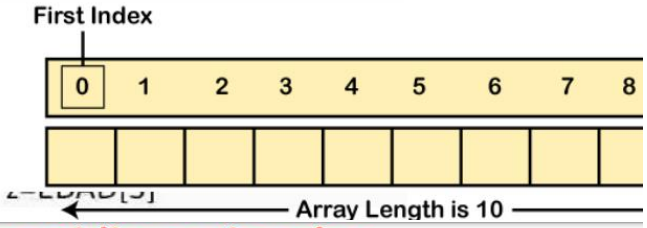
Ejemplo:

Asignación de Posiciones

EDAD[1] = 18

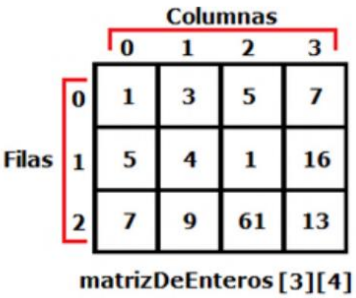
EDAD[2] = 22

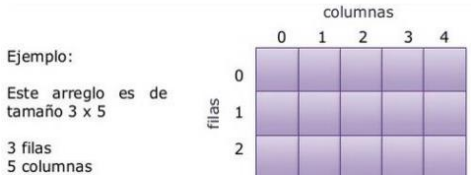
EDAD[3] = 25



Arreglos Bidimensionales

- ✓ Este tipo de arreglos al igual que los anteriores es un tipo de dato estructurado, finito ordenado y homogéneo.
- ✓ El acceso a ellos también es en forma directa por medio de un PAR de índices.



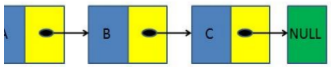


- ✓ Los arreglos bidimensionales se usan para representar datos que pueden verse como una tabla con filas y columnas.

LISTAS Es una SECUENCIA de nodos en los que se guarda información con una o dos referencias (enlaces o punteros) al nodo anterior o posterior. Por ello, los elementos son registros que contienen el dato aalmacenar y un enlace al siguiente elemento Los elementos de una lista, suelen recibir también el nombrede NODOS de la lista.

LISTAS SIMPLEMENTE ENLAZADAS

La lista enlazada es una estructura que nos permite almacenar datos de una forma organizada, al igual que las arreglos pero, a diferencia de estos, esta estructura es dinámica, por lo que no tenemos que saber "**A PRIORI**" los elementos que puede contener.

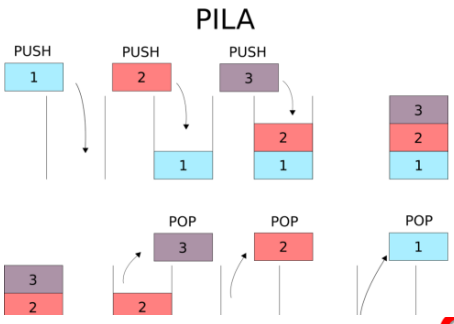


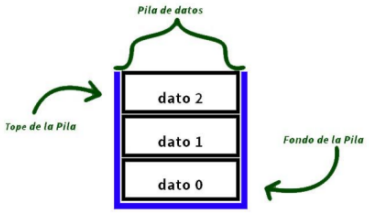
- En una lista, cada elemento apunta al siguiente excepto el último que no tiene sucesor y el valor del enlace es **NULL.**

PILAS

- *Es un contenedor de datos cuyo comportamiento está regido por el principio:*

LIFO (LAST INPUT FIRT OUTPUT).

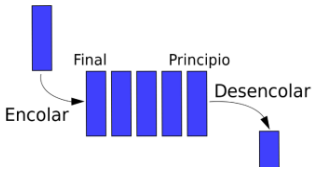




- *Una pila es una estructura de datos homogénea (elementos del mismo tipo), secuencial y de tamaño variable.*

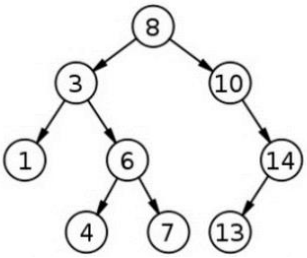
COLAS

- La cola se le considera un primo cercano de la pila.



- La cola puede definirse como un contenedor de datos que funciona de acuerdo al principio **FIFO** (**FIRST INPUT FIRST OUTPUT**) porque el primer elemento que entra a la cola es el primero que sale.

- Sea el siguiente árbol binario determinar los siguientes recorridos:



POS – ORDEN

1 4 7 6 3 13 14 10 8

IN – ORDEN

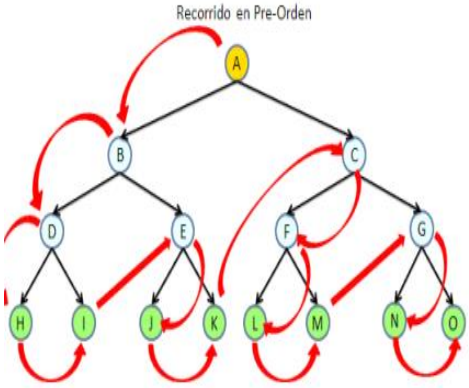
1 3 4 6 7 8 10 13 14

PRE – ORDEN

8 3 1 6 4 7 10 14 13

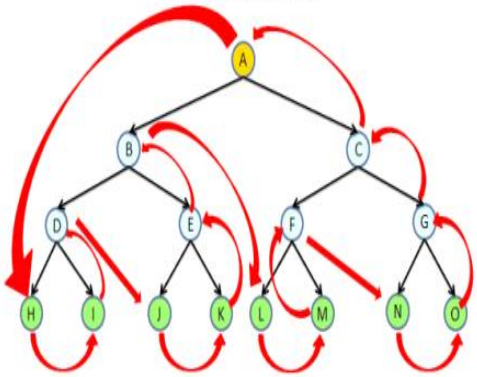
RECORRIDO DE ÁRBOLES

Recorrido en Pre-Orden



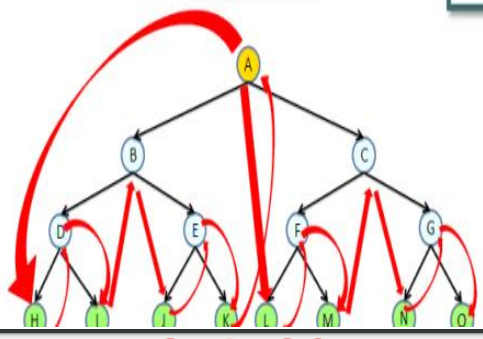
Recorrido: A, B, D, H, I, E, J, K, C, F, L, M, G, N, O

Recorrido en Post-Orden



Recorrido: H, I, D, J, K, E, B, L, M, F, N, O, G, C, A

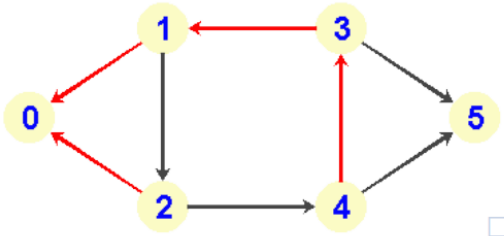
Recorrido en In-Orden

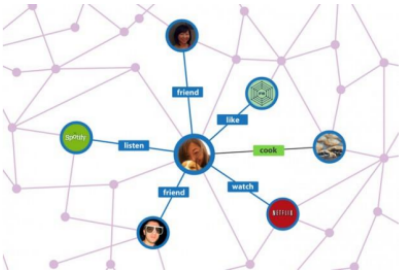


Recorrido: H, D, I, B, J, E, K, A, L, F, M, C, N, G, O

GRAFOS

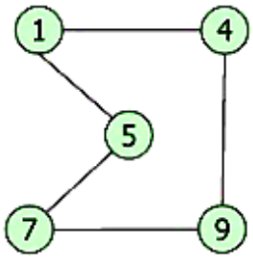
- *Es una colección de NODOS llamados VÉRTICES, los cuales están relacionados entre sí por medio de (ARCOS O ARISTAS).*





- *Los vértices los dibujaremos como puntos del plano, y las aristas serán líneas que unen estos puntos.*

GRAFOS



Un grafo **G = (V , E)**

- ✓ V, es el conjunto de vértices
- ✓ V={v1, v2, ..vn}

V = {1, 4, 5, 7, 9}

- ✓ E, es el conjunto de arcos o aristas
- ✓ E ={vivj, vmvn, ..}

E= {(1,4), (4,9), (9,7), (7,5), (5,1)}

GRADO DE UN VÉRTICE

Es el # de arcos que inciden en un vértice

GRADO DE UN GRAFO

Es la suma del # de arcos que inciden en todos los vértices

TEOREMA DE GRADO DE UN GRAFO

Suma de grados de vértices equivale al doble del número de arcos