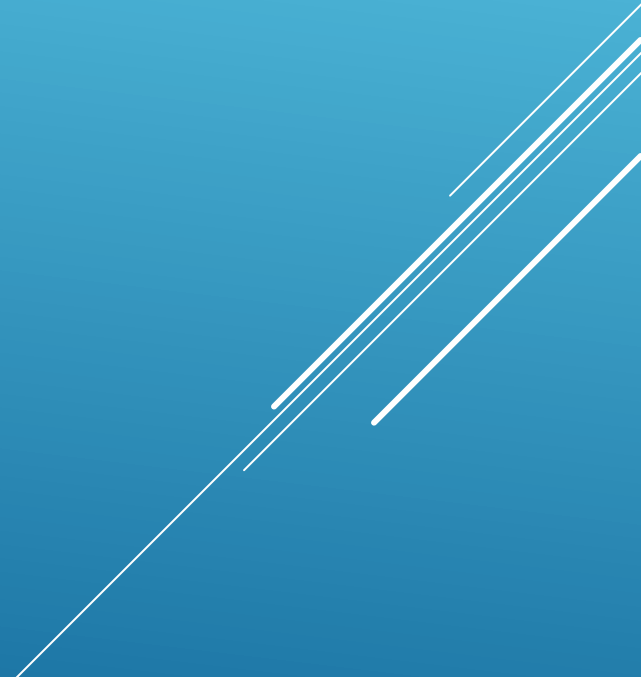


CLASE MODULO 3

PYTHON




CLASES PARA ESTE MODULO

- ▶ Clases divididas en 2 partes :
 - ▶ Parte 1 : teoria y practica
 - ▶ Parte 2 preguntas # El examen consiste en contestar preguntas
- 
- Several white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.




BIBLOGRAFIA





python for beginners
// part 1 of 44


Programming with Python



level: novice

44 lecciones

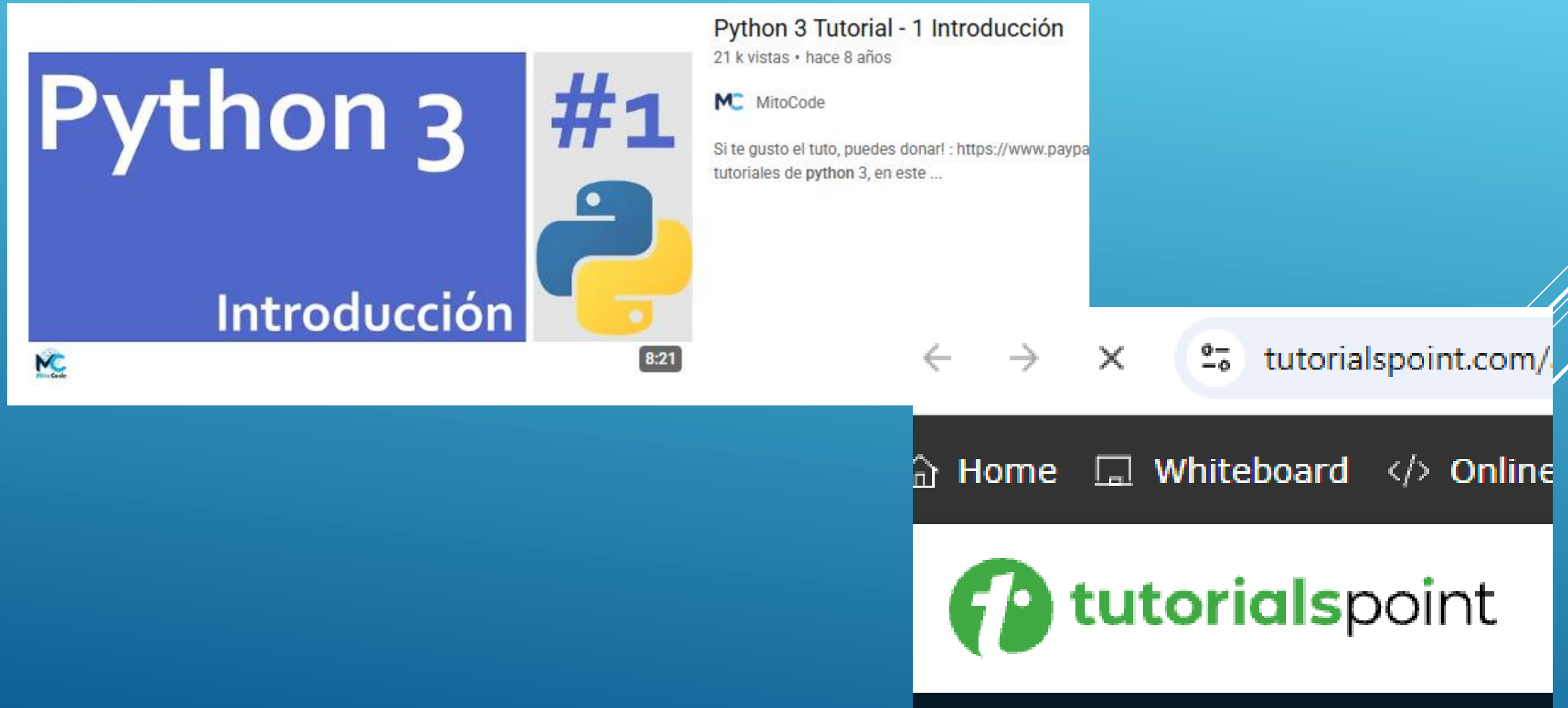
Python for Beginners

Microsoft Developer  • Curso

Programming with Python | Python for Beginner
Introducing Python | Python for Beginners [2 of 3]

Ver curso completo

BIBLOGRAFIA



The image displays a video player interface for a Python 3 tutorial. The video title is "Python 3 Tutorial - 1 Introducción", with 21 k views and posted 8 years ago. The channel is MitoCode. The video thumbnail features the text "Python 3" and "Introducción" on a blue background, alongside a "#1" and the Python logo. A duration of 8:21 is shown at the bottom right of the video player.

Below the video player, a browser window is visible, showing the URL "tutorialspoint.com/". The browser's navigation bar includes icons for Home, Whiteboard, and Online. The tutorialspoint logo is prominently displayed at the bottom of the browser window.

BIBLOGRAFIA

[About](#) ▾[Certifications](#) ▾[Study Resources](#) ▾[Community](#) ▾[Voucher Store](#)

Python Essentials 1

Dive into programming, learn Python from scratch, and prepare for the PCEP – Certified Entry-Level Python Programmer certification.

This introductory course gives you an opportunity to **dive into Python** and computer programming with **no specific prerequisites or prior knowledge** required. It will guide you from a state of complete programming illiteracy to a level of programming knowledge which will allow you to **design, write, debug, and run Python scripts**, and to understand the basic concepts of software development technology.

Having completed the course, you will be prepared to attempt the qualification [PCEP – Certified Entry-Level Python Programmer](#) certification, and to get your foot in the door to careers in **software development, data analysis, and testing**.

BEGINNER



General-Purpose
Programming Track

643954



Python Essentials 1

Beginner
 6-8 weeks
 English, Spanish
 Free

Python Essentials 1

This course is the first in a 2-course series that will prepare you for the *PCEP™ – Certified Entry-Level Python Programmer* certification exam, and help you build the essential foundations for the *PCAP™ – Certified Associate Python Programmer* certification exam.

Facilitador



Jonathan Aguirre



TIPOS DE DATOS

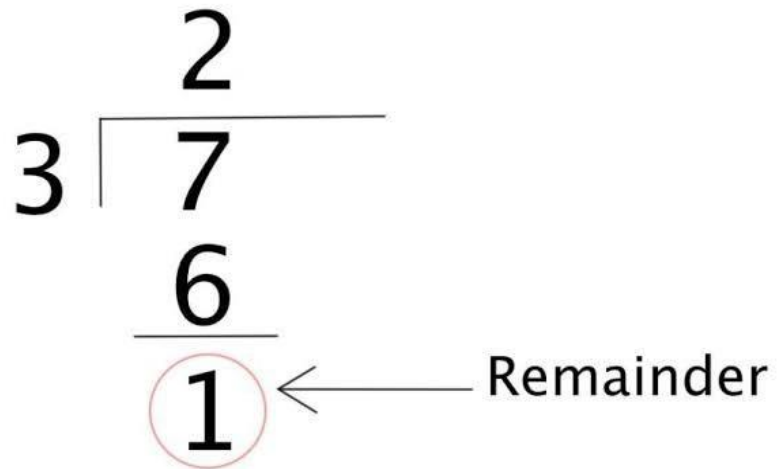
- Str : Letras o caracteres alfa numericos, "abc123# \$" o 'abc123# \$'
- Int : Numeros enteros , 1, 2, 3,
- Float : Numeros decimales, 3.14, 2.7278, 1.0
- Bool : condicion logica Verdadero o Falso, True, False
- `type(mi_variable)`

OPERADORES

- ▶ $(+, -, *, **, /, //,)$
- ▶ # division flotante : $/$,
- ▶ # division entera o piso : $//$ (hacia el **menor** entero más cercano).
- ▶ $5//2 = (2.5 \text{ division flotante}) = 2$ # no redondea positivos
- ▶ $-5//2 = (-2.5 \text{ division flotante}) = -3$ # no redondea positivos, menos 3 no es mayor que menos 2.5
- ▶ $(=, +=, -=, /=, \%=, //=, **=)$
- ▶ $A = A/b \rightarrow a /= b$

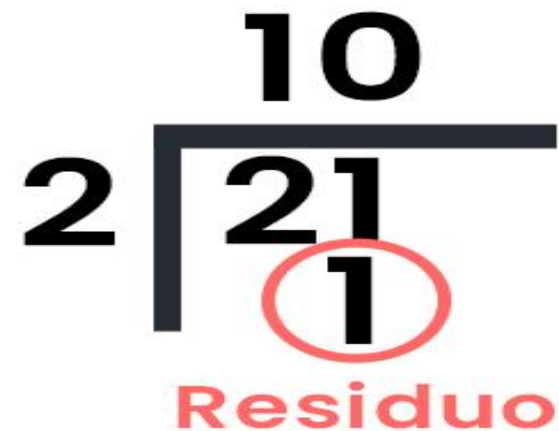
OPERADORES

- El operador módulo (%) en Python se usa para hallar el resto tras dividir un número entre otro. Funciona tanto con números enteros como con números de punto flotante.



A handwritten long division of 7 by 3. The number 3 is on the left, and 7 is under a horizontal line. Above the line is the quotient 2. Below 7 is the product 6, with a horizontal line underneath. Below 6 is the remainder 1, which is circled in red. An arrow points from the word "Remainder" to the circled 1.

$$\begin{array}{r} 2 \\ 3 \overline{) 7} \\ \underline{6} \\ 1 \end{array} \leftarrow \text{Remainder}$$



A handwritten long division of 21 by 2. The number 2 is on the left, and 21 is under a horizontal line. Above the line is the quotient 10. Below 21 is the product 20, with a horizontal line underneath. Below 20 is the remainder 1, which is circled in red. Below the circled 1 is the word "Residuo" in red.

$$\begin{array}{r} 10 \\ 2 \overline{) 21} \\ \underline{20} \\ 1 \end{array} \text{Residuo}$$

OPERADORES

| Operador | Sintaxis | Significado |
|----------|------------|-------------------|
| < | $a < b$ | Menor que |
| <= | $a \leq b$ | Menor o igual que |
| > | $a > b$ | Mayor que |
| >= | $a \geq b$ | Mayor o igual que |
| == | $a == b$ | Igual a |
| != | $a != b$ | Distinto de |
| ! | !a | Negación lógica |

CONVERSION DE TIPO DE DATOS

- ▶ `str(42)` # '42'
- ▶ `int('123')` # 123
- ▶ `float('3.1415')` # 3.1415
- ▶ `float(1)` # 10.0

CONVERSION DE TIPO DE DATOS

- ▶ `bool(True)` # *True*
- ▶ `bool("Hola")`
- ▶ `bool("")` # si es texto vacio, sera siempre falso

VARIABLES

- ▶ En Python, una variable es un nombre simbólico que actúa como una etiqueta o contenedor para almacenar datos (como números, textos, listas) en la memoria del ordenador, permitiendo acceder y manipular esa información fácilmente a lo largo del código. # (AUTOMATIZAR)

- ▶ **Nombres validos :**

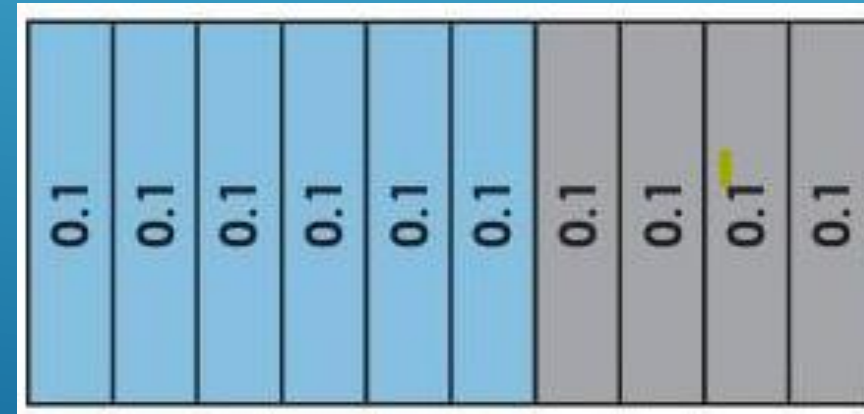
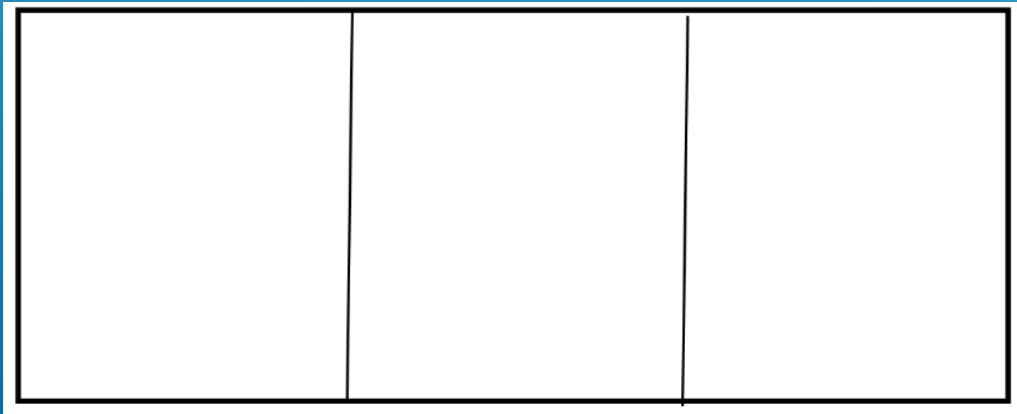
- ▶ edad_nombre, numero1, total_pago, _usuario, valor, x, _y, nombre_casada

- ▶ **Nombres NO Validos**

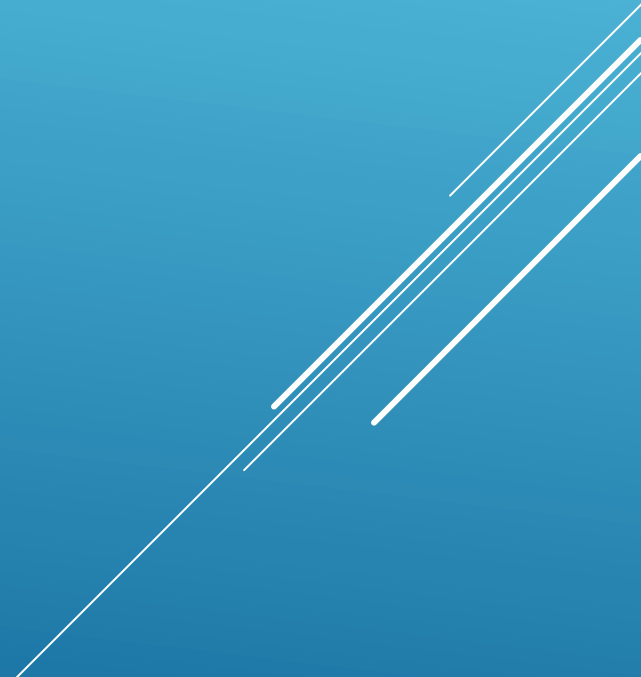
- ▶ 1variable # empieza con un dígito, @nombre # contiene un carácter no permitido (@)
total-pago # contiene un guion (-), class # palabra reservada en Python. Nombre casada

LISTAS

- ▶ Las listas se utilizan para almacenar varios elementos en una sola variable.
- ▶ Resuelve problemas del tipo : Necesitamos almacenar los nombres de los empleados de la empresa, sin la necesidad de crear demasiadas variables y la complejidad que esto implica
- ▶ Lista = [1,2,3,4,5]




LISTAS

- ▶ El método `sort()` ordena la lista en orden ascendente de forma predeterminada.
 - ▶ `cars = ['Ford', 'BMW', 'Volvo']`
 - ▶ `cars.sort()`
 - ▶ `cars.sort(reverse=True)`
- 
- Several white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.


LISTAS

- ▶ La función `sorted()` acepta cualquier iterable (listas, tuplas, cadenas, diccionarios, etc.) y devuelve una nueva lista ordenada, dejando el iterable original sin cambios.
- ▶ `data = (5, 2, 8, 1, 9)` # A tuple
- ▶ `sorted_data = sorted(data)`
- ▶ `print(sorted_data)` # Output: `[1, 2, 5, 8, 9]`
- ▶ `print(data)` # Output: `(5, 2, 8, 1, 9)`

LISTAS

- ▶ El método `extend()` añade los elementos de una lista al final de otra lista in situ. Es eficiente para listas grandes cuando no se necesita la original.
 - ▶ `list1 = [1, 2, 3]`
 - ▶ `list2 = [4, 5, 6]`
 - ▶ `list1.extend(list2)`
 - ▶ `print(list1)`
- 
- Several white lines of varying lengths and orientations are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

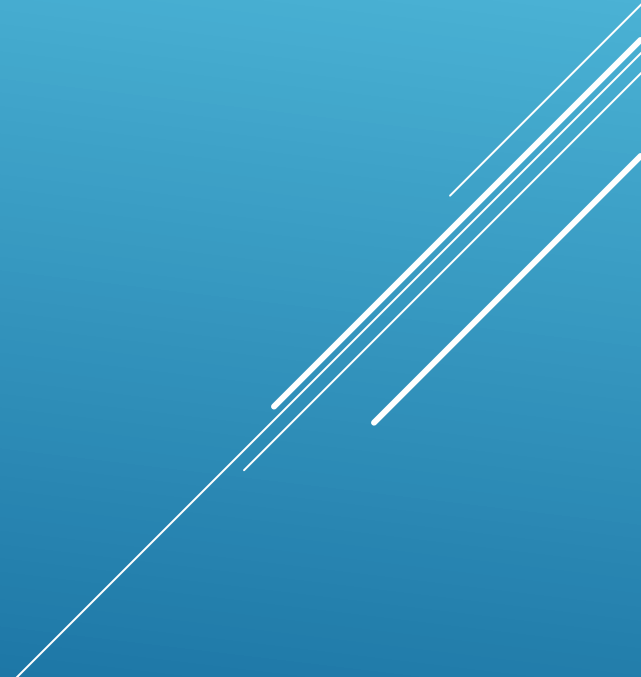
LISTAS

- ▶ El operador + crea una nueva lista concatenando los elementos de ambas listas. Es simple y legible para listas pequeñas.
 - ▶ `list1 = [1, 2, 3]`
 - ▶ `list2 = [4, 5, 6]`
 - ▶ `list3 = list1 + list2`
 - ▶ `print(list3)`
- 
- Several white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.


LISTAS

- ▶ En Python, la forma principal de añadir un elemento a una lista es mediante el método integrado `append()`. Este método modifica la lista original directamente añadiendo un elemento al final.
- ▶
- ▶ `fruits = ['apple', 'banana', 'cherry']`
- ▶
- ▶
- ▶ `fruits.append('orange')`

LISTAS

- ▶ Para insertar un elemento de lista en un índice específico, utilice el método `insert()`.
 - ▶ El método `insert()` inserta un elemento en el índice especificado:
 - ▶ `thislist = ["apple", "banana", "cherry"]`
 - ▶ `thislist.insert(1, "orange")`
 - ▶ `print(thislist)`
- 
- Several white lines of varying lengths and orientations are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.


LISTAS

- ▶ Utilice el método `remove()` cuando conozca el valor del elemento que desea eliminar. Este método elimina la primera ocurrencia del valor especificado.
 - ▶ `my_list = ["apple", "banana", "cherry"]`
 - ▶ `my_list.remove("banana")`
 - ▶ `print(my_list)`
- 
- Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

LISTAS

- ▶ Utilice el método `pop()` cuando conozca el índice del elemento que desea eliminar. Este método también devuelve el elemento eliminado, lo cual puede ser útil si necesita almacenarlo o usarlo.
- ▶ `my_list = ["apple", "banana", "cherry"]`
- ▶ `removed_item = my_list.pop(1)`
- ▶ `print(my_list)`

LISTAS

- ▶ La instrucción `del` es una opción flexible para eliminar elementos por índice o eliminar un rango de elementos (una porción). A diferencia de `pop()`, no devuelve el valor eliminado.
 - ▶ `my_list = ["apple", "banana", "cherry", "date", "elderberry"]`
 - ▶ `del my_list[1]`
 - ▶ `print(my_list)`
- 
- Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

LISTAS

- ▶ Para encontrar el índice (posición) de la primera aparición de un elemento, utilice el método `list.index()`. Tenga en cuenta que esto genera un `ValueError` si no se encuentra el elemento.
- ▶ `my_list = ["apple", "banana", "cherry", "banana"]`
- ▶ `index = my_list.index("banana")`
- ▶ `print(f'First occurrence is at index: {index}')`

LISTAS

- ▶ La forma más común y eficiente de comprobar si un elemento está en una lista es mediante el operador in. Este devuelve Verdadero o Falso.
- ▶ `my_list = ["apple", "banana", "cherry"]`
- ▶ `if "banana" in my_list:`
 - ▶ `print("Element found!")`
 - ▶ `else:`
 - ▶ `print("Element not found.")`

LISTAS

- ▶ El método `list.count()` devuelve el número de veces que un elemento aparece en una lista. Si el recuento es mayor que cero, el elemento existe.
- ▶ `my_list = ["apple", "banana", "cherry", "banana"]`
- ▶ `count = my_list.count("banana")`
- ▶ `print(f'The element 'banana' appears {count} times.')`

LISTAS

- ▶ La función `max()` es una característica incorporada en Python que compara eficientemente todos los elementos de un iterable (como una lista, tupla o conjunto) y devuelve el elemento con el valor más alto.
- ▶ `my_list = [10, 24, 76, 23, 12]`
- ▶ `maximum_value = max(my_list)`
- ▶ `print(maximum_value)`

LISTAS

- ▶ La función `min()` toma un iterable (como una lista) como argumento y devuelve el elemento más pequeño.
- ▶ `numbers = [4, 7, 2, 9, 1, 5]`
- ▶ `minimum_value = min(numbers)`
- ▶ `print("Minimum number:", minimum_value)`

LISTAS

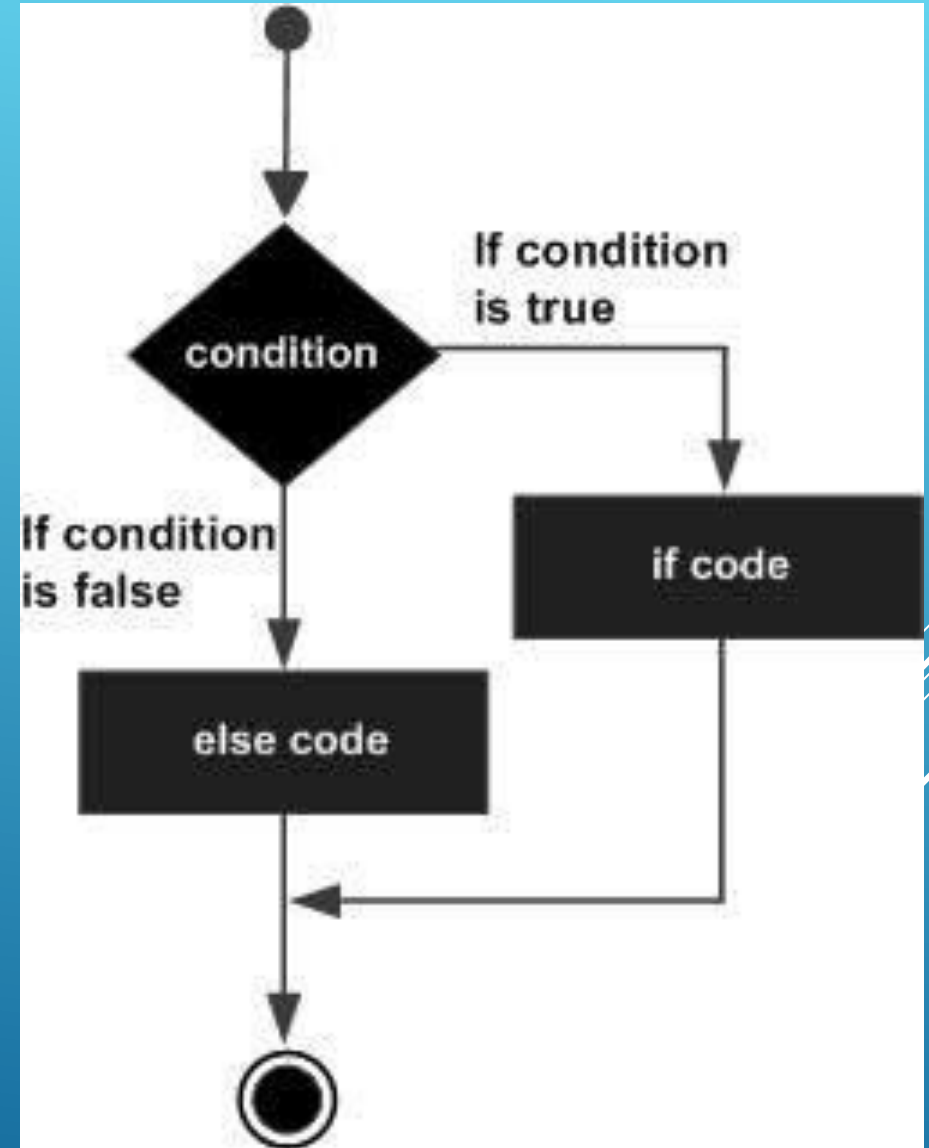
- ▶ La función `min()` toma un iterable (como una lista) como argumento y devuelve el elemento más pequeño.
- ▶ `numbers = [4, 7, 2, 9, 1, 5]`
- ▶ `minimum_value = min(numbers)`
- ▶ `print("Minimum number:", minimum_value)`

IF ELSE

Una declaración if-else es una condición básica de programación que ejecuta un bloque de código si una condición especificada es verdadera y un bloque de código alternativo si la condición es falsa.

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```



BUCLES

- ▶ for item in sequence:
 - ▶ # code block to be executed
 - ▶ # 'item' takes the value of each element in the sequence
- ▶ while condition:
 - ▶ # code block to be executed
 - ▶ # make sure to change variables involved in the condition
 - ▶ # to avoid an infinite loop

BUCLES

- ▶ Para recorrer un conjunto de código una cantidad específica de veces, podemos usar la `range()` función
- ▶ `#inicio default = 0`
- ▶ `#salto default = 1`

- ▶ `for x in range(inicio, final, salto):`
- ▶ `print(x)`

BUCLES

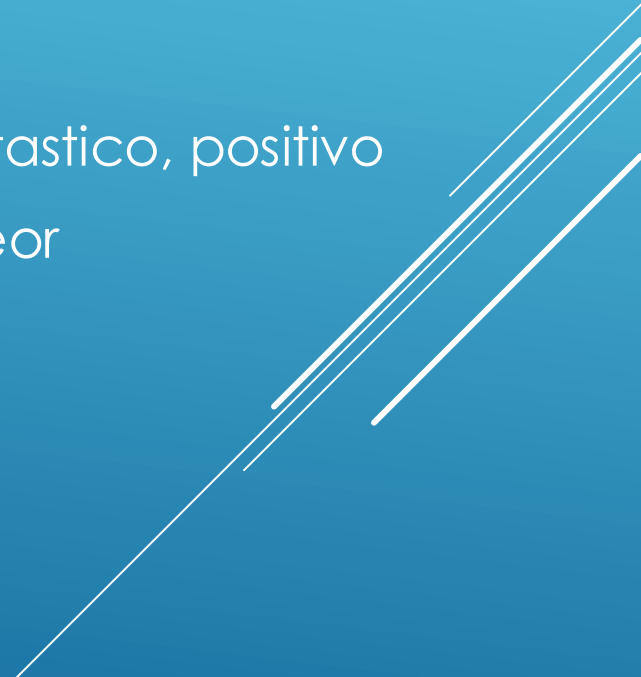
- ▶ For letra in "palabra":
- ▶ print(letra)

- ▶ adj = ["red", "big", "tasty"]
- ▶ fruits = ["apple", "banana", "cherry"]

- ▶ for x in adj:
- ▶ for y in fruits:
- ▶ print(x, y)

- ▶ For x in array
- ▶ Print(x)
- ▶ Else:
- ▶ Print("else")

EJERCICIO

- ▶ Determine si una frase del usuario es positiva, negativa o neutro:
 - ▶ Palabras positivas : bueno, excelente, agradable, estupendo, fantastico, positivo
 - ▶ Palabras negativas : malo, negativo, nefasto, feo, mal, horrible, peor
- 
- Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

EJERCICIO

- ▶ Desarrolle el ordenamiento de burbuja :
- ▶ https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

BUCLES

- ▶ While condicion:
- ▶ Codigo
- ▶ Incremento

- ▶ While condicion:
- ▶ Codigo
- ▶ Incremento
- ▶ Else:
- ▶ Print("else")

PASS

- ▶ En la programación Python, la `pass` declaración es una declaración nula que puede usarse como marcador de posición para código futuro.
- ▶ `Def funcion():`
 - ▶ `pass`

CONTINUE

- ▶ una declaración de control de bucle que omite el código restante dentro de un bucle solo para la iteración actual y salta a la siguiente iteración
- ▶

```
for number in range(5):
```
- ▶

```
    if number == 3:
```
- ▶

```
        continue # Skip the rest of the code in the current iteration
```
- ▶

```
    print(number)
```

BREAK

- ▶ La sentencia `break` en Python se utiliza para finalizar prematuramente un bucle (`for` o `while`), saliendo inmediatamente de él y continuando con el resto del código.
- ▶ `# Detener el bucle cuando 'i' sea 5`
- ▶ `for i in range(10):`
- ▶ `if i == 5:`
- ▶ `break # Sale del bucle completamente`
- ▶ `print(i)`

EJERCICIO

▶ *****345

FUNCIONES DE PYTHON

- ▶ Una función es un bloque de código que solo se ejecuta cuando se lo llama.
- ▶ Una función puede devolver datos como resultado.
- ▶ Una función ayuda a **evitar la repetición** de código.
- ▶

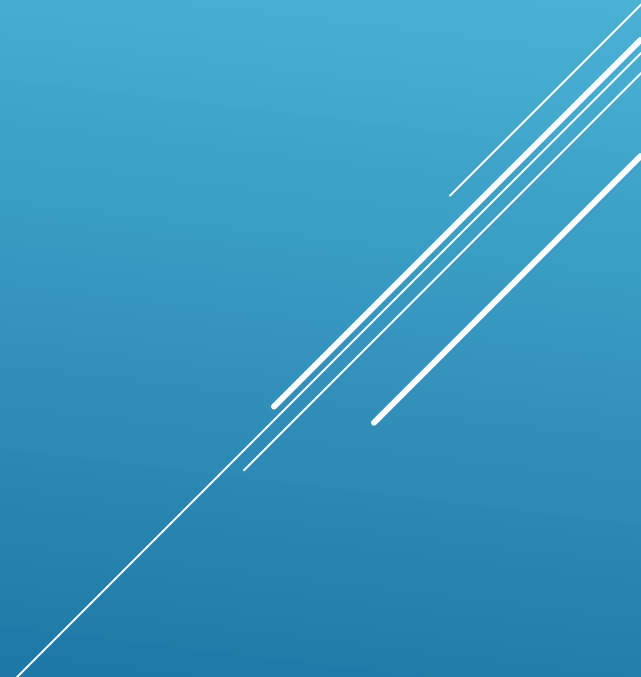
```
def my_function():
```
- ▶

```
    print("Hello from a function")
```

VALORES DE RETORNO

- ▶ Las funciones pueden enviar datos al código que las llamó mediante la declaración `return`.
- ▶ Cuando una función llega a una `return` declaración, deja de ejecutarse y envía el resultado de vuelta:
- ▶ `def get_greeting():`
- ▶ `return "Hello from a function"`
- ▶ `message = get_greeting()`
- ▶ `print(message)`

PREGUNTAS



- ¿Cuál es el resultado esperado del siguiente código?
- `print(1 / 1)`
-
- A) 1.0
- B) no puede ser evaluado
- C) error de syntaxs
- F) ninguna de las anteriores

- Cual es el resultado del siguiente codigo ?
- `num = 2 + 3 * 5`
- `print(num)`
-
-
- A) 25.0
- B) 17
- C) error
- D) ninguna de las anteriores

- Cual de los siguientes no es un operador arimetico en Python ?

-
- A) +
 - B) -
 - C) x
 - D) **

- cual es el valor de la variable j ?

- $J = \text{"B"}$

- $J = \text{srt}(\text{"A"})$

- $C = 2 - 4$

- $D = \text{"A"}$

- A) J

- B) P

- C) A

- D) B

- ¿Qué operador se puede usar para comparar si dos valores son semejantes ?
-

- A) =
- B) <>
- C) ==
- D) equal

- cuantos errores visualiza en el siguiente programa ?
-

- `I = I`
- `While I < 6`
- `If (x < 2)`
- `I *= 2`
- `elseif`
- `i**I`
- `endif`
- `compile program`

- ¿Cuál es el resultado esperado del siguiente código?
- `x = '\\\\'`
- `print(len(x))`
-
-
- A) 4
- B) 2
- C) 1
- F) ninguna de las anteriores

- Como se realiza un comentario de bloque en python

-
- A) `/* */`
 - B) `# block1`
`#block2`
 - C) `“”” “””`
 - D) `<!-- -->`

- ¿Cómo se crea una variable con el número flotante 2.8?
-

- A) `x = 2.8`
- B) `x = float(2.8)`
- C) ambas formas son correctas

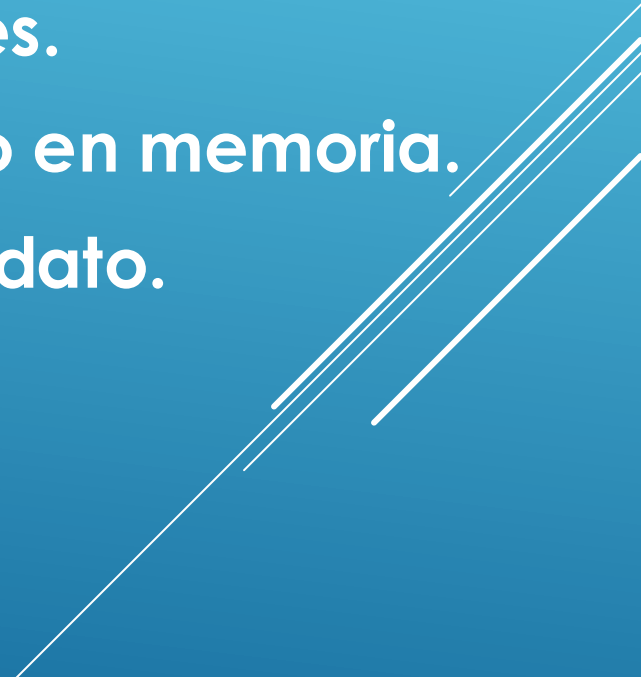
- ¿Cuál es la forma correcta de crear una función en Python?
-

- A) create MyFunction()
- B) def Myfunction()
- C) function myFunction()

- ¿Cuál de estas colecciones define una Lista ?
-

- A) {2,3,4,5}
- B) (2,3,4,5)
- C) [2,3,4,5]
- D) list(2,3,4,5)

► ¿Qué hace el operador is en Python?

- A) Compara si los valores de dos variables son iguales.
 - B) Compara si dos variables apuntan al mismo objeto en memoria.
 - C) Compara si dos variables tienen el mismo tipo de dato.
 - D) Compara si una variable es mayor que otra.
- 
- Several white lines of varying lengths and orientations are drawn in the bottom right corner of the slide, creating a modern, abstract design element.

▶ # Ejemplo de uso del operador 'is'

▶ `a = [1, 2, 3]`

▶ `b = a`

▶ `c = [1, 2, 3]`

▶ `print(a is b)` # True, porque a y b apuntan al mismo objeto en memoria

▶ `print(a is c)` # False, aunque tienen el mismo contenido, son objetos diferentes

- ¿Cuál es la salida del siguiente código?

- valueOne = 5 ** 2

- ~~• valueTwo = 5 ** 3~~

- print(valueOne)

- print(valueTwo)

- 10

- 15

- 25

- 125

- Evalúe la siguiente expresión aritmética de Python:
 - $(3 * (1 + 2) ** 2 - (2 ** 2) * 3)$
 - Cual sera el resultado ?
-

- A) 15
- B) 13
- C) 88
- D) ninguna de las anteriores

- Cuál es la salida del siguiente código

- `def printSalary():`

- `salary = 12000`

- `print("Salary:", salary)`

-

- `printSalary()`

- `print("Salary:", salary)`

- A) Salary: 12000 Salary: 8000

- B) Salary: 8000 Salary: 12000

- C) The program failed with errors

- Cual es el resultado del siguiente codigo ?

- `List = [1,2,3]`

- `List*3`

- A) `[1, 2, 3, 1, 2, 3, 1, 2, 3]`
- B) `[3,6,9]`
- C) error
- D) ninguna de las anteriores

- Cual es el resultado del siguiente codigo ?

- $Lis = [9,8,7]$

- $Lis + 2$

- A) $[9,8,7,2]$

- B) $[9, 8, 7, 9, 8, 7]$

- C) error

- D) $[11,10,9]$

- Que tipo de datos es el 'numero' + |E| |

-
- A) int
 - B) float
 - C) str
 - D) ninguna de las anteriores

- ¿Cuál es el resultado esperado del siguiente código?

- `x = 0`

- `y = 1`

- `x = x ** y`

- `y = x ** y`

- `y = x ** y`

- `print(x, y)`

- A) 0 0

- B) 1 1

- C) 1 0

- D) 0 1

- F) ninguna de las anteriores

Cual es el resultado del siguiente codigo ?

```
def x():  
    return 2
```

```
x = 1 + x()  
print(x)
```

- A) 3
- B) 2
- C) error
- D) ninguna de las anteriores

Con cual de los siguientes comandos podemos insertar un elemento a una lista en python

A) `set(pos, element)`

B) `add(pos, element)`

C) `insert(pos, element)`

D) ninguna de las anteriores

¿Cuál es el resultado esperado del siguiente código?

```
def func(x):  
    if x % 2 == 0:  
        return 1  
    else:  
        return
```

```
print(func(func(2)) + 1)
```

A) 1

B) 2

C) 12

D) error

Cual es el resultado del siguiente codigo ?

```
data = ['Peter', 'Paul', 'Mary']  
print(data[int(-1 / 2)])
```

- A) Peter
- B) Paul
- C) Error
- D) ninguna de las anteriores

¿Cuál es el resultado esperado del siguiente código?

```
def func(text, num):  
    while num > 0:  
        print(text)  
        num = num - 1
```

```
func('Hello', 3)
```

- A) Hello x 3
- B) Hello x 2
- C) bucle infinito
- D Ninguna de las anteriores

| Cual es el resultado del siguiente codigo ?

```
x = 1 // 5 + 1 / 5
```

```
print(x)
```

A) 0.5

B) 0.2

C) 0.4

D) ninguna de las anteriores

¿Cuántas estrellas imprimirá el siguiente código en el monitor?

```
i = 0
while i <= 3:
    i += 2
    Print('*')
```

A) 2

B) 3

C) 5

D) ninguna de las anteriores

- El operador +=, cuando se aplica a cadenas, realiza:
-
-
- A) suma
- B) concatenacion
- D) suma de unidad
- F) ninguna de las anteriores

¿CUÁL ES EL RESULTADO ESPERADO DEL SIGUIENTE CÓDIGO?

```
X = [0, 1, 2]  
X.INSERT(0, 1)  
DEL X[1]  
PRINT(SUM(X))
```

- A) 3
- B) 4
- C) 2
- D) 5

¿CUÁL ES EL RESULTADO ESPERADO DEL SIGUIENTE CÓDIGO?

```
DEF FUN():  
    RETURN TRUE  
X = FUN(FALSE)  
PRINT(X)
```

- A) ERROR
- B) 1
- C) 0
- D) TRUE

¿Cuál será el resultado del siguiente fragmento de código?

a) `a = [1, 2, 3, 4, 5, 6, 7, 8, 9]`

b) `print(a[::2])`

A) `[1, 3, 5, 7, 9]`

B) `[1,2]`

C) `[1,2,3]`

D) Ninguna de las anteriores

▶ ¿Cuál es el resultado esperado del siguiente código?

▶ `list1 = [1, 3]`

▶ `list2 = list1`

▶ `list1[0] = 4`

▶ `print(list2)`

▶ A) `[4,3]`

▶ B) `[1,3]`

▶ C) `[1,4]`

▶ D) `[1,3,4]`

▶ ¿Cuál es el resultado del siguiente fragmento de código?

▶ `def test(x=1, y=2):`

▶ `x = x + y`

▶ `y += 1`

▶ `print(x, y)`

▶ `test(2, 1)`

▶ A) 3, 2

▶ B) 1, 3

▶ C) 2, 3

▶ D) 3, 3

▶ F) El código tiene errores