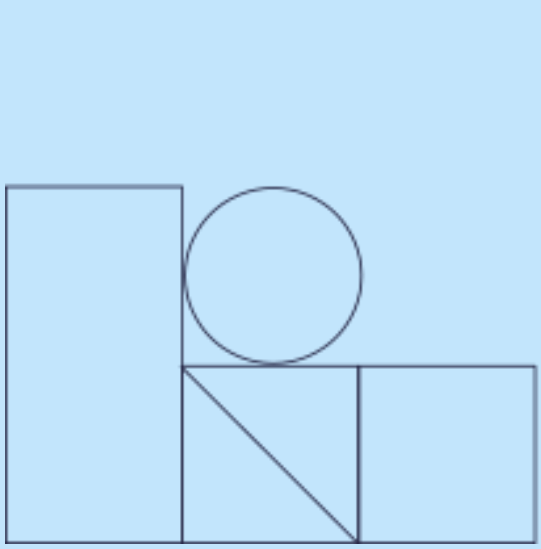
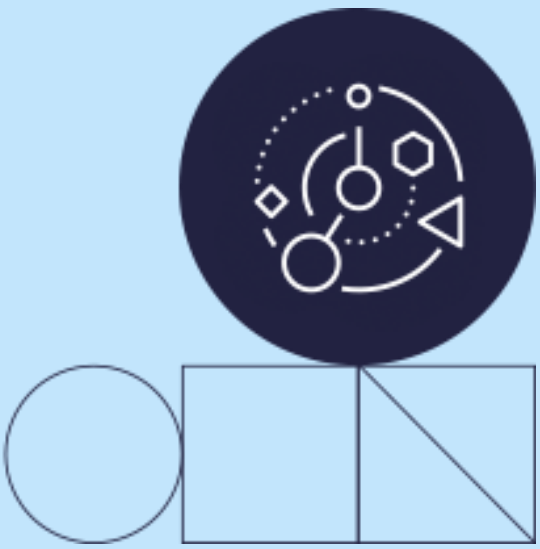
IBM SkillsBuild

IBM

# Conceptos básicos y sintaxis de Python

## Ejemplos de tipos de datos y su sintaxis





# Índice

# Introducción

El siguiente código contiene una relación de declaraciones y operaciones comunes en Python. El formato es el de un archivo **.py** totalmente funcional y ejecutable, por lo que el alumno podrá copiar el código y pegarlo en un archivo **.py** y de esa forma tenerlo como archivo de ejemplo de cómo trabajar con los tipos más comunes de Python.

# Variables

```
# VARIABLES

# Lo ideal es declara e inicializar
siempre las variables.
# ------------------------------------
------------------

# En Python podemos asignar una variable
a otra variable diferente.

var = "Hola mundo"
var2 = var
print(var2)

#la variable var y var2 apuntan a la
misma cadena de texto Hola mundo

#Los nombres de las variables en Python
pueden tener cualquier longitud y
# pueden consistir en letras mayúsculas y
minúsculas (A-Z, a-z), dígitos (0-9) y
# el carácter de subrayado (_). Cualquier
otro carácter dará error:

var& = "Hola mundo"

#Aunque el nombre de una variable puede
contener dígitos, el primer carácter de
# un nombre de variable no puede ser un
dígito.
2var = "Hola mundo"

#El nombre de las variables en Python es
sensible a mayúsculas y minúsculas
Var3 = "Hola mundo"
print(var3)              #Error, no se
encuentra var3


# Declaración de variable numérica
entera:
n_edad = 47
```

```
# Declaración de variable numérica de
coma flotante:
n_numero = -23.5245

# Declaración de variable de tipo string:
s_nombre = 'Manolo es "amigo" mío'

# Declaración de variable de tipo string
en varias líneas:
s_textoLargo = """Esto es un mensaje
...con tres saltos
...de linea"""

# Sobreescribimos el valor de la variable
s_edad y ahora la ponemos como string:
s_edad = "47"

# Declaración de constante:
NUMEROPI = 3.14159

# Declaración de un boolean
is_verdadero = True
is_casado = False

# True = 1 y False = 0
True == 1
False == 0

print(True + 2)

# Cuando se valida una condición , Python
devuelve True o False:
print(10 > 9)
print(10 == 9)
print(10 < 9)

# Declaración múltiple
# En una sola instrucción, estamos
declarando tres variables: a, b y c, y
asignándoles un valor concreto a cada
una.

a, b, c = 'string', 15, True

# Sería como poner:
a = 'string'
b = 15
c = True
```

```
# Para verificar el tipo de cualquier
objeto en Python, usamos la función
type() :

print(type(n_edad))
print(type(n_numero))
print(type(s_nombre))
print(type(NUMEROPI))
print(type(is_verdadero))
print(type(is_casado))
```

## Casting

```
# Forzado de tipo, CASTING:

# Forzado de tipo Enteros:
x = int(1)       # x Valdrá 1
y = int(2.8)     # y Valdrá 2
z = int("3")     # z Valdrá 3

# Forzado de tipo Float:
x = float(1)     # x Valdrá 1.0
y = float(2.8)   # y Valdrá 2.8
z = float("3")   # z Valdrá 3.0
w = float("4.2") # w Valdrá 4.2

# Forzado de tipo string:
x = str("s1")    # x Valdrá 's1'
y = str(2)       # y Valdrá '2'
z = str(3.0)     # z Valdrá '3.0'

# CASTING. Reconversión de tipos:
# Casting de int a float:
n_numero = 13
n_numero_2 = float(n_numero)

# Casting de float a int:
n_numero_3 = 24.876
n_numero_4 = int(n_numero_3)

# Casting de string a int
s_texto = "13"
n_numero_5 = int(s_texto)

# Casting de int a string
```

```
n_numero_6 = 27
s_texto_2 = str(n_numero_6)


print(n_numero_2)
print(type(n_numero_2))
print(n_numero_4)
print(type(n_numero_4))
print(n_numero_5)
print(type(n_numero_5))
print(s_texto_2)
print(type(s_texto_2))
```

## Comentarios

```
# COMENTARIOS

# Los comentarios son anotaciones que
pondremos en nuestro código que el
programa no va a tener en cuenta.
# Existen dos tipos de comentarios:

# Esto es un comentario de una línea

"""Esto es un comentario
que me va a ocupar
varias líneas"""
```

## Métodos de los strings

```
#   TRABAJAR CON STRINGS
"""Los strings son secuencias de
caracteres de texto.
Todos los objetos en Python se engloban
en dos categorías: mutables o inmutables.
Los tipos básicos mutables son las
listas, los diccionarios y los sets.
Los tipos básicos inmutables son los
números, los strings y las tuplas.
Los objetos mutables pueden ser cambiados
en el mismo objeto, mientras que los
inmutables no.
```

```
"""
#   Para concatenar textos se hace con
“+” o simplemente con una coma.
#   Si ponemos coma nos pone entre los
textos un espacio, con + no lo hace.

print("Esta frase" , "termina aquí.")
print("Esta frase " + "termina aquí.")

#   Contatenación y multiplicación de
strings
a = "uno"
b = "dos"
c = a + b        # c es "unodos"
c = a * 3        # c es "unounouno"


#---------------------------------------
------

#   MÉTODOS DE LOS STRINGS:

#   lower(): Convierte en minúsculas un
string.
s_texto1 = "ESTE TEXTO ESTÁ INICIALMENTE
EN MAYÚSCULAS"
print(s_texto1.lower())

#   capitalize(): Pone la primera letra
en mayúscula.
s_texto2 = "este texto no tenía la
primera letra en mayúsculas"
print(s_texto2.capitalize())


#   count(): Cuenta cuantas veces aparece
una letra o una cadena de caracteres.
s_texto3 = "Vamos a ver cuántas veces
aparece la letra c"
print(s_texto3.count('c'))

#   find(): Representa el índice o la
posición en el cual aparece un carácter o
un grupo de caracteres. Si aparece varias
veces me dice sólo el primero.
s_texto4 = "Vamos a ver en qué posición
aparece primero la letra e"
print(s_texto4.find('e'))
```

```
#   rfind(): Igual que find() pero
contando desde atrás.
s_texto5 = "Vamos a ver en qué posición
aparece la palabra desde, contando desde
atrás"
print(s_texto5.rfind('desde'))

#   isdigit(): Devuelve un boolean, nos
dice si el valor introducido es un
string. Tiene que ser un valor
introducido entre comillas o dará error.
s_texto6 = "6"
print(s_texto6.isdigit())

#   isalum(): Devuelve un boolean,
Devuelve verdadero si todos los
caracteres de la cadena son numéricos y
hay al menos un carácter. En caso
contrario, devuelve falso.
s_texto7 = "9857654gf7"
print(s_texto7.isalnum())

#   isalpha(): Devuelve un boolean,
comprueba si hay sólo letras. Los
espacios no cuentan.
s_texto8 = "Holamundo"
print(s_texto8.isalpha())

#   split(): Separa por palabras
utilizando espacios. Crea una lista.
s_texto9 = "Vamos a separar esta frase
por los espacios"
print(s_texto9.split())

#   Podemos usar otro carácter como
separador, por ejemplo una coma:
s_texto10 = "Esta sería la primera
parte,y esta la segunda"
print(s_texto10.split(","))

#   strip(): Borra los espacios sobrantes
al principio y al final.
s_texto11 = "   En este texto había
espacios al principio y al final    "
print(s_texto11.strip())

#   replace(): Cambia una palabra o una
letra por otra.
```

```
s_texto12 = "Vamos reemplazar la palabra casa"
print(s_texto12.replace("casa", "hogar"))

#   Te invito a que inspecciones el resto de funciones predefinidas para los strings en:
#   https://www.freecodecamp.org/espanol/news/metodos-de-string-de-python-explicados-con-ejemplo/
```

## Salida de datos .print()

```
#Salida de directa de datos
print("En esta ocasión hemos imprimido por pantalla este string")


#Salida de datos calculados
n_numero_1 = 4
n_numero_2 = 6
print("El resultado de sumar" , n_numero_1, "y" , n_numero_2 , "es" , (n_numero_1+n_numero_2))

#Si concatenamos int y strings usando el signo + nos puede dar problemas.
print("El resultado de sumar " + n_numero_1 + " y " + n_numero_2 + " es " + (n_numero_1+n_numero_2))

print("El resultado de sumar" + " " + "os numeros")
```

## Entrada de datos .input()

```
s_nombreIntroducido = input("Introduzca su nombre: ")

print("Bienvenido", s_nombreIntroducido)
```

```
# ------------------------------------

"""   IMPORTANTE: Todo lo introducido por input() se considera string, aunque sea un número,
por lo que, si necesitamos operar matemáticamente con números, tendremos que hacer un casting:
"""

s_edad = int(input("Introduzca su edad: "))

print("El año que viene tendrá usted ", s_edad + 1, "años")
```

## Operadores

```
#   Módulo. Nos devuelve el resto de una división:

n_numerador = 85
n_denominador = 9
n_resto = n_numerador%n_denominador
print("El resto de dividir" , n_numerador , "entre" , n_denominador , "es" , n_resto)


#   ==  Igual que...
#   No confundir con el operador de asignación =
#   Con = le damos un valor a una variable. Con == comprobamos si dos objetos son iguales.

n_numero1 = 34
s_texto1 = "34"
n_numero1 == s_texto1


n_numero2 = 34
n_numero3 = 34
n_numero2 == n_numero3
```

```
#   !=  Diferente que...

n_numero4 = 34
n_numero5 = 34
n_numero4 != n_numero5

#   +=  Suma e incremento

n_numero6 = 34
n_numero6 += 1 #Sería como poner:
n_numero6 = n_numero6 +1
print(n_numero6)
```

## Operadores lógicos

```
a = True
b = False
resultado = a and b
# print(resultado)

resultado = a or b
# print(resultado)

resultado = not a
print(resultado)

#---------------------------------
# Sintaxis simplificada para varios
operadores lógicos
edad = int(input('Introduce tu edad: '))

#veintes = edad >= 20 and edad < 30
#print(veintes)
#treintas = edad >= 30 and edad <40
#print(treintas)

if ( 20 <= edad < 30) or (30 <= edad
<40):
    print('Dentro de rango (20\'s) o
(30\'s)')
#    if veintes:
#        print('Dentro de los 20\'s')
#    elif treintas:
#        print('Dentro de los 30\'s')
#    else:
#        print('Fuera de rango')
else:
    print("No está dentro de los 20's ni
30's")
```

## Objetos mutables e inmutables

```
#   Obtener la dirección de memoria de
una variable

a = 65
print("La dirección de memoria es" ,
id(a))

#   Obtener la dirección de memoria de
una variable que apunta a otra

miNumero = 65
miNumero2 = miNumero
print("La dirección de memoria es" ,
id(miNumero))
print("La dirección de memoria es" ,
id(miNumero2))

#   Si cambio la variable, realmente creo
una copia en otra dirección de memoria:

a = 65
print("La dirección de memoria es" ,
id(a))
a+=2
print("La dirección de memoria es" ,
id(a))

#   Obtener la dirección de memoria de
una tupla
a = (1, 2, 3, 4, 5)
print("La dirección de memoria es" ,
id(a))

#   Obtener la dirección de memoria de
una lista
```

```
a = [1, 2, 3, 4, 5]
print("La dirección de memoria es" ,
id(a))

#   Obtener la dirección de memoria de un
diccionario
a = {'a': 1, 'b': 2}
print(a)
print("La dirección de memoria es" ,
id(a))

a["c"] = 3
print(a)
print("La dirección de memoria es" ,
id(a))
```