**Módulo 5:**

**Condiciones y Bucles**

La sintaxis básica es la siguiente:

|  |
| --- |
| while (condición):  # Código que se ejecuta **MIENTRAS** la condición es verdadera - True  Código que se ejecuta cuando la condición pasa a ser falsa - False |

En Python, la función while se utiliza para crear bucles o ciclos que se ejecutan mientras una determinada condición sea verdadera. El bloque de código dentro del while se repetirá continuamente hasta que la condición se evalúe como falsa.

|  |
| --- |
| contador = 0  while contador < 5:  print(f"El contador es: {contador}")  contador =contador + 1  print("Adios...") |

Salida esperada por consola

|  |
| --- |
| El contador es: 0  El contador es: 1  El contador es: 2  El contador es: 3  El contador es: 4  Adios... |

En el ejemplo anterior, creamos una variable contador inicializada en 0. Luego, utilizamos la función while para repetir un bloque de código mientras la condición contador < 5 sea verdadera.

Dentro del bucle while, imprimimos el valor actual del contador y luego incrementamos su valor en 1 utilizando contador = contador + 1.

Esto asegura que el bucle eventualmente terminará cuando la condición se evalúe como falsa.

En el siguiente trabajamos a la inversa

|  |
| --- |
| contador = 5  while contador !=0:  print(f"El contador es: {contador}")  contador = contador - 1  print("Adios...") |

Salida esperada por consola

|  |
| --- |
| El contador es: 5  El contador es: 4  El contador es: 3  El contador es: 2  El contador es: 1  Adios... |

.

* **break**

|  |
| --- |
| contador = 15  while contador !=0:  print(f"El contador es: {contador}")  contador = contador - 1  if contador == 5:  print("break")  break  print("Adios...") |

Salida esperada por consola

|  |
| --- |
| El contador es: 15  El contador es: 14  El contador es: 13  El contador es: 12  El contador es: 11  El contador es: 10  El contador es: 9  El contador es: 8  El contador es: 7  El contador es: 6  break  Adios... |

* **continue**·

|  |
| --- |
| contador = 15  while contador !=0:  print(f"El contador es: {contador}")  contador = contador - 1  if contador%2 == 0:  print("\t\tcontinue x par")  continue  print("\timpar")  print("Adios...") |

Salida esperada por consola

|  |
| --- |
| El contador es: 15  continue x par  El contador es: 14  impar  El contador es: 13  continue x par  El contador es: 12  impar  El contador es: 11  continue x par  El contador es: 10  impar  El contador es: 9  continue x par  El contador es: 8  impar  El contador es: 7  continue x par  El contador es: 6  impar  El contador es: 5  continue x par  El contador es: 4  impar  El contador es: 3  continue x par  El contador es: 2  impar  El contador es: 1 |

* **Anidación**

|  |
| --- |
| externo=0  while externo <10:  interno=0  while interno <10:  print (f"{externo} - {interno}", end=" | ")  interno+=1  print()  externo+=1 |

Salida esperada por consola

|  |
| --- |
| 0 - 0 | 0 - 1 | 0 - 2 | 0 - 3 | 0 - 4 | 0 - 5 | 0 - 6 | 0 - 7 | 0 - 8 | 0 - 9 |  1 - 0 | 1 - 1 | 1 - 2 | 1 - 3 | 1 - 4 | 1 - 5 | 1 - 6 | 1 - 7 | 1 - 8 | 1 - 9 |  2 - 0 | 2 - 1 | 2 - 2 | 2 - 3 | 2 - 4 | 2 - 5 | 2 - 6 | 2 - 7 | 2 - 8 | 2 - 9 |  3 - 0 | 3 - 1 | 3 - 2 | 3 - 3 | 3 - 4 | 3 - 5 | 3 - 6 | 3 - 7 | 3 - 8 | 3 - 9 |  4 - 0 | 4 - 1 | 4 - 2 | 4 - 3 | 4 - 4 | 4 - 5 | 4 - 6 | 4 - 7 | 4 - 8 | 4 - 9 |  5 - 0 | 5 - 1 | 5 - 2 | 5 - 3 | 5 - 4 | 5 - 5 | 5 - 6 | 5 - 7 | 5 - 8 | 5 - 9 |  6 - 0 | 6 - 1 | 6 - 2 | 6 - 3 | 6 - 4 | 6 - 5 | 6 - 6 | 6 - 7 | 6 - 8 | 6 - 9 |  7 - 0 | 7 - 1 | 7 - 2 | 7 - 3 | 7 - 4 | 7 - 5 | 7 - 6 | 7 - 7 | 7 - 8 | 7 - 9 |  8 - 0 | 8 - 1 | 8 - 2 | 8 - 3 | 8 - 4 | 8 - 5 | 8 - 6 | 8 - 7 | 8 - 8 | 8 - 9 |  9 - 0 | 9 - 1 | 9 - 2 | 9 - 3 | 9 - 4 | 9 - 5 | 9 - 6 | 9 - 7 | 9 - 8 | 9 - 9 | |

·

**Operadores de asignación:**

Estos operadores se utilizan para asignar valores a variables.

Estos operadores se utilizan para asignar valores a variables.

|  |  |
| --- | --- |
| Los operadores de asignación en Python son: | |
| =  +=  -=  /=  //=  %=  \*=  \*\*= | Asigna el valor de la expresión a la variable.  Suma el valor de la expresión a la variable y almacena el resultado.  Resta el valor de la expresión a la variable y almacena el resultado.  Divide el valor de la expresión a la variable y almacena el resultado.  Divide el valor entero de la expresión a la variable y almacena el resultado.  Calcula el resto de la división entre la variable y la expresión y lo almacena.  Multiplica el valor de la expresión a la variable y almacena el resultado.  Exponencia el valor de la expresión a la variable y almacena el resultado. |

·

|  |  |  |  |
| --- | --- | --- | --- |
| **Operadores de asignación de Python** | | | |
| = | x = 7 | x = 7 | 7 |
| += | x += 8 | x = x + 8 | 15 |
| -+ | x -= 10 | x = x - 10 | 5 |
| \*= | x \*= 8 | x = x \* 8 | 40 |
| /= | x /= 8 | x = x / 8 | 2 |
| \*\*= | x \*\*= 10 | x = x \*\* 10 | 1024 |
| //= | x //= 5 | x = x // 5 | 204 |
| %= | x %= 5 | x = x % 5 | 4 |
| **asignación a nivel de bits se verán por separado** | | | |

·

**Métodos de string**

En Python, los strings (cadenas de texto) son objetos inmutables que tienen una serie de métodos y atributos incorporados para manipular y trabajar con ellos. Algunos de los métodos y atributos más comunes de los strings en Python:

|  |
| --- |
| **Un input siempre devuelve un string**  **Un string es una cadena de caracteres**  **Un carácter no es solo el abecedario, sino de todo símbolo, letra, numero, etc. de los que esta dentro de las tablas unicode / utf8 según corresponde.**  **Esto significa que un string puede tenes caracteres en chino, cirílico, números, signos, etc.** |

consola:

|  |
| --- |
| string ="Ingresa su apellido: "  print (f"métodos y atributos de string :\n {**dir**(string)}") |

Salida esperada por consola

|  |
| --- |
| ['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_getstate\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'] |

|  |
| --- |
| Introducimos los métodos de string pero en vez de ir por orden alfabético (como esta en el dir)  lo hacemos mediante una división con respecto al resultado. |

<class 'str'> =

Estilos

['capitalize', 'casefold', 'lower', 'swapcase', 'title', 'upper']

|  |
| --- |
| string='HoLa MuNdO It. CuRsO pYtHoN'  print (f'{string.capitalize()=}')  print (f'{string.casefold()=}')  print (f'{string.lower()=}')  print (f'{string.swapcase()=}')  print (f'{string.title()=}')  print (f'{string.upper()=}') |

Salida esperada por consola

|  |
| --- |
| string.capitalize()='Hola mundo it. curso python'  string.casefold()='hola mundo it. curso python'  string.lower()='hola mundo it. curso python'  string.swapcase()='hOlA mUnDo iT. cUrSo PyThOn'  string.title()='Hola Mundo It. Curso Python'  string.upper()='HOLA MUNDO IT. CURSO PYTHON' |

Ubicación

['center','ljust', 'rjust', ]

|  |
| --- |
| string='datos'  print (f'|',string.center(20),'|')  print (f'|',string.ljust(20),'|')  print (f'|',string.rjust(20),'|') |

Salida esperada por consola

|  |
| --- |
| | datos |  | datos |  | datos | |

Booleanas

[ 'isalnum', 'isalpha', 'isascii', 'isidentifier', 'islower', 'isprintable', 'isspace', 'istitle', 'isupper',

'endswith','startswith',

'isdecimal', 'isdigit', 'isnumeric']

**Solo tiene una salida booleana. True o False**

|  |
| --- |
| string='Hola mundo Python 3'  print (f'{string.isalnum()=}')  print (f'{string.isalpha()=}')  print (f'{string.isascii()=}')  print (f'{string.isidentifier()=}')  print (f'{string.islower()=}')  print (f'{string.isupper()=}')  print (f'{string.istitle()=}')  print (f'{string.isprintable()=}')  print (f'{string.isspace()=}') |

Salida esperada por consola

|  |
| --- |
| string.isalnum()=False  string.isalpha()=False  string.isascii()=True  string.isidentifier()=False  string.islower()=False  string.isupper()=False  string.istitle()=False  string.isprintable()=True  string.isspace()=False |

·

|  |
| --- |
| string='\t¡¡¡¡Hola mundo Python 3.# !!!'  print (f'{string.isalnum()=}')  print (f'{string.isalpha()=}')  print (f'{string.isascii()=}')  print (f'{string.isidentifier()=}')  print (f'{string.islower()=}')  print (f'{string.isupper()=}')  print (f'{string.istitle()=}')  print (f'{string.isprintable()=}')  print (f'{string.isspace()=}') |

Salida esperada por consola

|  |
| --- |
| string.isalnum()=False  string.isalpha()=False  string.isascii()=False  string.isidentifier()=False  string.islower()=False  string.isupper()=False  string.istitle()=False  string.isprintable()=False  string.isspace()=False |

·

|  |
| --- |
| string="HOLA MUNDO IT"  print (f'{string.islower()=}')  print (f'{string.isupper()=}')  print (f'{string.istitle()=}') |

Salida esperada por consola

|  |
| --- |
| string.islower()=False  string.isupper()=True  string.istitle()=False |

·

|  |
| --- |
| string="hola mundo it"  print (f'{string.islower()=}')  print (f'{string.isupper()=}')  print (f'{string.istitle()=}') |

Salida esperada por consola

|  |
| --- |
| string.islower()=True  string.isupper()=False  string.istitle()=False |

·

|  |
| --- |
| string="Hola Mundo It"  print (f'{string.islower()=}')  print (f'{string.isupper()=}')  print (f'{string.istitle()=}') |

Salida esperada por consola

|  |
| --- |
| string.islower()=False  string.isupper()=False  string.istitle()=True |

·

|  |
| --- |
| string=" "  print (f'{string.isspace()=}')  string="Hola Python"  print (f'{string.isspace()=}') |

Salida esperada por consola

|  |
| --- |
| string.isspace()=True  string.isspace()=False |

·

|  |
| --- |
| string='Hey mundo IT'  print (f'{string.startswith( "Hola")=}')  print (f'{string.endswith( "Python")=}') |

Salida esperada por consola

|  |
| --- |
| string.startswith( "Hola")=False  string.endswith( "Python")=False |

·

|  |
| --- |
| string='Hola mundo Python'  print (f'{string.startswith( "Hola")=}')  print (f'{string.endswith( "Python")=}') |

Salida esperada por consola

|  |
| --- |
| string.startswith( "Hola")=True  string.endswith( "Python")=True |

·

|  |
| --- |
| string="UNO DOS TRES"  print (f'{string.isdecimal()=}')  print (f'{string.isdigit()=}')  print (f'{string.isnumeric()=}') |

Salida esperada por consola

|  |
| --- |
| string.isdecimal()=False  string.isdigit()=False  string.isnumeric()=False |

·

|  |
| --- |
| string="123"  print (f'{string.isdecimal()=}')  print (f'{string.isdigit()=}')  print (f'{string.isnumeric()=}') |

Salida esperada por consola

|  |
| --- |
| string.isdecimal()=True  string.isdigit()=True  string.isnumeric()=True |

·

Modificación simple

['replace']

|  |
| --- |
| string="Hola Mundo Python"  print (f'{string.replace("Hola","Grande")=}') |

Salida esperada por consola

|  |
| --- |
| string.replace("Hola","Grande")=' Grande Mundo Python' |

· Relleno

['zfill']

|  |
| --- |
| string="12345"  print (f'{string.zfill(10)=}') |

Salida esperada por consola

|  |
| --- |
| string.zfill(10)='0000012345' |

·

Para mas adelante :)

['count, 'index', 'join', 'lstrip', 'maketrans', 'partition', 'rfind', 'rindex', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'strip', 'translate', 'removeprefix', 'removesuffix']

|  |
| --- |
| optar = ""  while optar < 5:  optar = input("Ingresa una opción:  1) para Abrir  2) para Borrar  3) para Copiar  4) para Salir")  match (int(optar)):  case 1:  print (f" opción abrir")  case 2:  print (f" opción borrar")  case 3:  print (f" opción copiar")  case 4:  print (f" adios")  case other:  print (f" opción no valida") |

·

**Validación de datos : while**

·

|  |
| --- |
| while entrada != 'S':  entrada = input ('¿ desea salir ? (S/N) :').upper()  print('seguimos en el while')  print ('Salimos, la variable entrada es igual a S') |

Salida esperada por consola

|  |
| --- |
| ¿ desea salir ? (S/N) : n  seguimos en el while  ¿ desea salir ? (S/N) : s  seguimos en el while  Salimos, la variable entrada es igual a S |

·

|  |  |
| --- | --- |
| **Operadores de identidad en condicionales:** | |
| **is** | True si ambos operandos hacen referencia al mismo objeto.  False en caso contrario. |
| **is not** | True si ambos operandos NO hacen referencia al mismo objeto.  False en caso contrario. |

·

|  |
| --- |
| Registramos un objeto string para poder usar sus métodos isnumber() o isdigit() o isnumeric()  Generamos un while MIENTRAS el objeto string ingresado NO este compuesto solo por números.  Ingresamos un objeto string mediante un input dentro de un while  El bucle while se mantiene mientras la condición sea True por lo que si la niego o pido que sea falsa False obtendré una salida en el momento que ingrese un numero |

·

|  |
| --- |
| Un string es una cadena de caracteres.  Cada eslabón es un carácter.  isnumber() o isdigit() o isnumeric() evalua que cada carácter de la cadena sea un numero (0 a 9) |

·

|  |
| --- |
| Creo un objeto tipo str para poder usar sus métodos  Todo input devuelve un string  Para validar usamos algún método booleano de string  Si es necesario validar números .isdecimal() o isnumeric() o isdigit() por ahora son similares  casting de str a int |

·

|  |
| --- |
| entrada=""  while entrada.isdecimal() is False:  entrada = input ('un numero entero :')  print ('Salimos')  print (f"El tipo de dato es {type(entrada)}")  entrada = int(entrada)# casting de str a int  print (f'la variable entrada es igual a {entrada}')  print (f"El tipo de dato es {type(entrada)}") |

Salida esperada por consola

|  |
| --- |
| un numero entero : puntos 3  un numero entero : 3 puntos  un numero entero : 8  Salimos  El tipo de dato es <class 'str'>  la variable entrada es igual a 8  El tipo de dato es <class 'int'> |

·

|  |
| --- |
| Si es un numero con punto decimal  usamos replace dentro del while donde si encontramos puntos ('.') lo cambiamos por un string vacio ('')  Un string es una cadena de caracteres.  Cada eslabón es un carácter.  En un numero decimal tengo un carácter punto (.) que separa la parte entere de la decimales  pi= 3.14159  \*\*no es numero  isnumber() o isdigit() o isnumeric() evaluá que cada carácter de la cadena sea un numero (0 a 9)  pi= 3.14159  pi.replace(".","")  se lee como 314159 sin punto, Esto permite usar isnumber() o isdigit() o isnumeric() |

·

|  |
| --- |
| entrada=""  while entrada.replace(".","").isdecimal() is False:  entrada = input ('un numero flotante :')  print ('Salimos')  print (f"El tipo de dato es {type(entrada)}")  entrada = float(entrada)# casting de str a float  print (f'la variable entrada es igual a {entrada}')  print (f"El tipo de dato es {type(entrada)}") |

Salida esperada por consola

|  |
| --- |
| un numero flotante : 3 puntos  un numero flotante : 3 puntos 14159  un numero flotante : 3.14159  Salimos  El tipo de dato es <class 'str'>  la variable entrada es igual a 3.14159  El tipo de dato es <class 'float'> |

·

|  |
| --- |
| Se puede usar la misma función evaluada e formas distintas.  while entrada.replace(".","").isdecimal() **is False:**  while entrada.replace(".","").isdecimal() **is not True:**  while **not** entrada.replace(".","").isdecimal() **:**  **comprobar que el bucle mientras no se cumple deja encerrado al usuario.** |

·

|  |
| --- |
| Vale la pena mostrar que la función **while** lleva una condición como un **if** pero el bloque de información que esta indentada se repite, itera, hasta que la condición sea **True** |

·