Estructura de datos abstractos - TDA

- Pilas
- Colas

Tipos de Datos (primitivos)

El tipo de datos define un conjunto de valores admisibles (restringe el valor que una expresión puede tomar) y define las operaciones que pueden efectuarse sobre datos de ese tipo.

float

int

bool

str

Tipos de Datos (primitivos)

Python define el tipo de datos en la Asignacion (=) según la expresión que evalúa.

n=15

Importe=100.25

par=n%2

encontrado=True

cad="Programacion I"

booleanos, los *numéricos* (enteros, punto flotante y complejos) y las *cadenas de caracteres*.

Tipos de Datos (primitivos)

Otros lenguajes requieren explícitamente indicar el tipo de dato previo a ser utilizada la variable.

```
int n;
…
n=15;
```

```
float importe;
…
Importe=100.25;
```

byte
short
int
long
float
double
boolean
char

Estructura de datos

Un tipo de datos que permite almacenar o guardar más de un dato a la vez.

Listas

Conjuntos

Diccionarios

Tuplas

Secuencias: Los tipos list, tuple y range
Mapas: El tipo dict
Conjuntos: El tipo set

TDA

Si a la estructura de datos le asociamos un conjunto de operaciones para operar con esos datos, se convierte en un

Tipo de Dato Abstracto (TDA)

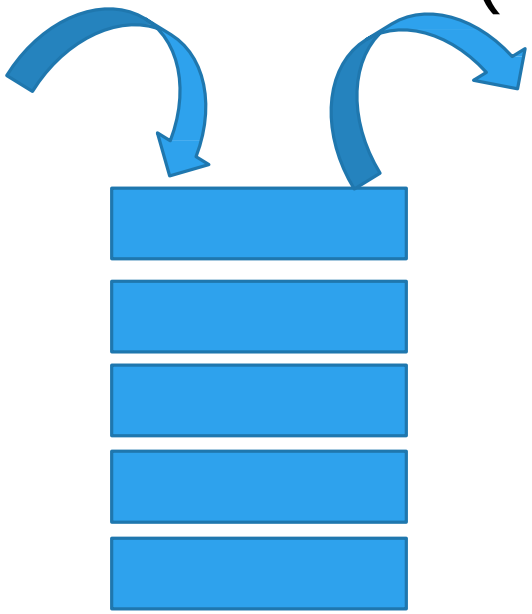
Listas

- append()
- pop()
- insert()
- sort()

métodos:
Definen el comportamiento
Sintáxis:
lista.operación()

Pila

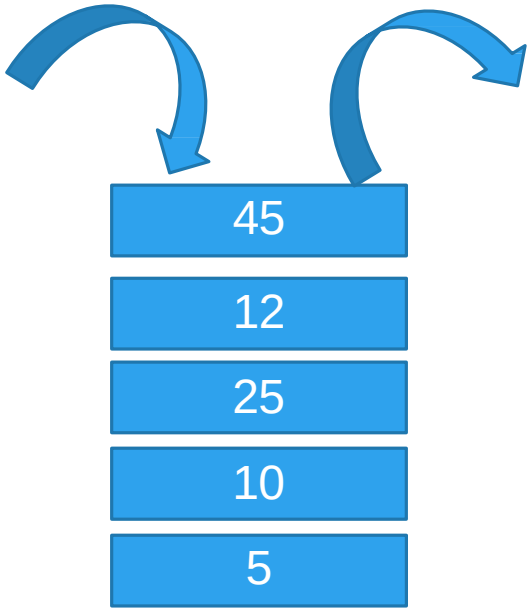
El último elemento que se añade a la estructura es el primero en salir.
Este modo de funcionamiento se conoce como política **LIFO** (Last In, First Out).



PILA

En todo momento, el elemento que ocupa el extremo superior de la pila se denominan **tope**. En toda pila el tope es el **único** elemento visible.

operaciones de TDA-Pila



PILA

`inicializar_pila():` Prepara la pila para ser utilizada, vaciando su contenido.

`apilar(<pila>, <elemento>):` Inserta el elemento en el tope de la pila.

`desapilar(<pila>):` Retira el elemento que se encuentre en el tope de la pila y lo descarta. Genera un error si la pila está vacía.

`tope(<pila>):` Devuelve el elemento que se encuentra en el tope de la pila, pero sin eliminarlo. Genera un error si la pila está vacía.

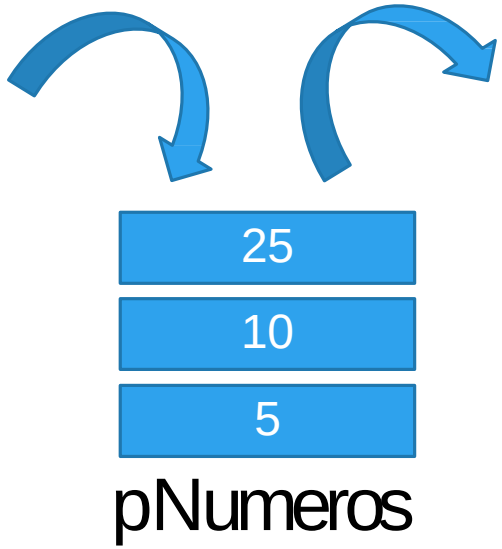
`pila_vacia(<pila>):` Devuelve un 0 (cero => falso) si la pila contiene por lo menos un elemento, o -1 (verdadero) en caso contrario.

EJERCITACIÓN

Pilas:

1. Ingresar tres valores y guardarlos en una Pila.
2. Ingresar valores hasta -1 y guardarlos en una pila.
3. Mostrar los elementos de una pila.

1. Ingresar tres valores y guardarlos en una Pila.

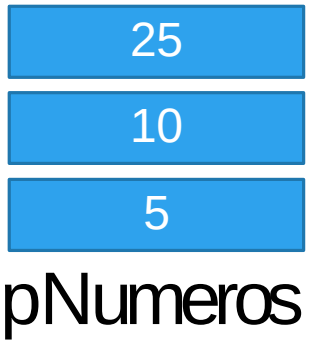


1. Crear e inicializar Pila pNumeros
2. Tres veces:
   - solicitar un numero por teclado
     apilar el numero en la Pila pNumeros

```
pnum=inicializar_pila()
for i in range(3):
    dato=int(input("Ingrese un valor"))
    apilar(pnum, dato)
```

Ingresar valores hasta -1 y guardarlos en una pila.

1. **Crear e inicializar Pila pNumeros**
2. **Mientras Verdadero:**
   solicitar un numero por teclado
   si el numero es -1
   finaliza ciclo
   apilar el numero en la Pila pNumeros

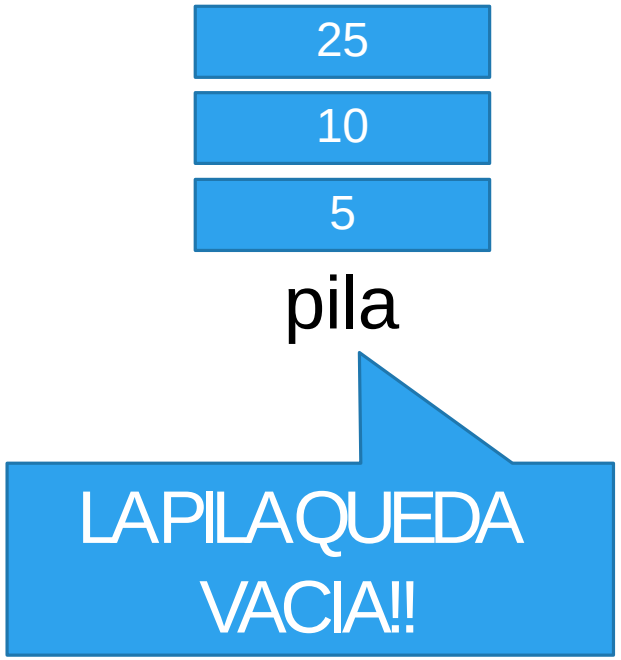



```
from pila import *

pNumeros = inicializar_pila()
numero = int(input("Ingrese un número: "))
while numero != -1:
    apilar(pNumeros, numero)
    numero = int(input("Ingrese un número: "))

print(pNumeros)
```

Mostrar los elementos de una pila

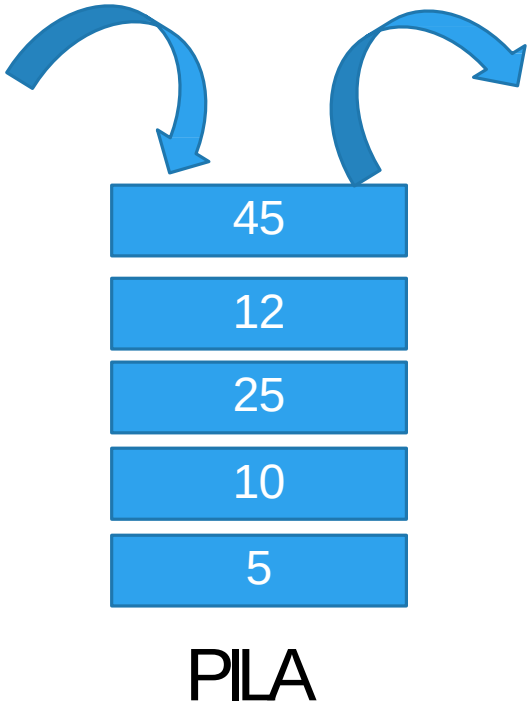
Función: MostrarPila(pila)

1. Mientras no este vacia la pila
obtengo el tope de la pila
mostrar el tope
desapilo un elemento de la pila




```
def mostrarPila(p):
    while not pila_vacia(p):
        print(tope(p))
        desapilar(p)
```

Implementar un TDA-Pila




```
def inicializar_pila():
    pila=[]
    return pila

def apilar(pila, dato):
    pila.append(dato)

def desapilar(pila):
    pila.pop()

def tope(pila):
    return pila[-1]

def pila_vacia(pila):
    return len(pila)==0
```

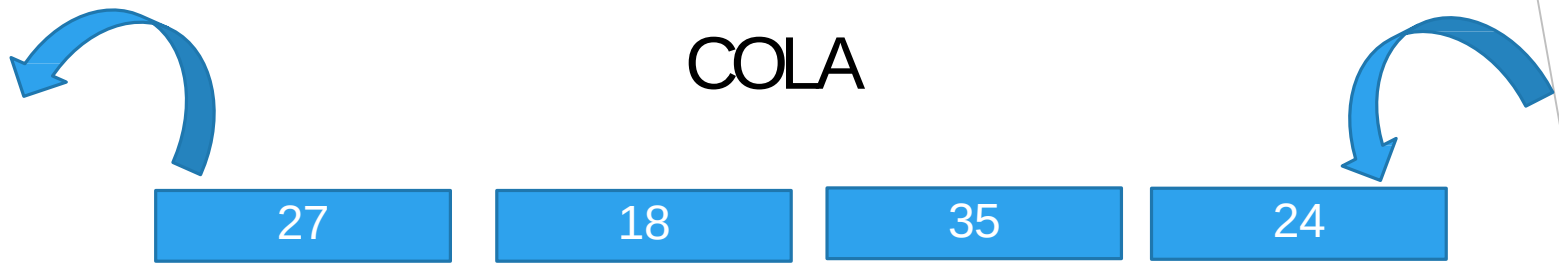
Colas

Los elementos ingresan por un extremo (denominado **fondo**) y salen por el otro extremo, llamado **primero.**
Este modo de funcionamiento se conoce como política **FIFO** (First In, First Out).

COLA

En toda cola el **primero** es el elemento visible.

Operaciones de TDA-Cola



`inicializar_cola()`: Prepara la cola para ser utilizada, vaciando su contenido.

`acolar(<cola>, <elemento>)`: Inserta el elemento en el fondo de la cola.

`desacolar(<cola>)`: Retira el elemento que se encuentre en el frente de la cola y lo descarta. Genera un error si la cola está vacía.

`primero(<cola>)`: Devuelve el elemento que se encuentra en la cabeza de la cola, pero sin eliminarlo. Genera un error si la cola está vacía.

`cola_vacia(<cola>)`: Devuelve Verdadero si no contiene elementos, Falso en caso contrario.

EJERCITACIÓN

Colas:

1. Ingresar tres valores y guardarlos en una Cola.
2. Ingresar valores hasta -1 y guardarlos en una Cola.
3. Mostrar los elementos de una Cola.

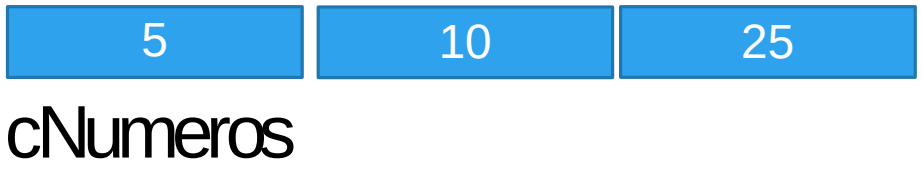
1. Ingresar tres valores y guardarlos en una Cola.

1. Crear e inicializar Cola cNumeros
2. Tres veces:
   - solicitar un numero por teclado
     acolar el numero en la Cola cNumeros


```
cNumeros=inicializar_cola()
for i in range(3):
    dato=int(input("Ingrese un valor"))
    acolar(cNumeros, dato)
```

Ingresar valores hasta -1 y guardarlos en una cola.

1. **Crear e inicializar Cola cNumeros**
2. **Mientras Verdadero:**
   solicitar un numero por teclado
   si el numero es -1
   finaliza ciclo
   acolar el numero en la Cola cNumeros




```
from cola import *


cNumeros = inicializar_cola()
numero = int(input("Ingrese un número: "))
while numero != -1:
    acolar(cNumeros, numero)
    numero = int(input("Ingrese un número: "))
```

Mostrar los elementos de una cola

Función: MostrarCola(cola)

1. Mientras no este vacia la cola

obtengo el primero de la cola

mostrar el primero

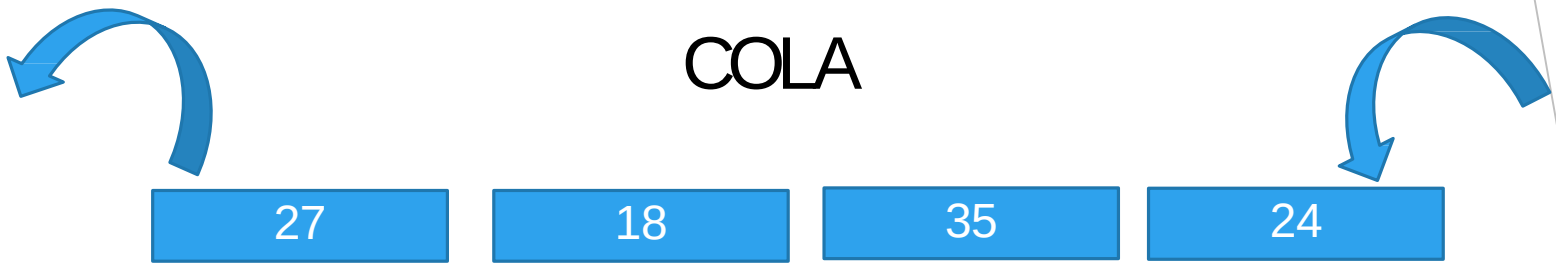
desacolar un elemento de la cola

cNum

LA COLA QUEDA VACIA!!

```
def mostrarCola(cNum):
    while not cola_vacia(cNum):
        print(primero(cNum), end=" ")
        desacolar(cNum)
    print()
```

Implementar un TDA-Cola




```
def inicializar_cola():
    cola=[]
    return cola
```

```
def acolar(cola, dato):
    cola.append(dato)

def desacolar(cola):
    cola.pop(0)

def primero(cola):
    return cola[0]
```

```
def cola_vacia(cola):
    return len(cola)==0
```

EJERCITACIÓN

1. Crear una pila, pasar la base al tope de la misma.

2. Invertir el orden de los elementos de una cola, utilizando solo colas como estructura auxiliar.

EJERCITACIÓN

TP 9 COMPLETA