GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN

Agenda
Sesión 11/18

1. **Punto 1:** introducción a los arreglos
2. **Punto 2:** longitudes array
3. **Punto 3:** crear y recorrer arreglos
4. **Punto 4:** pruebas automatizadas

PROGRAMACIÓN CON PYTHON

ARREGLOS EN PYTHON

Arreglos (Listas)

¿Qué son?

En otros lenguajes de programación son conocidos como arreglos, arrays o vectores. En Python se conocen como listas

Arreglos (Listas)

¿Qué son?

- **Son una estructura de datos o un tipo de dato que nos permiten almacenar gran cantidad de valores.**
- **Python nos permite almacenar diferentes tipos de valores (enteros, booleanos, strings, etc) esto no es posible en otros lenguajes de programación.**
- **Se pueden expandir dinámicamente añadiendo nuevos elementos (a diferencia de otros lenguajes que no lo permiten hacer directamente sino mediante otras estructuras)**

Arreglos (Listas)

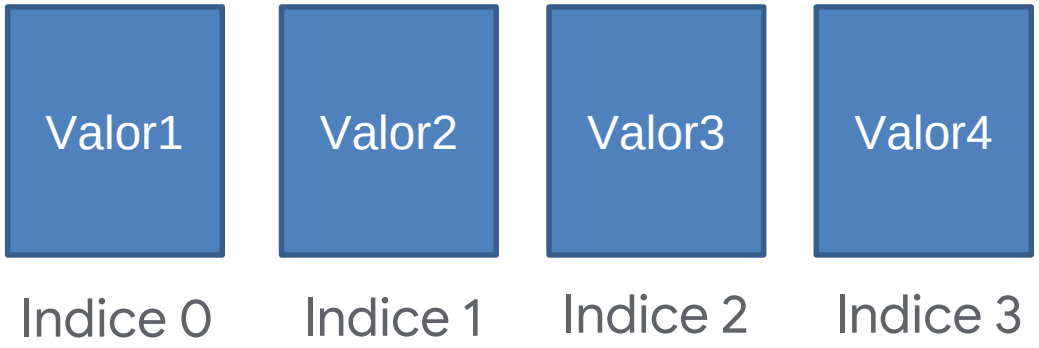
Sintaxis

- **milista=[valor1, valor2, valor3, valor4….]**

Arreglos (Listas)

- **milista=[valor1, valor2, valor3, valor4….]**

Los índices



Arreglos (Listas)

```
lista_participantes=["Francisco","Luis","Ana","Carlos","Juan"]
print(lista_participantes[:])
```

Consultar lista

```
####################
```

```
lista_participantes=["Francisco","Luis","Ana","Carlos","Juan"]
print(lista_participantes[2])
```

Arreglos (Listas)

```
lista_participantes=["Francisco","Luis","Ana","Carlos","Juan"]
lista_participantes.append("Susana")#agrega el elemento al final
print(lista_participantes)
```

```
##################
```

Agregar elementos

```
lista_participantes=["Francisco","Luis","Ana","Carlos","Juan"]
lista_participantes.insert(3,"Susana")

#insert necesita 2 argumentos
#el primero el indice y el segundo el item a agregar
print(lista_participantes)
```

Arreglos (Listas)

Agregar elementos

```
lista_participantes=["Francisco","Luis","Ana","Carlos","Juan"]

lista_participantes.extend(["Ernesto","Fátima","Ruth","Pedro"])

print(lista_participantes)
```

Arreglos (Listas)

Buscar elementos

```
lista_participantes=["Francisco","Luis","Ana","Carlos","Juan"]

lista_participantes.extend(["Ernesto","Fátima","Ruth","Pedro"])

#¿dónde está Fátima?
print(lista_participantes.index("Fátima"))
```

Arreglos (Listas)

Diferentes datos

```
lista_participantes=["Francisco","Luis","Ana",10,"Carlos","Juan",False]

lista_participantes.extend(["Ernesto","Fátima","Ruth","Pedro"])

#¿dónde está Fátima?
print(lista_participantes.index)
```

Arreglos (Listas)

```
lista_participantes=["Francisco","Luis","Ana",10,"Carlos","Juan",False]
```

```
lista_participantes.extend(["Ernesto","Fátima","Ruth","Pedro"])
```

Eliminar elementos

```
lista_participantes.remove("Fátima")
```

```
#¿dónde está Fátima?
print("Fátima" in lista_participantes)
print(lista_participantes)
```

Arreglos (Listas)

```
lista_participantes=["Francisco","Luis","Ana",10,"Carlos","Juan",False]

lista_participantes.extend(["Ernesto","Fátima","Ruth","Pedro"])
```

Eliminar el último elemento.

```
lista_participantes.pop()#elimina el último elemento

#¿dónde está Fátima?
print("Fátima" in lista_participantes)
print(lista_participantes)
```

Arreglos (Listas)

```
lista_participantes=["Francisco","Luis","Ana",10,"Carlos","Juan",False]

otra_lista_participantes=["Ernesto","Fátima","Ruth","Pedro"]
```

Unir listas.

```
lista_final=lista_participantes+otra_lista_participantes

print(lista_final)
```

Arreglos (Listas)

```
lista_participantes=["Francisco","Luis","Ana",10,"Carlos","Juan",False] * 4
```

Repetir listas.

```
print(lista_participantes)
```

PROGRAMACIÓN CON PYTHON

RECORRER LISTA

Recorrer lista

```
numeros_primos = [1,3,5,7,11,13,17,19,23,29]

for numero in numeros_primos:
        print(numero)

############
numeros_primos = [1,3,5,7,11,13,17,19,23,29]

for numero in numeros_primos:
        print(numero*2)
#print(numeros_primos)
```

PROGRAMACIÓN CON PYTHON

LONGITUD DE LISTA

Longitud de lista

```
eltexto = "Python es una gran onda"
print(len(eltexto))

##############

def calcula_longitud(lalongitud):
        contador = 0

        for caracter in lalongitud:
                contador += 1
        return contador

eltexto = "Python es una gran onda"
print(calcula_longitud(eltexto))
```

PROGRAMACIÓN CON PYTHON

PRUEBAS AUTOMATIZADAS

Pruebas automatizadas

- **Diferentes formas para realizar pruebas automatizadas.**
- **Dentro de las practicas de buen desarrollador se encuentra las pruebas de los diferentes desarrollos que realicen, lo que los diferencia es la forma en la que realizan dichas pruebas. Por eso es tan importante la implementación de pruebas ya que ayudan a verificar errores y dificultades del desarrollo rápidamente, para poder solucionarlos en la brevedad.**

¿Por qué automatizar?

- La principal razón es el tiempo y con las pruebas automatizadas se puede reducir el tiempo de las pruebas. Además al automatizar las actividades comunes que no requieren de inteligencia humana, los testers pueden dedicar mayor tiempo a pruebas más críticas y caminos más elaborados dejando los caminos básicos a las pruebas automatizadas.
- Rapidez: Las herramientas de testing automatizado corren las pruebas significativamente más rápido que los testers humanos.
- Fiabilidad: Las pruebas ejecutan precisamente las mismas operaciones cada vez que se ejecutan, eliminando el error humano.
- Repetición: Se puede testear como reacciona el script bajo repetidas ejecuciones de las mismas operaciones.
- Programable: Se pueden programar pruebas sofisticadas y complejas que muestren información oculta de la aplicación.
- Reusabilidad: Se pueden rehusar los scripts con pruebas automatizadas

Pruebas unitarias (Unittest)

- **Verifican una unidad (unit) de código.**
- **Una unidad es la parte más pequeña que se puede testear (rutina, método, objeto, función, etc)**

Ventajas

Pruebas unitarias (Unittest)

- **Podemos dividir nuestro código en unidades, esto permite encontrar los bugs de una forma más sencilla y rápida.**
- **Fácil mantenimiento de nuestro código.**
- **Documentaremos nuestro código.**
- **Trabajan de forma independientes, cada prueba no tiene relación ni depende de otras.**

Doctest

Document testing

```
#Función que calcula el área de un triángulo
def calcula_area(base,altura):
        '''
        Comprobando el funcionamiento del script

        >>> calcula_area(10,5)
        25.0

        '''
        return (base*altura)/2

import doctest #importar módulo
doctest.testmod()
```

Doctest

Document testing

```
#Función que calcula el área de un triángulo
def calcula_area(base,altura):
        '''
        Comprobando el funcionamiento del script

        >>> calcula_area(10,5)
        25
        '''
        return "El área del triángulo es: " + str((base*altura)/2)

import doctest
doctest.testmod()
```

Doctest

Document testing

```
#Función que calcula el área de un triángulo
def calcula_area(base,altura):
        """
        Comprobando el funcionamiento del script

        >>> calcula_area(10,5)
        'El área del triángulo es: 25.0'
        """
        return "El área del triángulo es: " + str((base*altura)/2)

import doctest
doctest.testmod()
```

Doctest

Varias pruebas al mismo tiempo

```
#Función que calcula el área de un triángulo
def calcula_area(base,altura):
        """
        Comprobando el funcionamiento del script

        >>> calcula_area(10,5)
        'El área del triángulo es: 25.0'
        >>> calcula_area(9,5)
        'El área del triángulo es: 22.5'
        >>> calcula_area(10,6)
        'El área del triángulo es: 30.0'
        """
        return "El área del triángulo es: " + str((base*altura)/2)
import doctest
doctest.testmod()
```

Doctest

```
def verifica_mail(elmail):

    """
    >>> verifica_mail('francisco@quezada.com')
    True
    """

    verifica_arroba=elmail.count('@') #cuenta las @en elmail
```

Valida correo electrónico

```
#si hay 0 @ o más de una
        #rfind si la @ esta al final
        #si no hay @ (otra forma de hacerlo)
        if(verifica_arroba!=1 or elmail.rfind('@')==(len(elmail)-1) or elmail.find('@')==0):
                return False
        else:
                return True

import doctest
doctest.testmod()
```

RESUMEN DE SESIÓN

GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN