In [2]:

```
#Para quitar el espacio entre las variables utilizar sep=""

print(s, 98, 29, sep="")
print(s, 98, 29)
```

```
39829
3 98 29
```

In [5]:

```
#El módulo muestra el resto de la división entre dos números

print(2%2) #Si es divisor (multipo) será 0, es decir, si el resto es cero
print(3%2) #No es divisor, contiene el número 2 una vez y le sobra 1
4%5 #No se puede dividir, pues el cociente es cero, por lo tanto sobran 4
```

```
0
1
```

Out[5]:

```
4
```

In [12]:

```
#Input recoge una entrada de texto de tipo String
edad = input("Introduce tu edad")
print(edad)
print(type(edad))
```

```
26
<class 'str'>
```

In [13]:

```
#Para recoger un número es necesario poner int
num_1 = int(input("Introduce tu edad"))
print(num_1)
print(type(num_1))
```

```
26
<class 'int'>
```

In [15]:

```
x = none

n = 5
s = "Cadena"
print (x + s)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-a62cd5d56d81> in <module>
----> 1 x = none
      2
      3 n = 5
      4 s = "Cadena"
      5 print (x + s)

NameError: name 'none' is not defined
```

In [18]:

```
#Listas y colecciones

#Lista de caracteres
```

```
s = "Cadena" #Python cuenta desde cero, por lo tanto la C es cero y la A es 1.
primer_elemento = s[0]#Esto te muestra cual es la primera posición
print(primer_elemento)
```

```
C
```

In [20]:

```
#Lista de elementos
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
message = "My first bicycle was a " + bicycles[-1] #Se utiliza menos para poner el último
elemento, en caso de ser el penúltimo se utiliza menos 2

print(message)
```

```
My first bicycle was a specialized
```

In [23]:

```
#Para saber la longitud de la lista
tamano_lista = len(bicycles)
print(tamano_lista)
ultimo_elemento_por_posicion =  tamano_lista - 1
print(ultimo_elemento_por_posicion)
```

```
4
3
```

In [24]:

```
#Existen dos tipos de funciones:
#1. Las que modifican los valores sin que tengamos que especificar una reasignacion a la
variable
#2. Los que solo devuelven la operación y no modifican el valor de la variable.

s = "String"
s.lower()
print(s.lower())
print(s) #En este caso no modifica para siempre, solo si lo llamas especificamente


cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.reverse()
print(cars) #En este caso lo modifica para siempre, es decir, se guarda el cambio al pone
r la función
```

```
string
String
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
```

In [25]:

```
#Para cambiar el orden de los caracteres
s = "Hola soy Clara"
print(s[::-1])
```

```
aralC yos aloH
```

In [27]:

```
s
```

Out[27]:

```
'Hola soy Clara'
```

In [28]:

```
s[3:]#Se incluya a partir de la 3
s[3:len(s)] #Esto es lo mismo
```

Out[28]:

```
'a soy Clara'
```

In [29]:

```
s[3:10] #El tres se incluye y la 10 no se incluye
```

Out[29]:

```
'a soy C'
```

In [35]:

```
motorcycle = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycle)

too_expensive = 'suzuki'
motorcycle.remove(too_expensive) #Para quitar un elemento
print(motorcycle)
print(too_expensive + " is too expensive for me")
```

```
['honda', 'yamaha', 'suzuki', 'ducati']
['honda', 'yamaha', 'ducati']
suzuki is too expensive for me
```

In [36]:

```
#Agrega un valor a la última posición de la lista
motorcycle.append('suzuki')
motorcycle
```

Out[36]:

```
['honda', 'yamaha', 'ducati', 'suzuki']
```

In [38]:

```
lista = ['honda', 2, 8.9, [2, 3], 'yamaha', 'suzuki', 'honda', 'honda']
lista.remove('honda') #Elimina la primera variable que se encuentra
print(lista)
```

```
[2, 8.9, [2, 3], 'yamaha', 'suzuki', 'honda', 'honda']
```

In [41]:

```
#Accedemos a la posicion 1 del elemento que está en la posición 2 de la lista
print(lista[2][1])
#Accedemos a la posición 2 del elemento que está en la tercera posición
print(lista[3][2])
```

```
3
m
```

In [58]:

```
x = (1 == 1)
x
```

Out[58]:

```
True
```

In [45]:

```
1 == 2
```

Out[45]:

```
False
```

In [50]:

```
"a" == "a"
```

Out[50]:

```
True
```

In [51]:

```
"a" != "a"
```

Out[51]:

```
False
```

In [52]:

```
4<2
```

Out[52]:

```
False
```

In [53]:

```
4>2
```

Out[53]:

```
True
```

In [54]:

```
4>=2
```

Out[54]:

```
True
```

In [55]:

```
4>=2
```

Out[55]:

```
True
```

In [ ]:

```
input ()
```

In [ ]:

```
== #Igualdad
!= #Diferente
< #Menor
> #Mayor
<= #Menor o igual
>= #Mayor o giaul
```

In [76]:

```
print((1 == 1) and (2 == 3)) #En este caso es False porque hay un False
print((1 == 1) or (2 == 3)) #or devolverá True si uno es True
print((1 == 1) or (1 == 2) and (1 == 1))
(1 == 1) or ((1 == 2) and (1 == 1))
```

```
False
True
True
```

Out[76]:

```
True
```

In [78]:

```
if 1 != 1: #Si es cierto imprime son iguales
    print("Son iguales") #Con else o if la siguiente linea empieza con tab o con cuatro e
spacios
else: #Si no es cierto imprime no entra en if
    print("No entra en if")
```

```
No entra en if
```

In [79]:

```
if 1>3:
    print("Es mayor")
elif 2 == 2: #Si entra por aquí, lo siguiente no lo mira
    print("Es igual")
elif 3 == 3:
    print("Es igual2")
else:
    print("Ninguna de las anteriores")
```

```
Es igual
```

In [81]:

```
if 2>3:
    print(1)
else:
    print("Primer else") #Bloque 1 de if
# ____________________
if 3 == 3:
    print(" 3 es 3 ") #Bloque 2 de if
 # ____________________
if 2 == 2:
    print(2)
else:
    print("Segundo else") #Bloque 3 de if
```

```
Primer else
 3 es 3
2
```

In [4]:

```
#Sinónimos de false: none, false, 0, 0.0, cualquier colección vacía
if None: #Sinónimo de False
    print("Hola")

if (not (1==1)): #Le da la vuelta, es decir, es lo contrario de lo que hay
    print("Hola")
```

In [5]:

```
if not None: #La nada actua como falsa, todo lo que no sea nada va actuar como algo, es d
ecir, True
    print("1")
```

```
1
```

In [8]:

```
if 1: #El 1 es algo, por tanto True
    print(2)
if "a":
    print(3)
```

```
2
3
```

In [10]:

```
if 0:
```

```
    print(0)
```

In [13]:

```
lista = [] #Todas las listas vacías son False
if lista:
    print(4)
```

In [12]:

```
lista = []
if not lista: #Al estar el not, le da la vuelta y lo convierte en True
    print(5)
```

```
5
```

In [14]:

```
x = True #Actua como 1
y = False

x + y
```

Out[14]:

```
1
```

In [15]:

```
x = True #Actua como 1
y = False

str(x) + str(y)
```

Out[15]:

```
'TrueFalse'
```

In [ ]:

```
#Colecciones
#1. Listas
#2. String (colección de caracteres)
#1. Tuplas
#1. Conjuntos (Set)
```

In [17]:

```
#Lista --->Mutable
lista = [2, 5, "caract", [9, "g",["j"]]]
print(lista[-1][-1][-1])
print(lista[3][2][0])
```

```
j
j
```

In [44]:

```
#Tupla --->NO mutable
tupla = (2, 5, "caract", [9, "g",["j"]]) #Diferencia en los parentesis
print(type(lista))
print(type(tupla))
```

```
<class 'list'>
<class 'tuple'>
```

In [45]:

```
tupla[3]
```

Out[45]:

```
[9, 'g', ['j']]
```

In [46]:

```
tupla[3].remove(9)
tupla
```

Out[46]:

```
(2, 5, 'caract', ['g', ['j']])
```

In [49]:

```
#Update listas --- cambiar un valor por otro (solo en listas, no en tupla)
lista = [2, "6", ["k","m"]]
lista[2] = 0
lista
```

Out[49]:

```
[2, '6', 0]
```

In [51]:

```
#Conjuntos - No permiten entrar a sus elementos, ni modificarlos. Además los conjuntos el
iminan los duplicados, es decir, que si tenemos dos números iguales elimina el primero.
conjunto2 = [2, 4, 6, "a", "z", "h"]
conjunto = set(conjunto2)
conjunto
```

Out[51]:

```
{2, 4, 6, 'a', 'h', 'z'}
```

In [54]:

```
conjunto3 = ["a", "z", "h", 2, 4, 6] #Pone los numeros primero
conjunto4 = set(conjunto3)
conjunto4
```

Out[54]:

```
{2, 4, 6, 'a', 'h', 'z'}
```

In [55]:

```
conjunto_tupla = ("a", "z", "h", 2, 4, 6, True, False) #False true despues de numeros aun
que sea tupla
conjunto = set(conjunto_tupla)
conjunto
```

Out[55]:

```
{2, 4, 6, False, True, 'a', 'h', 'z'}
```

In [ ]:

```
co
```