Curso Básico de Python

▼ Tabla de contenidos

Introducción a la programación con Python

Buenas prácticas

En todos los programas se debe colocar esta función para definir que a partir de esa línea de código se ejecute el programa.

- Una función se debe definir antes del código que la ejecute.
- Se debe dejar 2 espacios entre función y función.

Tu mejor herramienta: la consola

- ctrl + l = limpia la consola
- ls = lista de carpetas en el directorio actual
- cd nombreCarpeta = entro a una carpeta
- cd .. = regreso a la carpeta padre
- mkdir nombreNuevaCarpeta = crea una nueva carpeta
- touch archivo.extension = crea un archivo

Python shortcodes

- `ctrl + /` = comenta el codigo
- Se puede cancelar un ciclo infinito si presionas `Ctrl + c`

Herramientas para programar

Funciones

Una función se debe definir antes del código que la ejecute.

Conversor de monedas:

Este programa tiene:

- Una función para no repetir código.
- Tiene 2 parámetros que serán usados como variables.
- Lógica condicional.
- Código documentado con """ para que aparezca un mensaje.

```
def conversor(tipo_pesos, valor_dolar):
  pesos = float(input(f'Cuantos pesos {tipo_pesos} tienes? '))
  dolares = pesos / valor_dolar
  dolares = str(round(dolares, 2))
  print(f'Tienes ${dolares} dolares')

menu = """
Bienvenido al conversor de monedas 💰

1. Pesos colombianos
2. Pesos argentinos
3. Pesos mexicanos

Elige una opcion: """

opcion = input(menu)

if opcion == '1':
  conversor('colombianos', 3875)
elif opcion == '2':
  conversor('argentinos', 65)
elif opcion == '3':
  conversor('mexicanos', 24)
else:
  print('Ingresa una opcion correcta por favor')
```

Cadenas de texto

Hay varios métodos (funciones), que hacen cambios en los string:

```
nombre = 'tony'
>>> nombre.upper() # todo en mayúsculas
'TONY'
>>> nombre.capitalize() # primera letra en mayúsculas
'Tony'
>>> nombre = nombre.capitalize() # se debe asigar el método a la variable para que se guar
```

```
den los valores
>>> nombre = nombre.strip() # elimina los espacios extras antes o después de la string
>>> nombre = nombre.lower() # todo en minúsculas
'tony'
>>> nombre = nombre.replace('o', 'a') # reemplaza todas las 'o' por 'a'
'tany'
>>> len('mira, asi puedo contar caracteres')
33
```

Slices

variable[principio:final:pasos]

`variable[::-1]` devuelve la string asociada a la variable escrita al revés.

Proyecto: palíndromo

Escribes una palabra y te dice si es palíndromo o no:

```
def palindromo(palabra):
  palabra = palabra.replace(' ', '')
  palabra = palabra.lower()
  palabra_invertida = palabra[::-1]
  if palabra == palabra_invertida:
    return True
  else:
    return False


def run():
  palabra = input('Escribe una palabra: ')
  es_palindormo = palindromo(palabra)
  if es_palindormo == True:
    print('Es palindromo')
  else:
    print('No es palindromo')


if __name__ == '__main__':
  run()
```

Bucles

Se puede cancelar un ciclo infinito si presionas Ctrl + c.

Este código imprime las potencias de 2 hasta antes de llegar a 1M:

```
def run():
  LIMITE = 1000000
  contador = 0
  potencia_2 = 2**contador
  while potencia_2 < LIMITE:
    print(f'2 elevado a {contador} es igual a {potencia_2}')
    contador += 1
    potencia_2 = 2**contador



if __name__ == '__main__':
  run()
```

Otra forma de hacerlo usando recursividad:

```
def potencia(num,hasta):
  if num <= hasta:
    print(2 ** num)
    potencia(num+1,hasta)


if __name__=="__main__":
    hasta = int(input('Hasta qué potencia llevamos el número 2: '))
    potencia(0,hasta)
```

Ciclo for

Varias formas de imprimir los números del 1 al 1000:

```
# def contador():
#     n = 1
#     while n < 1001:
#         print(n)
#         n += 1


# a = list(range(1001))
# print(a)
```

```
# CÓDIGO FINAL
def run():
    for contador in range(1, 1001):
        print(contador)


if __name__ == '__main__':
    run()


# INICIO, FINAL, STEP DE RANGO
for i in range(0,11,2):
    print(i)
```

Recorrer un strig con for

Imprime el texto en mayúsculas:

```
def run():
    # nombre = input('Escribe tu nombre: ').capitalize()
    # for letra in nombre:
    #     print(letra)


    frase = input('Escribe una frase: ')
    for caracter in frase:
        print(caracter.upper())


if __name__ == '__main__':
    run()
```

Interrumpir un ciclo con break y continue

Continue sirve para correr el código **EXCEPTO** si pasa la condición dada a `continue`.

Break sirve para detener el código si se cumple la condición dada a `break`.

- `continue` le dice al ciclo que continúe esa parte del código sin ejecutarla y pase a la siguiente iteración.
- `break` detiene el ciclo si se cumple la funcion.

Este código no funciona, hay que comentarlo para que se quede solo 1 función:

```
def run():
    for contador in range(0,1001,2):
        print(contador)


    for contador in range(1001):
        if contador % 2 != 0:
            continue # si se cumple el if, el codigo no se ejecuta y continua a la siguien
te iteracion
        print(contador)


    for i in range(10000):
        print(i)
        if i == 56:
            break


    texto = input('Ecribe un texto: ')
    for letra in texto:
        if letra == 'o':
            break #si pones continue el texto sale sin las 'o'
        print(letra)


if __name__ == '__main__':
    run()
```

Proyecto: prueba de pimalidad

Determina si un número es o no es primo:

```
def es_primo(numero):
#     contador = 0

    for i in range(1, numero + 1):
        if i == 1 or i == numero:
            continue
        if numero % i == 0:
            contador += 1
            break
    if contador == 0:
        return True
    else:
        return False

    # if numero > 11:
    #     if numero % 2 != 0:
    #         if numero % 3 != 0:
```

```
    #             if numero % 5 != 0:
    #                 if numero % 7 != 0:
    #                     if numero % 11 != 0:
    #                         return True
    #     else:
    #         return False



def run():
    numero = int(input('Escribe un numero: '))
    if es_primo(numero):
        print('Es primo')
    else:
        print('No es primo')


if __name__ == '__main__':
    run()
```

Proyecto: videojuego

- Python tiene varias funciones comunes ya escritas que se las puede invocar con `import nombre_funcion`.

```
import random


def run():
    numero_aleatorio = random.randint(1, 100)
    numero_elegido = int(input('Elige un numero del 1 al 100: '))
    contador = 5
    intento = 'intentos'
    resultado = 'Ganaste!'
    while numero_elegido != numero_aleatorio:
        if contador == 1:
            intento = 'intento'
        if numero_elegido < numero_aleatorio:
            print(f'Busca un numero mas grande, tienes {contador} {intento}')
        else:
            print(f'Busca un numero mas pequeno, tienes {contador} {intento}')
        contador -= 1
        numero_elegido = int(input('Elige otro numero: '))
        if contador < 1:
            resultado = 'Perdiste :c'
            break
    print(resultado)
```

```
if __name__ == '__main__':
    run()
```

Estructuras de datos

Almacenar varios valores en una variable: listas

Las listas son dinámicas.

Se pueden almacenar varios tipos de elementos a una variable con este formato:

`objeto = ['objeto_1', 2, 3.4, False]`

También se les puede aplicar métodos como:

- `objeto.append(nombre_elemento)` agrega un elemento al final de la lista.
- `objeto.pop(#indice)` elimina el elemento en el índice especificado.
- Le puedes aplicar un ciclo for.

Tuplas

Son parecidas a las listas, pero ahorran memoria porque son estáticas (inmutables).

Se definan igual que las listas, pero en vez de usar `[ ]`, usa `( )`.

Diccionarios

Se definen con `{ }`, para ver sus **values** no se usa índices, sino key**s.**

```
poblacion_paises = {
        'Argentina': 44638712,
        'Brasil': 3400000,
        'Colombia': 210000000
    }

    print(poblacion_paises['Brasil'])
```

También se les puede agregar métodos:

```
for pais in poblacion_paises.keys():
        print(pais) # keys llama solo las llaves

for pais in poblacion_paises.values():
    print(pais) # values llama solo los valores

for pais, poblacion in poblacion_paises.items():
    print(pais + ' tiene ' + str(poblacion) + ' habitantes') # items muestra values & keys
```

Proyecto: generador de contraseñas

Genera contraseñas aleatorias de 15 dígitos

```
import random


def generar_contrasena():
    mayusculas = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
    minusculas = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
    simbolos = ['!', '@', '#', '$', '%', '^', '&']
    numeros = ['1', '2', '3', '4', '5', '6', '7', '8', '9']

    caracteres = mayusculas + minusculas + simbolos + numeros

    contrasena = []

    for i in range(15):
        caracter_random = random.choice(caracteres) # metodo de random para elegir un elem
ento al azar de la lista.
        contrasena.append(caracter_random)

    contrasena = ''.join(contrasena) # transforma listas en strigns.
    return contrasena


def run():
    contrasena = generar_contrasena()
    print(f'Tu nueva contrasena es: {contrasena}')


if __name__ == '__main__':
    run()
```