# 3_Introducción a Python

June 18, 2025

*Creado por:*

*Isabel Maniega*

## 1 -1.1- Introducción a Python (Continuación)

### 1.1 -1.1.0- Diccionarios

[1]:
```
# clave-valor
# "key": value
# {"key": value}
# {"key1": "value1", "key2": "value2",.....}
```

[2]:
```
diccionario = {"A": 10, "B": -2, "C": 35}
diccionario
```

[2]:
```
{'A': 10, 'B': -2, 'C': 35}
```

[3]:
```
diccionario["A"] # 10
```

[3]:
```
10
```

[4]:
```
diccionario["C"] # 35
```

[4]:
```
35
```

[5]:
```
diccionario["B"] # -2
```

[5]:
```
-2
```

[6]:
```
len(diccionario)
```

[6]:
```
3
```

[7]:
```
diccionario = {"clave1": 1, "clave2": 2, "clave3": 3}
diccionario
```

[7]:
```
{'clave1': 1, 'clave2': 2, 'clave3': 3}
```

[8]:
```
# Mostrar valores de claves
diccionario.keys()
```

[8]: dict_keys(['clave1', 'clave2', 'clave3'])

[9]:
```
type(diccionario.keys())
```

[9]: dict_keys

[10]:
```
# Mostrar valores de los valores
diccionario.values()
```

[10]: dict_values([1, 2, 3])

[11]:
```
# Mostrar información del diccionario
# Usada en Bucles
diccionario.items()
```

[11]: dict_items([('clave1', 1), ('clave2', 2), ('clave3', 3)])

[12]:
```
# Modificación de un valor
diccionario["clave1"] = 5
diccionario
```

[12]: {'clave1': 5, 'clave2': 2, 'clave3': 3}

[13]:
```
# Eliminacion de un campo del diccionario
del diccionario["clave3"]
diccionario
```

[13]: {'clave1': 5, 'clave2': 2}

[14]:
```
len(diccionario)
```

[14]: 2

### 1.2 Usos de tuplas/listas y diccionarios en Dataframe

[15]:
```
import pandas as pd
```

[16]:
```
# Usamos comillas dobles para los nombres de los estudiantes (E)
E = ["Andres", "Marcos", "Eva", "María"]
E
```

[16]: ['Andres', 'Marcos', 'Eva', 'María']

[17]:
```
# Notas de los exámenes (N), de 0 a 10, siendo 10 la nota más alta
N = [9, 7, 8, 6]
```

```
N
```

[17]:
```
[9, 7, 8, 6]
```

[18]:
```
data = list(zip(E, N))
data
```

[18]:
```
[('Andres', 9), ('Marcos', 7), ('Eva', 8), ('María', 6)]
```

[19]:
```
df = pd.DataFrame(data, columns=["Estudiantes", "Notas"])
df
```

[19]:

| | Estudiantes | Notas |
|---|---|---|
| 0 | Andres | 9 |
| 1 | Marcos | 7 |
| 2 | Eva | 8 |
| 3 | María | 6 |

[20]:
```
diccionario = {"Estudiantes": E, "Notas": N}
print(diccionario)
df2 = pd.DataFrame(diccionario)
df2
```

```
{'Estudiantes': ['Andres', 'Marcos', 'Eva', 'María'], 'Notas': [9, 7, 8, 6]}
```

[20]:

| | Estudiantes | Notas |
|---|---|---|
| 0 | Andres | 9 |
| 1 | Marcos | 7 |
| 2 | Eva | 8 |
| 3 | María | 6 |

[21]:
```
curso = dict(Estudiantes=E, Notas=N)
curso
```

[21]:
```
{'Estudiantes': ['Andres', 'Marcos', 'Eva', 'María'], 'Notas': [9, 7, 8, 6]}
```

### 1.3 -1.1.1- Strings

[22]:
```
s1 = "Hola, ¿Como estás?"
s1
```

[22]:
```
'Hola, ¿Como estás?'
```

[23]:
```
s1[-1], s1[17]
```

[23]:
```
('?', '?')
```

[24]:
```
len(s1)
```

[24]:
```
18
```

[25]:
```
s1[0], s1[1], s1[2], s1[3], s1[4], s1[5], s1[6], s1[7], s1[8], s1[9]
```

[25]:
```
('H', 'o', 'l', 'a', ',', ' ', '¿', 'C', 'o', 'm')
```

[26]:
```
s1[0]
```

[26]:
```
'H'
```

[27]:
```
# Ejemplo de lista para buscar palabras que empiezen por "J": ["Maria", "Juan",␣
↪"Elisa"]
# startswith = False, True (Boleano)
s1.startswith("J")
```

[27]:
```
False
```

[28]:
```
s1.startswith("H")
```

[28]:
```
True
```

[29]:
```
# Ejemplo de lista para buscar palabras que acaben por "J": ["Maria", "Juan",␣
↪"Elisa"]
# endswith = False, True (Boleano)
s1.endswith("J")
```

[29]:
```
False
```

[30]:
```
s1.endswith("?")
```

[30]:
```
True
```

### 1.4 -1.1.2- Listas con falta de valores (missing values)

[31]:
```
L = [10, -20, None, 80, -5, None, 20]
L
```

[31]:
```
[10, -20, None, 80, -5, None, 20]
```

[32]:
```
L[2] = -1
L
```

[32]:
```
[10, -20, -1, 80, -5, None, 20]
```

[33]:
```
L[-2] = -1
L
```

[33]:
```
[10, -20, -1, 80, -5, -1, 20]
```

[34]:
```
# Bucles "for" muestre en lista
for i in L:
    print(i)
```

```
10
-20
-1
80
-5
-1
20
```

[35]:
```
L = [10, -20, None, 80, -5, None, 20]
L
```

[35]:
```
[10, -20, None, 80, -5, None, 20]
```

[36]:
```
# range: limita los valores a mostrar.
# range decir empieza en posición 0 y acaba en posición 7
for i in range(0, len(L)):
    print(i)
```

```
0
1
2
3
4
5
6
```

[37]:
```
# condiciones: si esto es igual a x entonces....
# if i == None:
# sino haz esto otro = else:
```

[38]:
```
for i in range(0, len(L)):
    print(i, L[i])
```

```
0 10
1 -20
2 None
3 80
4 -5
5 None
6 20
```

[39]:
```
for i in range(0, len(L)):
    if L[i] == None:
        print(True)
    else:
```

```
        print(False)
```

```
False
False
True
False
False
True
False
```

[40]:
```
for i in range(0, len(L)):
    if L[i] == None:
        L[i] = -1

L
```

[40]:
```
[10, -20, -1, 80, -5, -1, 20]
```

### 1.4.1 Faltan valores en un Dataframe

[41]:
```
L = [10, -20, None, 80, -5, None, 20]
L
```

[41]:
```
[10, -20, None, 80, -5, None, 20]
```

[42]:
```
import pandas as pd
df = pd.DataFrame(L, columns=["Temperatura"])
df
```

[42]:

| | Temperatura |
|---|---|
| 0 | 10.0 |
| 1 | -20.0 |
| 2 | NaN |
| 3 | 80.0 |
| 4 | -5.0 |
| 5 | NaN |
| 6 | 20.0 |

[43]:
```
df.isnull()
```

[43]:

| | Temperatura |
|---|---|
| 0 | False |
| 1 | False |
| 2 | True |
| 3 | False |
| 4 | False |
| 5 | True |
| 6 | False |

[44]:
```
df.isnull().sum()
```

[44]:
```
Temperatura    2
dtype: int64
```

[45]:
```
df.describe()
```

[45]:
```
       Temperatura
count     5.000000
mean     17.000000
std      38.340579
min     -20.000000
25%      -5.000000
50%      10.000000
75%      20.000000
max      80.000000
```

[46]:
```
df
```

[46]:
```
   Temperatura
0         10.0
1        -20.0
2          NaN
3         80.0
4         -5.0
5          NaN
6         20.0
```

[47]:
```
df.Temperatura = df.Temperatura.fillna(df.Temperatura.mean())
df
```

[47]:
```
   Temperatura
0         10.0
1        -20.0
2         17.0
3         80.0
4         -5.0
5         17.0
6         20.0
```

[48]:
```
df['Humedad'] = [10, 20, 30, 40, 50, 60, 70]
df
```

[48]:
```
   Temperatura  Humedad
0         10.0       10
1        -20.0       20
2         17.0       30
3         80.0       40
```

```
4          -5.0       50
5          17.0       60
6          20.0       70
```

[49]:
```
df = df.drop(["Humedad"], axis=1)
df
```

[49]:
```
   Temperatura
0         10.0
1        -20.0
2         17.0
3         80.0
4         -5.0
5         17.0
6         20.0
```

### 1.4.2 Range de números

[50]:
```
# range(inicio, fin+1, salto)
# rango = range(1, 10) --> defecto el salto 1
rango = range(1, 10, 1)
rango
```

[50]:
```
range(1, 10)
```

[51]:
```
# imprimimos los valores del rango
for numero in rango:
    print(numero)
```

```
1
2
3
4
5
6
7
8
9
```

[52]:
```
# les almacenamos e imprimos
listado_rango = []
for numero in rango:
    listado_rango.append(numero)
    #print("Añadiendo valores: ", listado_rango)
print("Listado final: ", listado_rango)
```

```
Listado final:  [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

**np.arange: otra forma posible**

[53]:
```
import numpy as np
```

[54]:
```
# range(inicio, fin+1, salto)
# rango = range(1, 10, 1)
rango_np = np.arange(1, 10, 1)
rango_np
```

[54]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[55]:
```
# Convertir de array a lista
rango_lista = rango_np.tolist()
rango_lista
```

[55]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[56]:
```
# Aplicar saltos
# range(inicio, fin+1, salto)
num = np.arange(3, 37, 3).tolist()
num
```

[56]:
```
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36]
```

[57]:
```
# Cuando es de 1 en 1. No es necesario añadirlo:
# rango = range(1, 10, 1)
rango_np = np.arange(1, 10)
rango_np
```

[57]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[58]:
```
# si nosotros queremos que empiece 0, no es necesario indicarlo:
rango_np2 = np.arange(10)
rango_np2
```

[58]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[59]:
```
# equivale a:
# si nosotros queremos que empiece 0, no es necesario indicarlo:
rango_np2 = np.arange(0, 10)
rango_np2
```

[59]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

## 1.5 -1.2- Funciones, Clases y Objetos

### 1.5.1 Concepto de función general:

[60]:
```
# Funciones se definen:
# def nombre de la funcion():
```

[61]:
```
# 1º definir función:
def funcion_suma(x):
    y = x + 20
    return y
```

[62]:
```
funcion_suma(5)
```

[62]: 25

[63]:
```
funcion_suma(10)
```

[63]: 30

### 1.5.2 Función lambda

[64]:
```
# con funciones:
def funcion_suma_1(x):
    return x + 20
funcion_suma_1(5)
```

[64]: 25

[65]:
```
funcion_suma_1(10)
```

[65]: 30

[66]:
```
# con lambda:
(lambda x: x + 20)(5)
```

[66]: 25

[67]:
```
# con lambda:
(lambda x: x + 20)(10)
```

[67]: 30

### 1.5.3 Break (continue, pass, NO EXPLICADOS de momento)

[68]:
```
L = [2, 5, 7, 9, 12, 25, -6]
L
```

[68]: [2, 5, 7, 9, 12, 25, -6]

[69]:
```
for numero in L:
    if numero == 9:
        print("\n")
        break # Rompa el bucles: pare el bucle for
    else:
        print(numero)
print("hemos llegado al valor 9, y salió del bucle FOR")
# 2, 5, 7 Les imprime
# 9 Salta (NO IMPRIME), \n permite dejar la linea en blanco
# 12, 25, -6 NO se IMPRIMEN
```

```
2
5
7


hemos llegado al valor 9, y salió del bucle FOR
```

### 1.5.4 Clases

[70]:
```
# definir una clase pone class + nombre de la clase:
class Python:
    def funcion_imprimir(y):
        print("estamos aquí en la función imprimir: ", y)

    def funcion_suma():
        x = 2
        z = 4
        y = x + z
        print("Estamos en la función suma y la suma de y es %s" %(y))
```

[71]:
```
Python.funcion_imprimir(8)
```

```
estamos aquí en la función imprimir:  8
```

[72]:
```
Python.funcion_suma()
```

```
Estamos en la función suma y la suma de y es 6
```

## 1.6 Programación Orientada a Objetos (POO)

[73]:
```
class Empleado:
    # funciones se les llama MÉTODOS.
    # variables se les llama ATRIBUTOS
    # Init se pone con (__) dos barras bajas:
    def __init__(self, Id, Name, Age, Role):
        self.Id = Id
        self.Name = Name
```

```
        self.Age = Age
        self.Role = Role
# Instancio
# genero tantos clientes/empleados como quiera de esta forma
# empleado1 es el objeto1
empleado1 = Empleado(1, "Ana", 30, "Ingeniera")
empleado2 = Empleado(2, "Pedro", 26, "Arquitecto")
empleado3 = Empleado(3, "Maria", 54, "Abogado")
```

[74]:
```
empleado1.Age
```

[74]: 30

[75]:
```
empleado3.Role
```

[75]: 'Abogado'

### 1.6.1 Try - Except

[76]:
```
x = [1, 2, 3, 4]
x
```

[76]: [1, 2, 3, 4]

[77]:
```
try:
    print(w)
except:
    print("w no esta definida, no podemos imprimirla")
```

```
w no esta definida, no podemos imprimirla
```

[78]:
```
try:
    print(x)
except:
    print("w no esta definida, no podemos imprimirla")
```

```
[1, 2, 3, 4]
```

[79]:
```
try:
    print(w)
except Exception as e:
    print("### Error: %s" %str(e))
```

```
### Error: name 'w' is not defined
```

*Contenido creado por:*

*Isabel Maniega*