Unidad 2: Elementos básicos del lenguaje

¿Qué es una variable?

En programación, una variable es un espacio reservado en la memoria de nuestro ordenador para almacenar un dato que puede ser usado o modificado tantas veces como se desee. Además, las variables deben tener un nombre asociado al dato que almacenan. Es decir, podemos tener una variable llamada Nombre para almacenar el nombre (dato) de una persona, una variable llamada Edad para almacenar la edad (dato), etc.

```python
#* VARIABLE

nombre_variable = "hola"
nombre_variable = True
nombre_variable = 1

print(nombre_variable)
```

¿Qué es un tipo de dato?

En programación, un tipo de dato es el atributo que especifica al ordenador la clase de dato que tiene que manejar, para saber qué valores puede tomar y qué operaciones realizar. Los tipos de datos primitivos o elementales más comunes en los lenguajes de programación son los siguientes:

- Números enteros (y con signo negativo) y coma flotante (decimales)
- Cadenas de caracteres (textos y alfanuméricos)
- Estados lógicos (booleanos) ---> *"lo revisaremos en condicionales con más detenimiento"*

Python, es un lenguaje de programación de tipado débil, lo que significa que cualquier variable puede contener cualquier tipo de dato en cualquier momento, sin especificarlo en la creación de la variable. Esto lo hace un lenguaje de programación más sencillo en los comienzos del aprendizaje de la programación.

- Números enteros (y con signo negativo) y coma flotante (decimales)

```python
#* ENTEROS ---------------------------------------------------------------------
print("\nNUMEROS ---------------------------------------------------------------------")

number_one = 16 # dia de nacimiento
number_two = 4  # mes de nacimiento

result = number_one + number_two
print("Suma",result)

result = number_one - number_two
print("Resta",result)

result = number_one * number_two
print("Multiplicacion",result)

result = number_one / number_two #Regresa un numero flotante
print("Division",result)

result = number_one ** number_two #Regresa un numero a su exponencial
print("Exponencial",result)

result = number_one // number_two #Regresa un numero entero
print("Division",result)

# Modulo, Divide el operando de la izquierda por el operador del lado derecho y devuelve el resto.
result = number_one % number_two
print("Modulo",result) #Regresa el resto.

```

- Cadenas de caracteres (textos y alfanuméricos)

```python
#* STRINGS ---------------------------------------------------------------------

print("\nSTRINGS ---------------------------------------------------------------------")

my_string = "Hola Mundo!"
print(my_string)

my_string = "Hola Mundo! 'simples'"
print(my_string)

my_string = 'Hola Mundo! "dobles"'
print(my_string,"\n")

my_string = '''Hola Mundo! 1
Hola Mundo! 2
Hola Mundo! 3
Hola Mundo! final'''
print(my_string,"\n")

my_string = '''Hola Mundo! 1\nHola Mundo! 2\nHola Mundo! 3\nHola Mundo! final\n'''
print(my_string)

course = "Python 3"
name = "Benjamin"
final_string = course + name
print(final_string, "\n")


my_string = "Curso Intensivo de Python!"
# C r u ...
# 0 1 2 ...
print(my_string)
print("Posicion [ 0]",my_string[0])
print("Posicion [ 1]",my_string[1])
print("Posicion [-1]",my_string[-1])
print("Posicion [-2]",my_string[-2], "\n")

# Función len() devuelve el tamaño de la cadena
print("len() - El tamaño de la cadena es:", len(my_string))

```

Cadenas de escape

Para escapar caracteres dentro de cadenas de caracteres se usa el carácter \ seguido de cualquier carácter ASCII.

Secuencia Escape	Significado
\newline	Ignorado
\\	Backslash (\)
\'	Comillas simple (')
\"	Comillas doble (")
\a	Bell ASCII (BEL)
\b	Backspace ASCII (BS)
\f	Formfeed ASCII (FF)
\n	Linefeed ASCII (LF)
\N{name}	Carácter llamado *name* en base de datos Unicode (Solo Unicode)
\r	Carriage Return ASCII (CR)
\t	Tabulación Horizontal ASCII (TAB)
\uxxxx	Carácter con valor hex 16-bit *xxxx* (Solamente Unicode). Ver *hex*.
\Uxxxxxxxx	Carácter con valor hex 32-bit *xxxxxxxx* (Solamente Unicode). Ver *hex*.
\v	Tabulación Vertical ASCII (VT)
\ooo	Carácter con valor octal *ooo*. Ver *octal*.
\xhh	Carácter con valor hex *hh*. Ver *hex*.

- Métodos del Objeto Strings

```
#* STRINGS - METODOS ------------------------------------------------------------------

print("\nSTRINGS - METODOS ------------------------------------------------------------------")

course = 'Curso'
my_string = 'Intensivo de Python!'

''' metodos de formato'''
result = '{a} de {b}'.format(a = course, b = my_string)
result = result.lower()
#result = result.upper()
#result = result.title()


'''metodos de busqueda'''
busqueda = result.find('curso')
count = result.count('c')
replace = result.replace('c', 'x')
split = result.split(' ')


print(result)
print("result.find('curso') = ",busqueda)
print("result.count('c') = ",count)
print("result.replace('c', 'x') = ",replace)
print("result.split(' ') = ",split)

```

Métodos	Descripción
capitalize()	Convierte el primer carácter en mayúsculas
casefold()	Convierte la cadena en minúsculas
center()	Devuelve una cadena centrada
count()	Devuelve el número de veces que se produce un valor especificado en una cadena
encode()	Devuelve una versión codificada de la cadena
endswith()	Devuelve true si la cadena termina con el valor especificado
expandtabs()	Establece el tamaño de tabulación de la cadena
find()	Busca en la cadena un valor especificado y devuelve la posición de donde se encontró
format()	Formatos de valores especificados en una cadena
format_map()	Formatos de valores especificados en una cadena
index()	Busca en la cadena un valor especificado y devuelve la posición de donde estaba Encontrado
isalnum()	Devuelve True si todos los caracteres de la cadena son alfanuméricos
isalpha()	Devuelve True si todos los caracteres de la cadena están en el alfabeto
isdecimal()	Devuelve True si todos los caracteres de la cadena son decimales
isdigit()	Devuelve True si todos los caracteres de la cadena son dígitos
isidentifier()	Devuelve True si la cadena es un identificador
islower()	Devuelve True si todos los caracteres de la cadena están en minúsculas
isnumeric()	Devuelve True si todos los caracteres de la cadena son numéricos
isprintable()	Devuelve True si todos los caracteres de la cadena se pueden imprimir
isspace()	Devuelve True si todos los caracteres de la cadena son espacios en blanco
istitle()	Devuelve True si la cadena sigue las reglas de un título
isupper()	Devuelve True si todos los caracteres de la cadena son mayúsculas
join()	Une los elementos de un iterable al final de la cadena
ljust()	Devuelve una versión justificada a la izquierda de la cadena
lower()	Convierte una cadena en minúsculas
lstrip()	Devuelve una versión de recorte izquierda de la cadena
maketrans()	Devuelve una tabla de traducción que se utilizará en las traducciones
partition()	Devuelve una tupla donde la cadena se divide en tres partes
replace()	Devuelve una cadena donde un valor especificado se reemplaza por un valor especificado
rfind()	Busca en la cadena un valor especificado y devuelve la última posición de donde se encontró
rindex()	Busca en la cadena un valor especificado y devuelve la última posición de donde se encontró
rjust()	Devuelve una versión justificada derecha de la cadena
rpartition()	Devuelve una tupla donde la cadena se divide en tres partes
rsplit()	Divide la cadena en el separador especificado y devuelve una lista
rstrip()	Devuelve una versión de recorte derecha de la cadena
split()	Divide la cadena en el separador especificado y devuelve una lista
splitlines()	Divide la cadena en saltos de línea y devuelve una lista
startswith()	Devuelve true si la cadena comienza con el valor especificado
strip()	Devuelve una versión recortada de la cadena
swapcase()	Intercambia casos, minúsculas se convierte en mayúsculas y viceversa
title()	Convierte el primer carácter de cada palabra a mayúsculas
translate()	Devuelve una cadena traducida
upper()	Convierte una cadena en mayúsculas
zfill()	Rellena la cadena con un número especificado de 0 valores al principio

Estructuras de datos (listas, tuplas ,conjuntos y diccionarios)

- Listas

```python
#* LISTAS ------------------------------------------------------------------------

print("\nLISTAS ------------------------------------------------------------------------")


my_list = ["palabra", 15, 12.9, True]
'''
   ["palabra", 15, 12.9, True ]
      (0)     (1)  (2)   (3)

   my_list[0] -> "palabra"
   my_list[1] -> 15
   my_list[2] -> 12.9
   my_list[3] -> True
'''
```

```python
my_list.append(6) # agrega el elemento al final de la lista
print(my_list)

my_list.insert(1, "insert") # agrega en la posicion que especificada
print(my_list)

my_list.remove(15) # elimina el elemento senalado
print(my_list)

last_value = my_list.pop()
print(last_value)
print(my_list, "\n")

my_list = [ 1, 9, 22, 6, 8, 65, 14, 99]
my_list_two = [ 22, 23]
print("my_list",my_list)
print("my_list_two",my_list_two, "\n")

my_list.sort() # ordena de mayor a menor la lista
print("my_list.sort() = ",my_list)

my_list.sort(reverse = True)
print("my_list.sort(reverse = True) = ",my_list, "\n")


my_list.extend(my_list_two)
print(my_list)

my_list.append(my_list_two)
print(my_list )
```

- Tuplas

```python
#* TUPLAS ------------------------------------------------------------------------

print("\nTUPLAS ------------------------------------------------------------------------")

my_tuple = (1,"palabra", True)
print(my_tuple)
print(my_tuple[0])

# my_tuple[0] = 2
# print(my_tuple[0])
'''
Traceback (most recent call last):
  File "main.py", line 153, in <module>
    my_tuple[0] = 2
TypeError: 'tuple' object does not support item assignment
'''
```

- Diccionarios

```python
#* DICCIONARIOS ----------------------------------------------------------------------

print("\nDICCIONARIOS ----------------------------------------------------------------------")

diccionario = { 'a': 55,
                5 : "esto es un string"
}
print(diccionario)


diccionario['c'] = 'un nuevo string' # agregamos o re asignamos valores
diccionario['a'] = False

valor = diccionario['a'] #obtenemos valores
print(valor)

llaves = diccionario.keys()
print(" diccionario.keys() = ", llaves)

valores = diccionario.values()
print("diccionario.values() = ",valores)

#listas puras de llaves
llaves_two = list(diccionario.keys())
print(llaves_two)

valores_two = list(diccionario.values())
print(valores_two)


llaves_three = tuple(diccionario.keys())
print(llaves_three)
valores_three = tuple(diccionario.values())
print(valores_three)

diccionario_two = {'z': 23 , 'w':555}
diccionario.update(diccionario_two)
print(diccionario)
```

Funciones

```python
#* FUNCIONES ----------------------------------------------------------------------

print("\nFUNCIONES ----------------------------------------------------------------------")

def nombre_funcion():
    pass

def suma1(a,b):
    c = a + b
    return c

def suma2(a = 1,b = 1):
    c = a + b
    return c


a = int(input("valor da a: "))
b = int(input("valor da b: "))
print(suma2(a,b))
```