El futuro digital es de todos

MinTIC

Mision TIC 2022

Aprendiendo a programar

Diego Iván Oliveros Acosta

UNIVERSIDAD DE ANTIOQUIA

Facultad de Ingeniería

El futuro digital es de todos

MinTIC

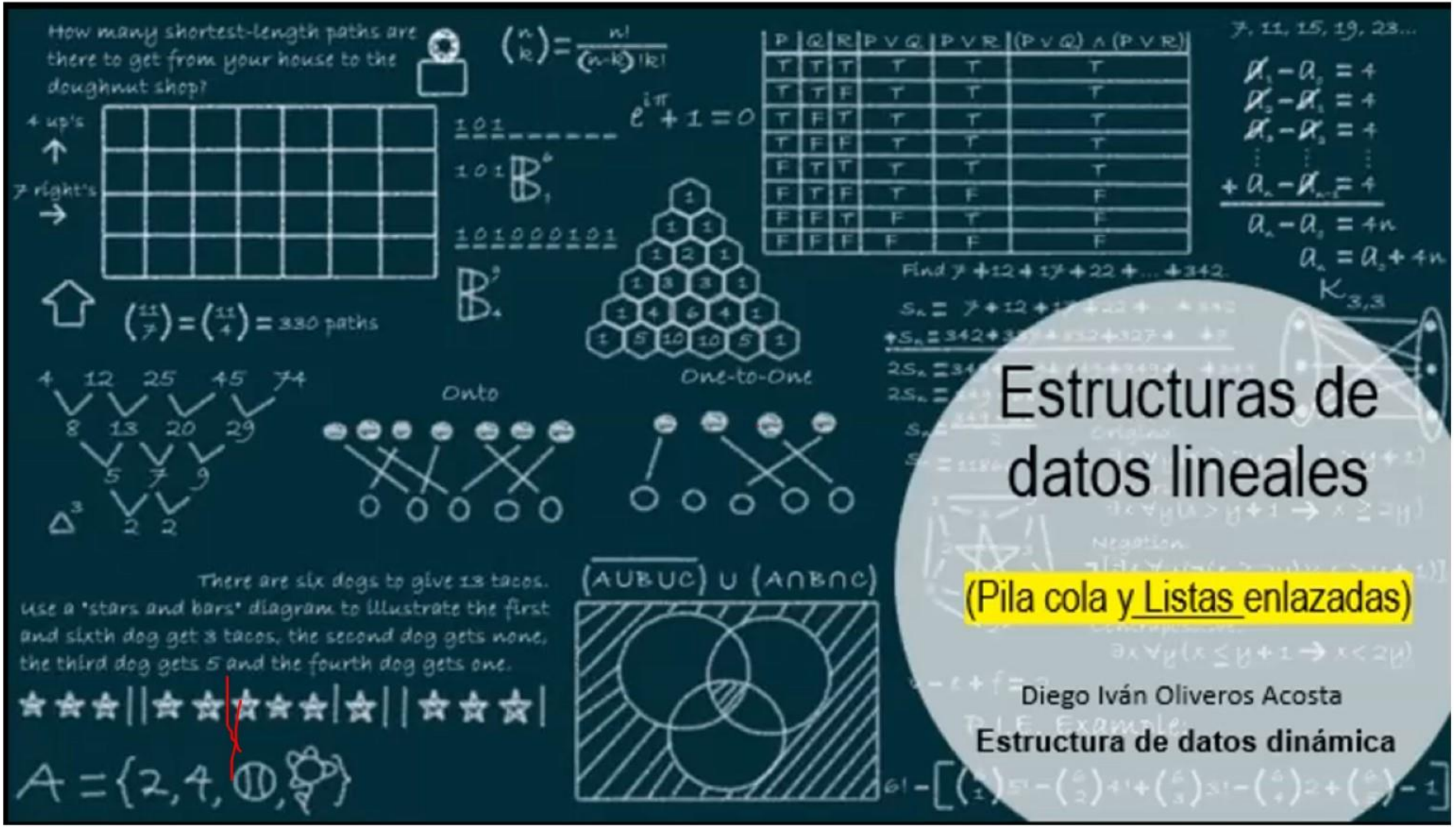
UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Misión TIC 2022

CICLO 1 - FUNDAMENTOS DE PROGRAMACIÓN

Introducción a la Algoritmia e Introducción al Lenguaje de Programación Python

Diego Iván Oliveros Acosta
@scalapp.co


How many shortest-length paths are there to get from your house to the doughnut shop?
4 up's
7 right's
$\binom{11}{7} = \binom{11}{4} = 330$ paths
$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
$e^{i\pi} + 1 = 0$
101
101
B^0
B_1
10100010101
B^9
B_4
P | Q | R | P ∨ Q | P ∨ R | (P ∨ Q) ∧ (P ∨ R)
7, 11, 15, 19, 23...
$a_1 - a_0 = 4$
$a_2 - a_1 = 4$
$a_3 - a_2 = 4$
$+ a_n - a_{n-1} = 4$
$a_n - a_0 = 4n$
$a_n = a_0 + 4n$
$K_{3,3}$
Find 7 + 12 + 17 + 22 + ... + 342.
4 12 25 45 74
8 13 20 29
5 7 9
Δ^3
2 2
Onto
One-to-One
There are six dogs to give 13 tacos. Use a "stars and bars" diagram to illustrate the first and sixth dog get 3 tacos, the second dog gets none, the third dog gets 5 and the fourth dog gets one.
$A = \{2, 4, 10, \}$
$(\overline{A \cup B \cup C}) \cup (A \cap B \cap C)$
Estructuras de datos lineales
(Pila cola y Listas enlazadas)
Diego Iván Oliveros Acosta
Estructura de datos dinámica

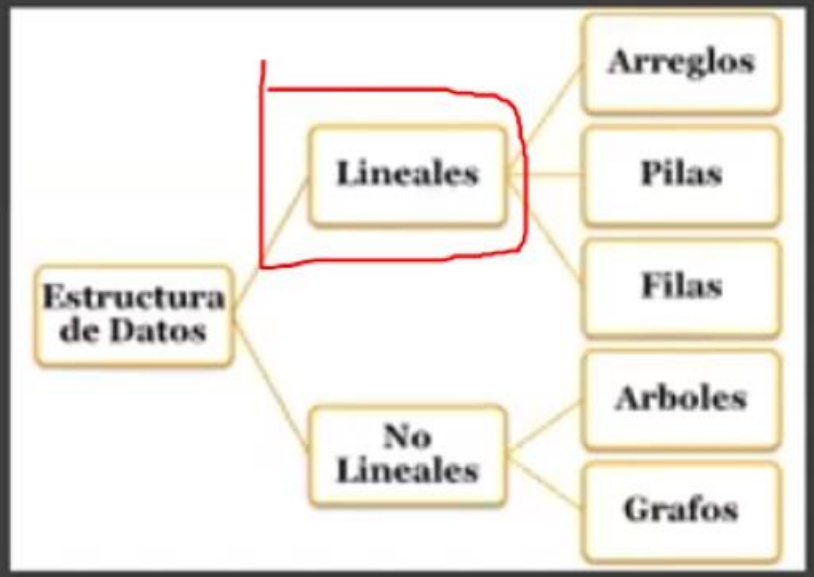
Colecciones de Python (matrices)

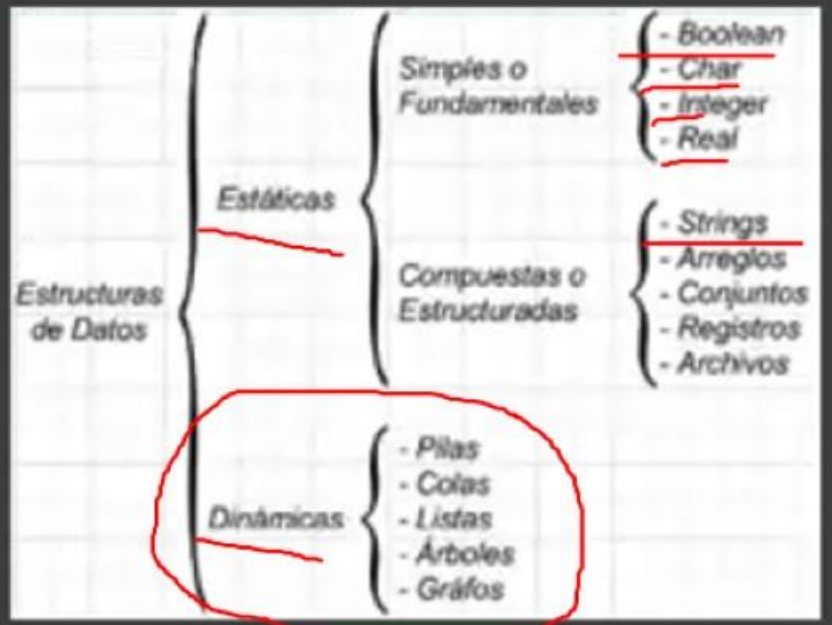
Hay cuatro tipos de datos de recopilación en el lenguaje de programación Python:

- La lista
- La tupla
- El conjunto
- El diccionario

- ¿y los set?

Concepto

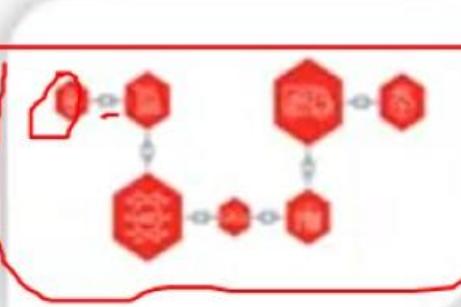

Estructura de Datos
Lineales
Arreglos
Pilas
Filas
No Lineales
Arboles
Grafos

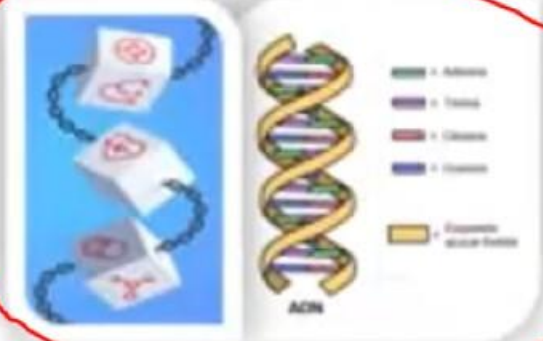

Estructuras de Datos
Estáticas
Simples o Fundamentales
- Boolean
- Char
- Integer
- Real
Compuestas o Estructuradas
- Strings
- Arreglos
- Conjuntos
- Registros
- Archivos
Dinámicas
- Pilas
- Colas
- Listas
- Árboles
- Gráfos

Pila

Cola

Lista enlazada

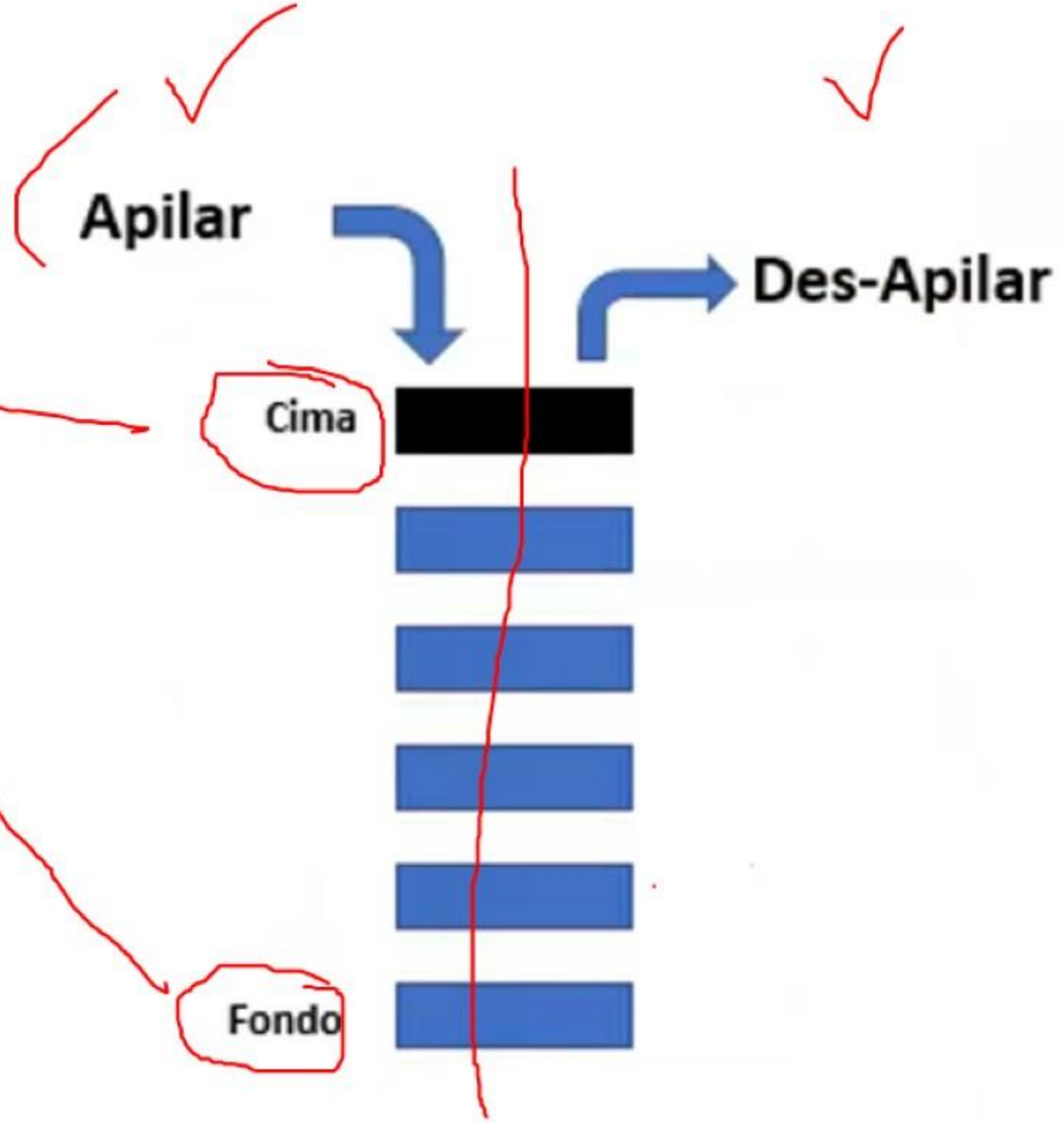




Pila

- LIFO (**L**ast **I**n **F**irst **O**ut)
- Tope, fondo.
- Apilar()
- Desapilar() //*
- Eliminar()
- Está vacia()

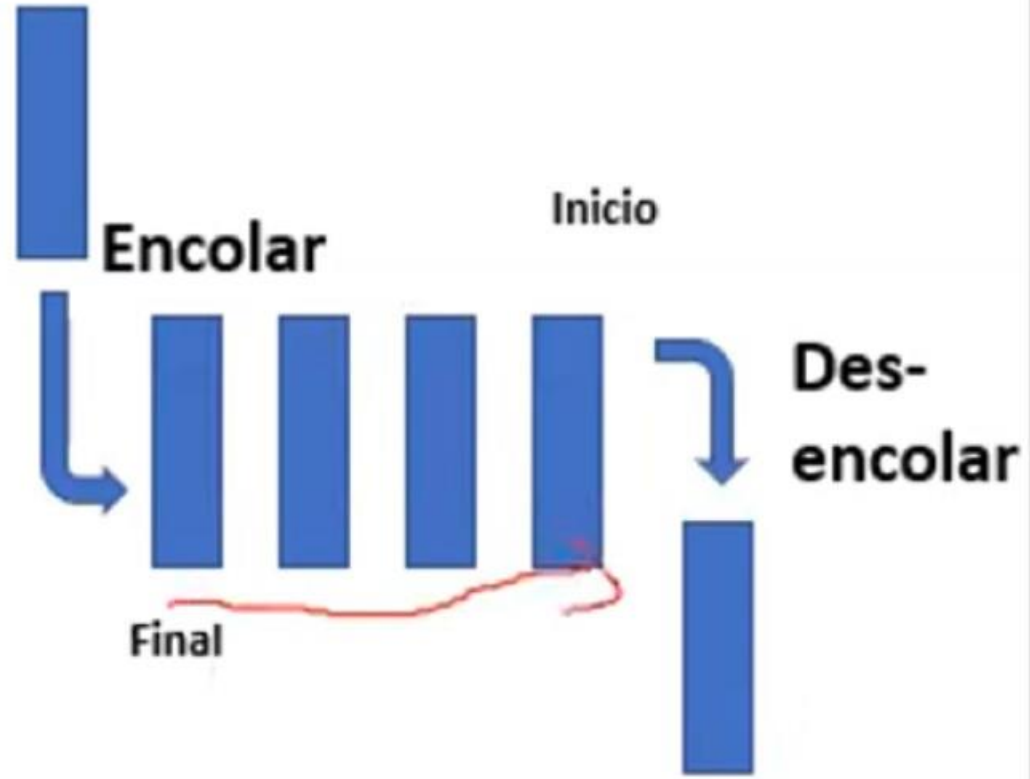
- If p es no vacía **then** desapilar



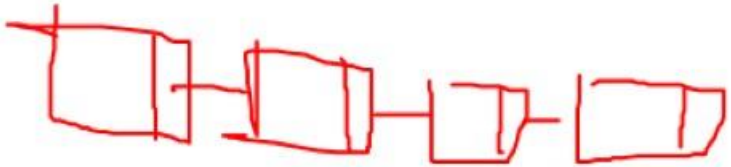
Cola

- FIFO (**F**irst **I**n **F**irst **O**ut)
- Final, inicio
- Encolar()
- Desencolar() //*
- Está vacia
- Eliminar()

- **Colas circulares**



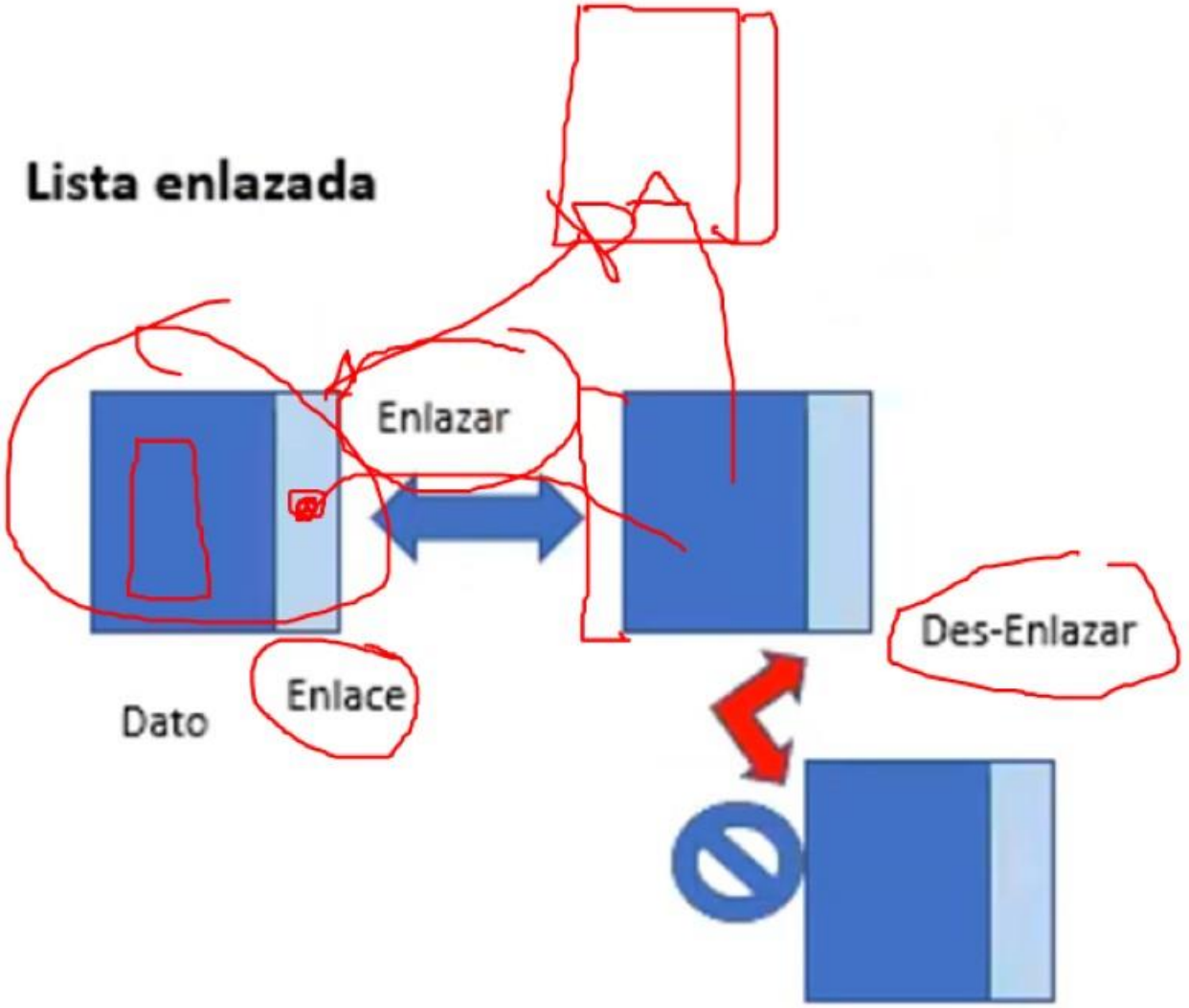
Lista enlazada



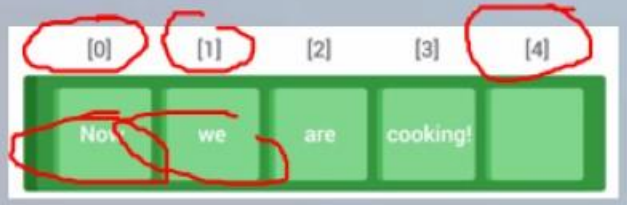
Punto de enlace o interés

- **Son Homogéneas, doblemente enlazadas. Acceso secuencial.**
- Punto de enlace
- Dato / *null
- Siguiente, anterior
- Desenlazar ()
- Enlazar()
- Inicio()
- **Crear nodo, igualar nodo, recorrer/buscar**

Lista enlazada



Implementación en Python



Operaciones con listas en Python

- Las listas se utilizan para almacenar varios elementos en una sola variable. Las **listas** se crean usando corchetes.
- Las listas son uno de los 4 tipos de datos incorporados en Python que se utilizan para almacenar colecciones de datos, los otros 3 son Tuple , Set y Dictionary , todos con diferentes calidades y usos.
  - Mostrar un item
  - Cambiar un valor
  - Adicionar
  - Insertar
  - Remover
  - Índice negativo
  - Imprimir un rango
  - Calcular longitud

Pilas

```
#Pilas

pila=[1,2,3,4]

#Agregar elementos a la pila

pila.append(5)
pila.append(6)

#Sacar elementos de la pila

n=pila.pop()
n=pila[-1]
```

Colas

```
#Colas

Cola=["Juan", "Karla"]

#Agregar elementos a la Cola

Cola.append("María")
Cola.append("Pedro")    //insert(i,op)

#Sacar elementos de la Cola

n=Cola.pop(0)
n=Cola[0]
```

Lista enlazada

```
# Creamos la clase node
class node:
    def __init__(self, data = None, next = None):
        self.data = data
        self.next = next

# Creamos la clase linked_list
class linked_list:
    def __init__(self):
        self.head = None
```

What is a dictionary?

```
• file_counts ={"jpg":10, "txt" :14, "csv":2,
  "py":23}
• print(file_counts)
• file_counts["txt"]
• "Jpg" in file_counts
• "html" in file_counts
• file_counts["cfg"]=8
• file_counts["cvs"]=17
```

- Puede iterar sobre diccionarios usando un bucle for, al igual que con cadenas, listas y tuplas. Esto iterará sobre la secuencia de claves en el diccionario. Si desea acceder a los valores correspondientes asociados con las claves, puede usar las claves como índices.
- O puede usar el método de elementos en el diccionario, como **dictionary.items().** Este método devuelve una tupla para cada elemento del diccionario, donde el primer elemento de la tupla es la clave y el segundo es el valor.
- Si solo quisiera acceder a las claves en un diccionario, podría usar el método keys() en el diccionario: **dictionary.keys().** Si solo quisiera los valores, podría usar el método de valores(): **dictionary**.values().

```
ip_addresses = ["192.168.1.1", "127.0.0.1", "8.8.8.8"]

host_addresses = {"router": "192.168.1.1", "localhost": "127.0.0.1",
"google": "8.8.8.8"}
```

+ + +

Dictionary Methods Cheat Sheet

Definition

x = {key1:value1, key2:value2}

Diego Iván Oliveros Acosta
@scalapp.co

Operations

len(dictionary) - Returns the number of items in the dictionary

for key in dictionary - Iterates over each key in the dictionary

for key, value in dictionary.items() - Iterates over each key,value pair in the dictionary

if key in dictionary - Checks whether the key is in the dictionary

dictionary[key] - Accesses the item with key key of the dictionary

dictionary[key] = value - Sets the value associated with key

del dictionary[key] - Removes the item with key key from the dictionary

Methods

- dict.get(key, default) - Returns the element corresponding to key, or default if it's not present
- dict.keys() - Returns a sequence containing the keys in the dictionary
- dict.values() - Returns a sequence containing the values in the dictionary
- dict.update(other_dictionary) - Updates the dictionary with the items coming from the other dictionary. Existing entries will be replaced; new entries will be added.
- dict.clear() - Removes all the items of the dictionary

Ejercicio

- In Python, a dictionary can only hold a single value for a given key. To workaround this, our single value can be a list containing multiple values. Here we have a dictionary called "wardrobe" with items of clothing and their colors. Fill in the blanks to print a line for each item of clothing with each color, for example: "red shirt", "blue shirt", and so on.

```
wardrobe = {"shirt":["red","blue","white"], "jeans":["blue","black"]}
for llave in wardrobe.keys():
    for item in wardrobe[llave]:
        print("{item} {llave}".format(item=item, llave=llave))
```

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Misión TIC 2022

Tuplas y colecciones

- Recorrido de tuplas
- **Conjuntos**
- Añadir y eliminar datos en un conjunto
- Recorrido de conjuntos
- **Tuplas y sets**

Diego Iván Oliveros Acosta

FACULTAD DE INGENIERIA **UdeA**
Una facultad para la sociedad del aprendizaje

@scalapp.co

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

```
>>> # Tuples are immutable:
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item as
signment
>>> # but they can contain mutable objects:
... v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

- **Listas**
- **https://docs.python.org/3/tutorial/datastructures.html#more-on-lists**
- **Diccionarios:**
- https://docs.python.org/3/library/stdtypes.html#mapping-types-dict
- **Tuplas y secuencias:**
- https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences