

Clase 4

Contenido

1.- Principales sentencias de control.....	1
2.- Funciones.....	2
3.- Orientación a objetos.....	4
4.- Archivos.....	8

1.- Principales sentencias de control.

if, else y elif.

La sentencia if/else funciona evaluando la condición indicada, si el resultado es True se ejecutará la siguiente sentencia o sentencias, en caso negativo se ejecutarán las sentencias que aparecen a continuación del else.



```
clase4.py > ...
1
2 x = 4
3 y = 0
4 if x == 4:
5     y = 5
6 else:
7     y = 2
8
9 print(y)
10

TERMINAL ... Python + v [ ] [ ] ... ^ x
PS D:\Cursos\Python-Codigo\Inset\programacion2> & C
:/Users/jocef/AppData/Local/Programs/Python/Python3
10/python.exe d:/Cursos/Python-Codigo/Inset/program
acion2/clase4.py
5
PS D:\Cursos\Python-Codigo\Inset\programacion2>
```

for y while.

Para iterar contamos con dos sentencias que nos ayudarán a crear bucles, nos referimos a for y a while. For aplica una serie de sentencias sobre cada uno de los elementos que contiene el objeto sobre el que aplicamos la sentencia. Python incorpora una función llamada range() que podemos utilizar para iterar sobre una serie de valores.:

Clase 4

```
clase4.py > ...
1  for x in range(1, 3):
2      print(x)
3

TERMINAL ... Python + - [ ] [ ] ... ^ x

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C
:/Users/jocef/AppData/Local/Programs/Python/Python3
10/python.exe d:/Cursos/Python-Codigo/Inset/program
acion2/clase4.py
1
2
PS D:\Cursos\Python-Codigo\Inset\programacion2> |
```

2.- Funciones.

La palabra reservada `def` nos servirá para definir una función. Seguidamente deberemos emplear un nombre y, opcionalmente, una serie de argumentos. Esta será nuestra primera función:

```
clase4.py > ...
1  def test():
2      print("test ejecutada")
3
4  test()
5

TERMINAL ... Python + - [ ] [ ] ... ^ x

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C
:/Users/jocef/AppData/Local/Programs/Python/Python3
10/python.exe d:/Cursos/Python-Codigo/Inset/program
acion2/clase4.py
test ejecutada
PS D:\Cursos\Python-Codigo\Inset\programacion2> |
```

Clase 4

Paso de parámetros.

Para ilustrar el funcionamiento, observemos el siguiente ejemplo:



```
clase4.py > ...
1 def test2 (a, b):
2     a = 2
3     b = 3
4
5 c = 5
6 d = 6
7 test2 (c, d)
8 print(f"c={c}, d={d}")
9

PROBLEMAS  SALIDA  TERMINAL  ...  Python + v [ ] [ ] ... ^ x

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/joc
ef/AppData/Local/Programs/Python/Python310/python.exe d:/Curso
s/Python-Codigo/Inset/programacion2/clase4.py
c=5, d=6
PS D:\Cursos\Python-Codigo\Inset\programacion2> [ ]
```

Como podemos observar, el valor de las variables c y d, que son inmutables, no ha sido alterado por la función.

Valores por defecto y nombres de parámetros.

En Python es posible asignar un valor a un parámetro de una función. Ejemplo:



```
clase4.py > ...
1 def fun (a, b=2023):
2     print(b)
3
4 fun(4)
5

PROBLEMAS  SALIDA  TERMINAL  ...  Python + v [ ] [ ] ... ^ x

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/joc
ef/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Py
thon-Codigo/Inset/programacion2/clase4.py
2023
PS D:\Cursos\Python-Codigo\Inset\programacion2> [ ]
```

Funciones lambda.

Al igual que otros lenguajes de programación, Python permite el uso de funciones lambda. Este tipo especial de función se caracteriza por devolver una función anónima cuando es asignada a una variable. Las funciones lambda ejecutan una determinada expresión, aceptando o no parámetros y devuelven un resultado. A

su vez, la llamada a este tipo de funciones puede ser utilizada como parámetros para otras.

```
lambda <parámetros>:<expresión>
```

```

class4.py > ...
1  lam = lambda x: x*5
2  print(lam(3))
3

```

PROBLEMAS SALIDA TERMINAL ...

Python + -

```

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocfe/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Python-Codigo/Inset/programacion2/class4.py
15
PS D:\Cursos\Python-Codigo\Inset\programacion2>

```

Podemos categorizar los lenguajes de programación existentes en dos grandes bloques:

- Orientados a procedimientos
- Orientados a objetos

Orientados a procedimientos	Orientados a objetos
La longitud del código crece linealmente o peor con la complejidad	La longitud del código crece menos que linealmente con la complejidad
La reutilización del código requiere un esfuerzo consciente del programador	La reutilización del código viene empujada por el paradigma
Difícil de entender y mantener por alguien distinto del autor o por el mismo autor si ya se olvidó lo que hizo	Gracias a que se encapsula la funcionalidad junto a las cosas es más simple de entender y por lo tanto mantener por un programador que no sea el autor original.

Prof. Jorge Valdez

Clase 4

Clases y objetos.

Crear una clase en Python es tan sencillo como emplear la sentencia `class` seguida de un nombre. Por convención, el nombre de la clase empieza en mayúscula. También suele estar contenida en un fichero del mismo nombre de la clase, pero todo en minúscula. La clase más sencilla que podemos crear en Python es la siguiente:

```
clase4.py > ...
1
2 class First:
3     def __init__(self):
4         print("Constructor ejecutado")
5
6
```

`__init__` es nuestro método constructor.

Si invocamos a la clase para crear un objeto, quedaría así:

```
clase4.py > ...
1
2 class First:
3     def __init__(self):
4         print("Constructor ejecutado")
5
6 f = First()
7
```

PROBLEMAS SALIDA TERMINAL ... Python + - [] [] ... ^ x

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocelf/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Python-Codigo/Inset/programacion2/clase4.py

Constructor ejecutado

PS D:\Cursos\Python-Codigo\Inset\programacion2> []

Variables de instancia.

Anteriormente hemos mencionado a los atributos y los hemos señalado como la correspondencia a las características de los objetos del mundo real. En la terminología de Python, a estos atributos se les denomina variables de instancia y siempre deben ir precedidos de la referencia `self`.

Veamos un ejemplo de la clase `Moto`:

```
clase4.py > ...
1
2 class Moto:
3     def __init__(self, marca, modelo, color):
4         self.marca = marca
5         self.modelo = modelo
6         self.color = color
7
```

PROBLEMAS SALIDA TERMINAL ... Python + - [] [] ... ^ x

PS D:\Cursos\Python-Codigo\Inset\programacion2> []

Clase 4

Los atributos de nuestra clase son creados al inicializar el objeto, utilizando tres variables diferentes que son pasadas como parámetros del constructor. De esta forma, la siguiente creación e inicialización de objetos sería válida:

```
1
2 class Moto:
3     def __init__(self, marca, modelo, color):
4         self.marca = marca
5         self.modelo = modelo
6         self.color = color
7
8 ktm_duke200 = Moto("KTM", "Duke 200", "naranja")
9 suzuki_gsx = Moto("Suzuki", "GSX", "negra")
10
```

Métodos de instancia.

Este tipo de métodos son aquellos que, en la práctica, definen las operaciones que pueden realizar los objetos.

Ejemplo, añadamos dos métodos de instancia a nuestra clase Moto:

```

class4.py > ...
2  class Moto:
3      def __init__(self, marca, modelo, color):
4          self.marca = marca
5          self.modelo = modelo
6          self.color = color
7
8      def get_marca(self):
9          marca = "Nueva marca"
10         print(self.marca)
11
12     def acelerar(self, km):
13         print("acelerando {} km/h".format(km))
14
15 ktm_duke200 = Moto("KTM", "Duke 200", "naranja")
16 suzuki_gsx = Moto("Suzuki", "GSX", "negra")
17
18 ktm_duke200.get_marca()
19 suzuki_gsx.acelerar(40)
20

```

PROBLEMAS SALIDA TERMINAL ...

```

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocfe/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Python-Codigo/Inset/programacion2/clase4.py
KTM
acelerando 40 km/h
PS D:\Cursos\Python-Codigo\Inset\programacion2>

```

Variables de clase.

Relacionados con los atributos, también existen en Python las variables de clase. Estas se caracterizan porque no forman parte de una instancia concreta, sino de la clase en general. Esto implica que pueden ser invocados sin necesidad de

Clase 4

hacerlo a través de una instancia. Para declararlas bastan con crear una variable justo después de la definición de la clase y fuera de cualquier método. Veamos un ejemplo creando una variable `n_ruedas`.

```
class4.py > ...
1 class Moto:
2     n_ruedas = 2
3     def __init__(self, marca, modelo, color):
4         self.marca = marca
5         self.modelo = modelo
6         self.color = color
7
8     def get_marca(self):
9         marca = "Nueva marca"
10        print(self.marca)
11
12    def acelerar(self, km):
13        print("acelerando {} km/h".format(km))
14
15    ktm_duke200 = Moto("KTM", "Duke 200", "naranja")
16    suzuki_gsx = Moto("Suzuki", "GSX", "negra")
17
18    print(Moto.n_ruedas)
19
20    print(ktm_duke200.n_ruedas)
21
```

PROBLEMAS SALIDA TERMINAL ... Python + - □ □ ... ^ x

```
PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocef
/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Py
thon-Codigo/Inset/programacion2/clase4.py
2
2
PS D:\Cursos\Python-Codigo\Inset\programacion2>
```

Métodos de clase.

también permite crear métodos de clase. Estos se caracterizan porque pueden ser invocados directamente sobre la clase, sin necesidad de crear ninguna instancia. Ejemplo:

```
class4.py > ...
24 class Test:
25     def __init__(self):
26         self.x = 8
27     @classmethod
28     def metodo_clase(cls, param1):
29         print("Parámetro: {}".format(param1))
30
31    Test.metodo_clase(6)
32
```

PROBLEMAS SALIDA TERMINAL ... Python + - □ □ ... ^ x

```
PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocef
/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Py
thon-Codigo/Inset/programacion2/clase4.py
Parámetro: 6
PS D:\Cursos\Python-Codigo\Inset\programacion2>
```

Clase 4

Métodos estáticos.

La principal diferencia con respecto a los métodos de clase es que los estáticos no necesitan ningún argumento como referencia, ni a la instancia, ni a la clase. Lógicamente, al no tener esta referencia, un método estático no puede acceder a ningún atributo de la clase. Ejemplo:

```

class4.py > ...
24 class Test:
25     def __init__(self):
26         self.x = 8
27     @classmethod
28     def metodo_clase(cls, param1):
29         print("Parámetro: {}".format(param1))
30
31     @staticmethod
32     def metodo_estatico(valor):
33         print("Valor: {}".format(valor))
34
35 t = Test()
36 t.metodo_estatico(33)
37
PROBLEMAS SALIDA TERMINAL ... Python + - [ ] [X] ... ^ X
PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocef
/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Py
thon-Codigo/Inset/programacion2/clase4.py
o Valor: 33
PS D:\Cursos\Python-Codigo\Inset\programacion2>

```

4.- Archivos.

Archivos separados por coma (CSV).

Comandos principales para trabajar con estos tipos de archivos son:

```

open
csv.reader
csv.writer
csv.DictReader
csv.DictWriter
close

```

Tomemos el siguiente archivo para nuestro ejemplo:

Entero	Flotante	Nombre	Fecha
1	1,5	XQMSA	16/10/2019
2	2,5	AWILJ	15/10/2019
3	3,5	RGWAF	14/10/2019
4	4,5	BANUR	13/10/2019
5	5,5	QEUNE	12/10/2019
6	6,5	HCHXL	11/10/2019
7	7,5	CMZTB	10/10/2019
8	8,5	OQOVI	9/10/2019
9	9,5	REYAZ	8/10/2019
10	10,5	CUSFB	7/10/2019

Clase 4

Nuestro código Python para leerlo sería:

```
clase4.py > ...
23
24 import csv
25
26 f = open("PruebaLectura.csv","r")
27
28 reader = csv.reader(f,delimiter=";")
29
30 for fila in reader:
31     print(fila)
32
33 f.close()
34
```

PROBLEMAS SALIDA TERMINAL ... Python + - [] [] ... ^ X

```
PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocef
/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Py
thon-Codigo/Inset/programacion2/clase4.py
['1', '1,5', 'XQMSA', '16/10/2022']
['2', '2,5', 'AWILI', '15/10/2022']
['3', '3,5', 'RGWAF', '14/10/2022']
['4', '4,5', 'BANUR', '13/10/2022']
['5', '5,5', 'QEUNE', '12/10/2022']
['6', '6,5', 'HCHXL', '11/10/2022']
['7', '7,5', 'CMZTB', '10/10/2022']
['8', '8,5', 'OQOVI', '9/10/2022']
['9', '9,5', 'REYAZ', '8/10/2022']
['10', '10,5', 'CUSFB', '7/10/2022']
PS D:\Cursos\Python-Codigo\Inset\programacion2> [ ]
```

Cada renglón del archivo es leído en un objeto de la clase `_csv.reader`. Cada renglón es también una colección de los valores que van tomando los campos.

Notamos que:

- Todos los campos los reconoce inicialmente como strings
- Los puntos decimales los refleja como comas. Si necesitamos convertir ese número a flotante vamos a precisar reemplazarlo por un punto.

Clase 4

Ahora escribiremos la misma información en un archivo plano:

```

class4.py > ...
24 import csv
25 ##Lectura
26 f = open("PruebaLectura.csv","r")
27
28 reader = csv.reader(f,delimiter=";")
29
30 print(type(reader))
31
32 for fila in reader:
33     print(type(fila))
34     break
35
36 ##Escritura
37 f2 = open("PruebaEscritura.csv","w")
38
39 writer = csv.writer(f2,delimiter=";",lineterminator="\n")
40
41 writer.writerows(reader)
42
43 f2.close()
44 f.close()
45
46

```

PROBLEMAS SALIDA TERMINAL ... Python + - [] [X] ^ X

```

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocef/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Python-Codigo/Inset/programacion2/clase4.py
<class '_csv.reader'>
<class 'list'>
PS D:\Cursos\Python-Codigo\Inset\programacion2>

```

Documentación completa: [Python.org](https://www.python.org)

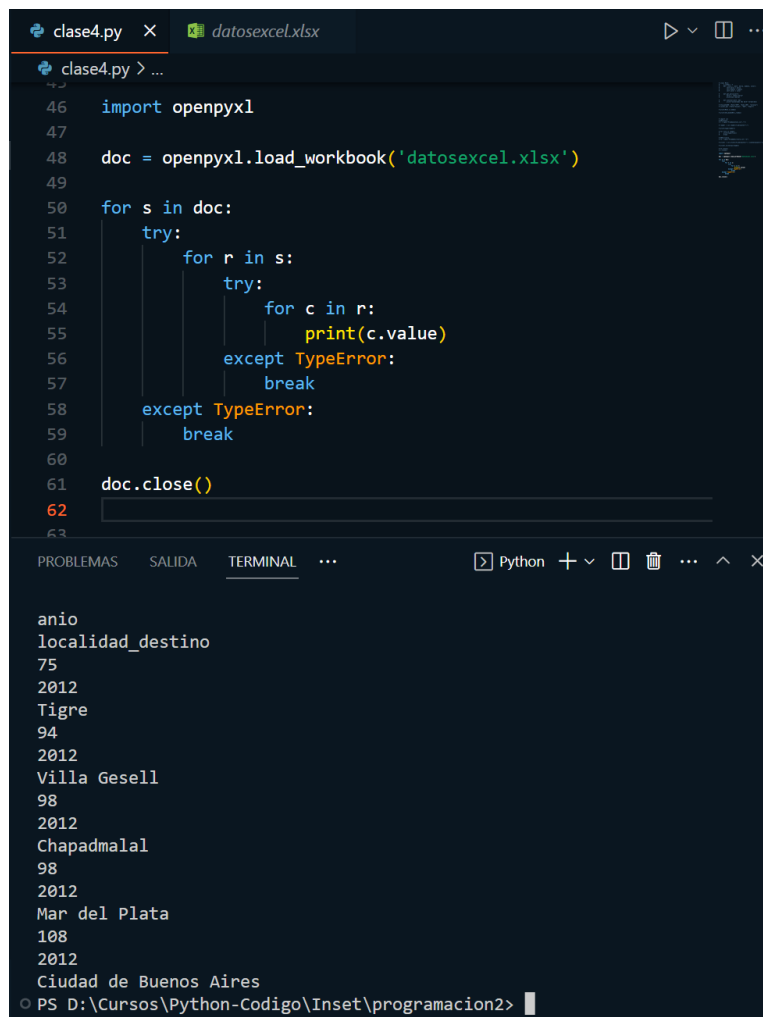
Archivos de Excel.

Para nuestro ejemplo usaremos el siguiente archivo Excel.

	A	B	C
1	id_hogar	año	localidad_destino
2	75	2,012	Tigre
3	94	2,012	Villa Gesell
4	98	2,012	Chapadmalal
5	98	2,012	Mar del Plata
6	108	2,012	Ciudad de Buenos Aires

Clase 4

Para leerlo usaremos la librería Openpyxl. Con el siguiente código:



```
clase4.py > ...
46 import openpyxl
47
48 doc = openpyxl.load_workbook('datosexcel.xlsx')
49
50 for s in doc:
51     try:
52         for r in s:
53             try:
54                 for c in r:
55                     print(c.value)
56             except TypeError:
57                 break
58         except TypeError:
59             break
60
61 doc.close()
62
```

PROBLEMAS SALIDA TERMINAL ... Python + - [] [X] ... ^ X

```
anio
localidad_destino
75
2012
Tigre
94
2012
Villa Gesell
98
2012
Chapadmalal
98
2012
Mar del Plata
108
2012
Ciudad de Buenos Aires
PS D:\Cursos\Python-Codigo\Inset\programacion2>
```

Ahora guardaremos los resultados obtenidos en un diccionario:

```

63 import openpyxl
64
65 doc = openpyxl.load_workbook('datosexcel.xlsx')
66
67 ls = []
68
69 for s in doc:
70     lr = []
71     try:
72         for r in s:
73             lc = []
74             try:
75                 for c in r:
76                     lc.append(c.value)
77             except TypeError:
78                 break
79             lr.append(lc)
80     except TypeError:
81         break
82     ls.append(lr)
83
84 doc.close()
85
86 ns = 1
87
88 for s in ls:
89     print("Hoja: ", ns)
90     for r in s:
91         print(r)
92     ns = ns+1
93

```

Para escribir un archivo en Excel, es necesario usar otra librería: `XlsxWriter`

La instalamos:

```
PS D:\Cursos\Python-Codigo\Inset\programacion> pip install XlsxWriter
Collecting XlsxWriter
  Downloading XlsxWriter-3.0.9-py3-none-any.whl (152 kB)
    152.8/152.8 kB 2.3 MB/s eta 0:00:00
Installing collected packages: XlsxWriter
Successfully installed XlsxWriter-3.0.9
PS D:\Cursos\Python-Codigo\Inset\programacion>
```

Partimos con la ventaja que ya conocemos el modelo de objetos de Excel.

El siguiente código muestra como lo haremos:

Clase 4

```

class4.py > ...
95  # Programa que genera un archivo.xlsx
96  # Importamos La librería
97  import xlswriter
98
99  # Abro el archivo - Creamos el libro
100 libro = xlswriter.Workbook("PruebaEscrituraExcel.xlsx")
101
102 # Agregamos una hoja
103 hoja = libro.add_worksheet()
104
105 # Generamos La lista de listas con los datos que queremos guardar
106 s = []
107 l = ['Entero', 'Flotante', 'Text', 'Fecha']
108 s.append(l)
109 for r in range(100):
110     l = []
111     l.append(r)
112     l.append(float(r)+.5)
113     l.append('abc'+str(r))
114     l.append('10/10/2022')
115     s.append(l)
116
117 # Finalmente, no queda grabar el texto
118 row = 0
119 col = 0
120
121 for r in s:
122     for c in r:
123         hoja.write(row, col, c)
124         col = col+1
125
126     row = row+1
127     col = 0
128
129 libro.close()
130

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL Python + v [icon] [icon] [icon] [icon] [icon] [icon] [icon]

PS D:\Cursos\Python-Codigo\Inset\programacion2> & C:/Users/jocef/AppData/Local/Programs/Python/Python310/python.exe d:/Cursos/Python-Codigo/Inset/programacion2/clase4.py

PS D:\Cursos\Python-Codigo\Inset\programacion2> [icon]

Y ahora veremos si ha guardado bien. Abrimos el archivo con Excel:

The screenshot shows an Excel spreadsheet titled 'PruebaEscrituraExcel.xlsx'. The data is organized into four columns: 'Entero' (Integer), 'Flotante' (Float), 'Text', and 'Fecha' (Date). The rows contain numerical values, floating-point numbers, text strings, and dates.

	A	B	C	D	E	F
1	Entero	Flotante	Text	Fecha		
2	0	0,5	abc0	10/10/2022		
3	1	1,5	abc1	10/10/2022		
4	2	2,5	abc2	10/10/2022		
5	3	3,5	abc3	10/10/2022		
6	4	4,5	abc4	10/10/2022		
7	5	5,5	abc5	10/10/2022		
8	6	6,5	abc6	10/10/2022		
9	7	7,5	abc7	10/10/2022		
10	8	8,5	abc8	10/10/2022		
11	9	9,5	abc9	10/10/2022		
12	10	10,5	abc10	10/10/2022		
13	11	11,5	abc11	10/10/2022		
14	12	12,5	abc12	10/10/2022		
15	13	13,5	abc13	10/10/2022		
16	14	14,5	abc14	10/10/2022		
17	15	15,5	abc15	10/10/2022		
18	16	16,5	abc16	10/10/2022		
19	17	17,5	abc17	10/10/2022		
20	18	18,5	abc18	10/10/2022		