

Repaso de Python

Agenda de la unidad:

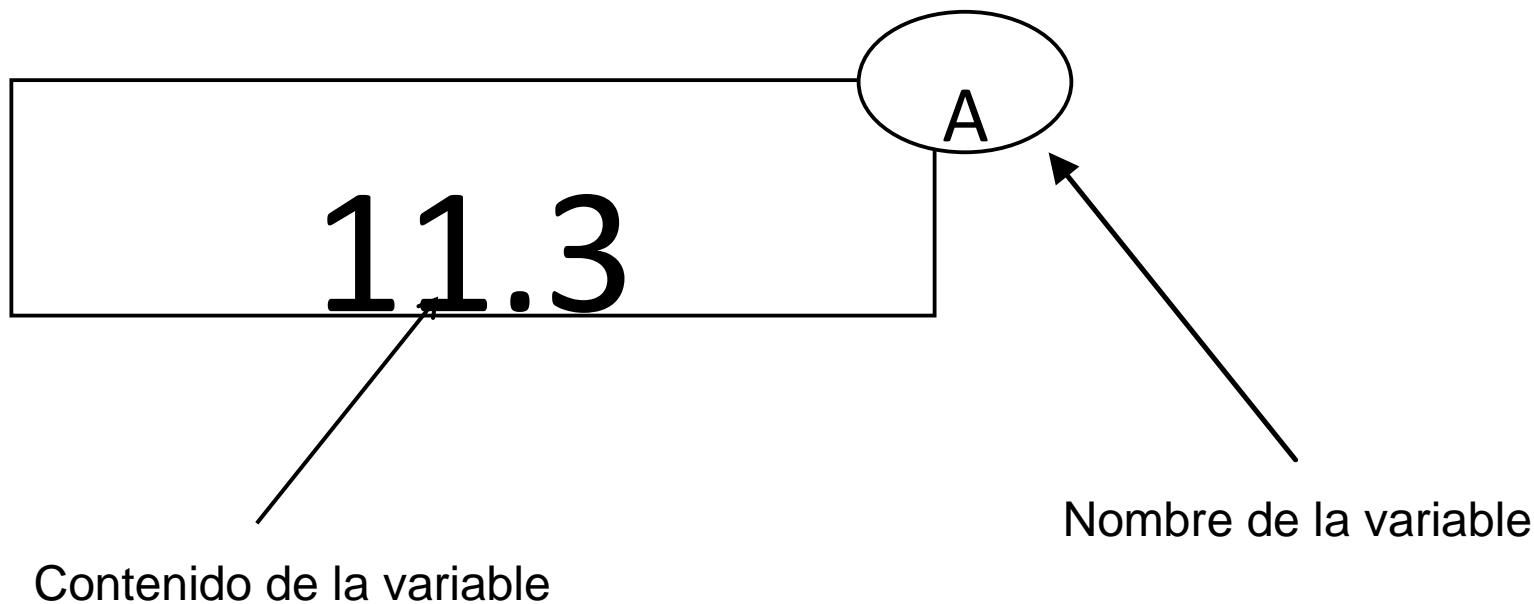
- *Introducción*
- *Manejo de archivos*
- *Objetos y herencia*
- *Archivos xlsx y csv*
- *Uso de mySQL*
- *Strings*

Introducción:

- *Tipos de datos básicos*
- *Variables*
- *I/O básico: print e input*
- *Estructuras condicionales*
- *Ciclos*
- *Funciones*
- *Ejercitación (a cada paso)*

- *Tipos de datos básicos:*
- *Numéricos:*
 - *Enteros: int*
 - *Flotantes: float*
 - *Complejos*
- *Textos*
- *Lógicas*

Variables:



Print e Input

print(valores a mostrar separados por comas)

variable = input("mensaje ")

- Aritméticos
 - Más +
 - Menos - – Multiplicación *
 - División /
 - Módulo %
 - Potencia **
 - División entera //

- Comparación
 - Igual que ==
 - Diferente de !=
 - Mayor que > – Menor que <
 - Mayor o igual que >=
 - Menor o igual que <=

- Lógicos
 - AND
 - OR
 - NOT
- Asignación
 - =

Operadores:

- Especiales
 - IS
 - IS NOT – IN

Operadores:

- Especiales
 - IS
 - IS NOT
 - IN

Los veremos más adelante

Funciones relacionadas con tipos:

int()

float()

)

type()

)

str()

Instrucciones condicionales (if)

```
if condicion:  
    bloque 1
```

Instrucciones condicionales (else)

```
if condicion:
bloque 1 else:
bloque 2
```

Instrucciones condicionales: (elif)

if condición 1:

bloque 1 elif

condición 2:

bloque 2 elif

condición 3:

...

Else:

bloque 3

Funciones

*def nombre de la función(parámetros):
cuerpo de la función
return variable con la que devuelve el
resultado*

Ciclos:

Definidos: for

Indefinidos: while

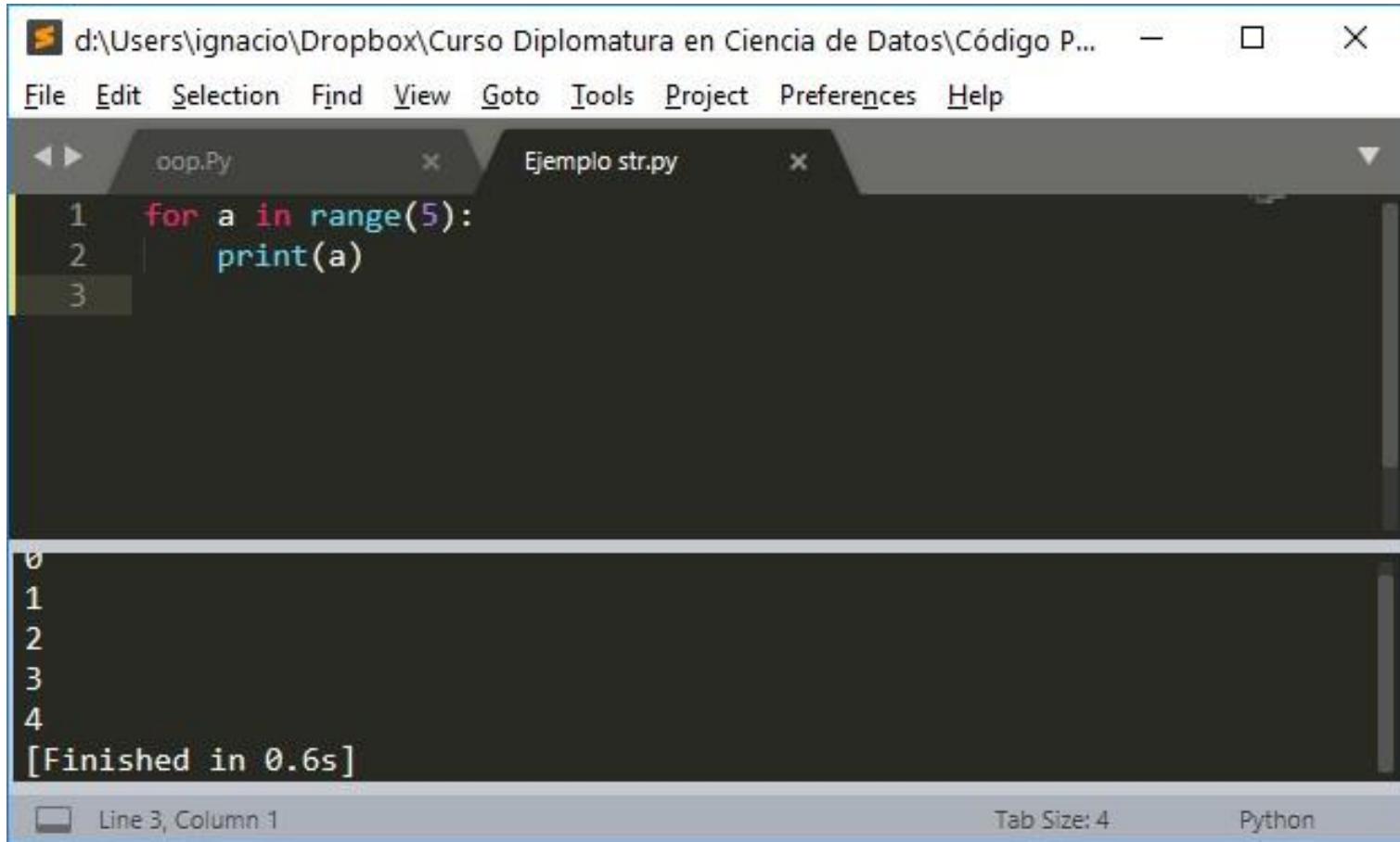
Ciclos definidos

for variable in objeto a recorrer

Por ejemplo

```
print(range(5))
```

Ejemplo:



A screenshot of a Python code editor window. The window title is "d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código P...". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. There are two tabs open: "oop.py" and "Ejemplo str.py". The "oop.py" tab contains the following code:

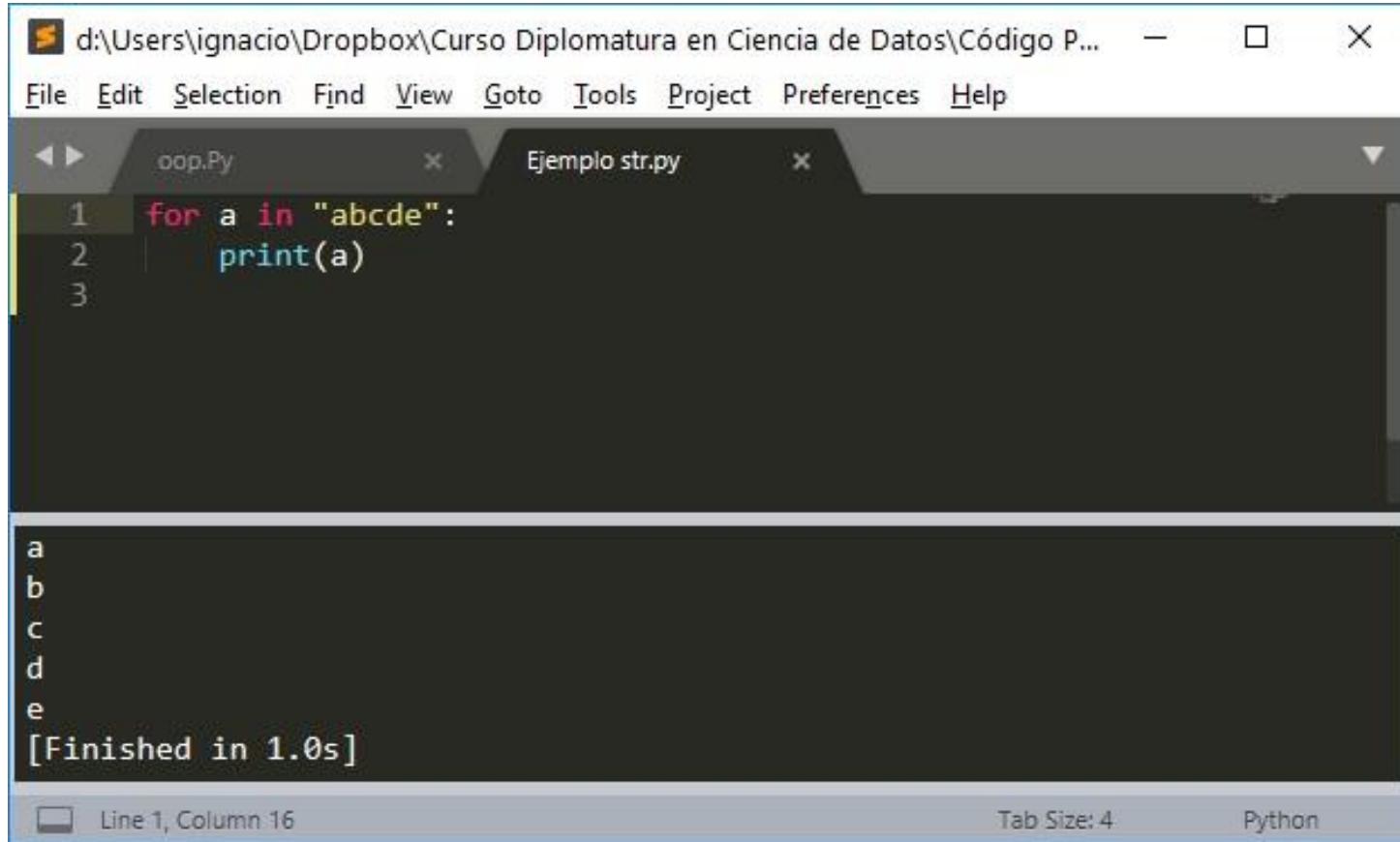
```
1 for a in range(5):
2     print(a)
3
```

The "Ejemplo str.py" tab is currently active. In the bottom right corner of the editor, there is a terminal window displaying the output of the code:

```
0
1
2
3
4
[Finished in 0.6s]
```

At the bottom of the editor window, status bars show "Line 3, Column 1", "Tab Size: 4", and "Python".

También podemos recorrer las letras de una palabra:



The screenshot shows a Python code editor window with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'Ejemplo str.py' tab is active, displaying the following code:

```
1 for a in "abcde":  
2     print(a)  
3
```

Below the code editor, the terminal output shows the letters 'a', 'b', 'c', 'd', and 'e' printed one by one. At the bottom of the terminal, the message '[Finished in 1.0s]' is displayed. The status bar at the bottom of the editor shows 'Line 1, Column 16' and 'Tab Size: 4'.

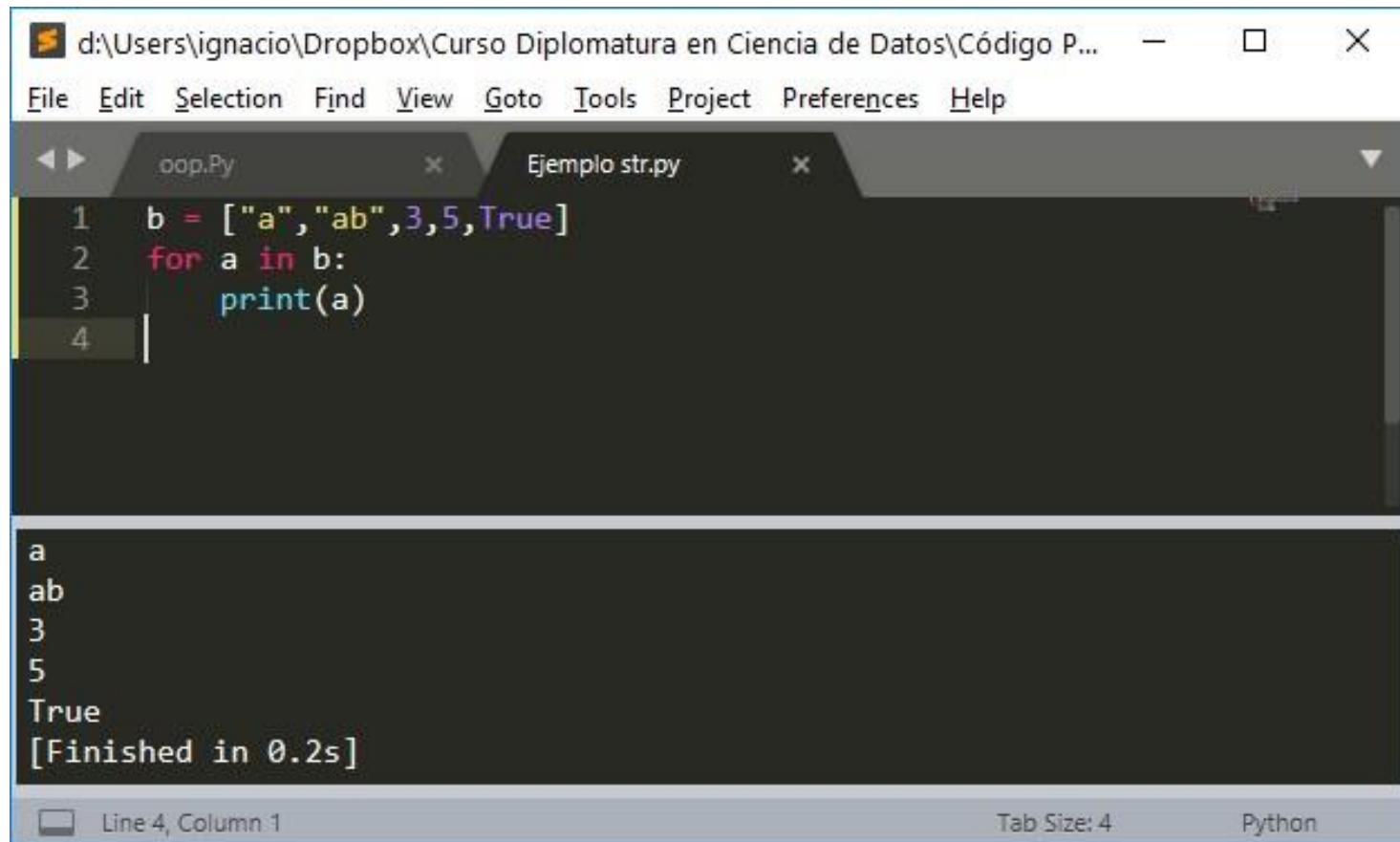
Listas:

Son colecciones de elementos

No necesitan ser uniformes en tipos

$A = [\text{elemento}1, \text{elemento}2, \dots, \text{elemento}N]$

Por ejemplo:



The screenshot shows a code editor window with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab contains the following code:

```
1 b = ["a", "ab", 3, 5, True]
2 for a in b:
3     print(a)
4 
```

The output window below shows the results of the execution:

```
a
ab
3
5
True
[Finished in 0.2s]
```

At the bottom of the editor, it says 'Line 4, Column 1' and has tabs for 'Tab Size: 4' and 'Python'.

Métodos para las listas:

append

remove

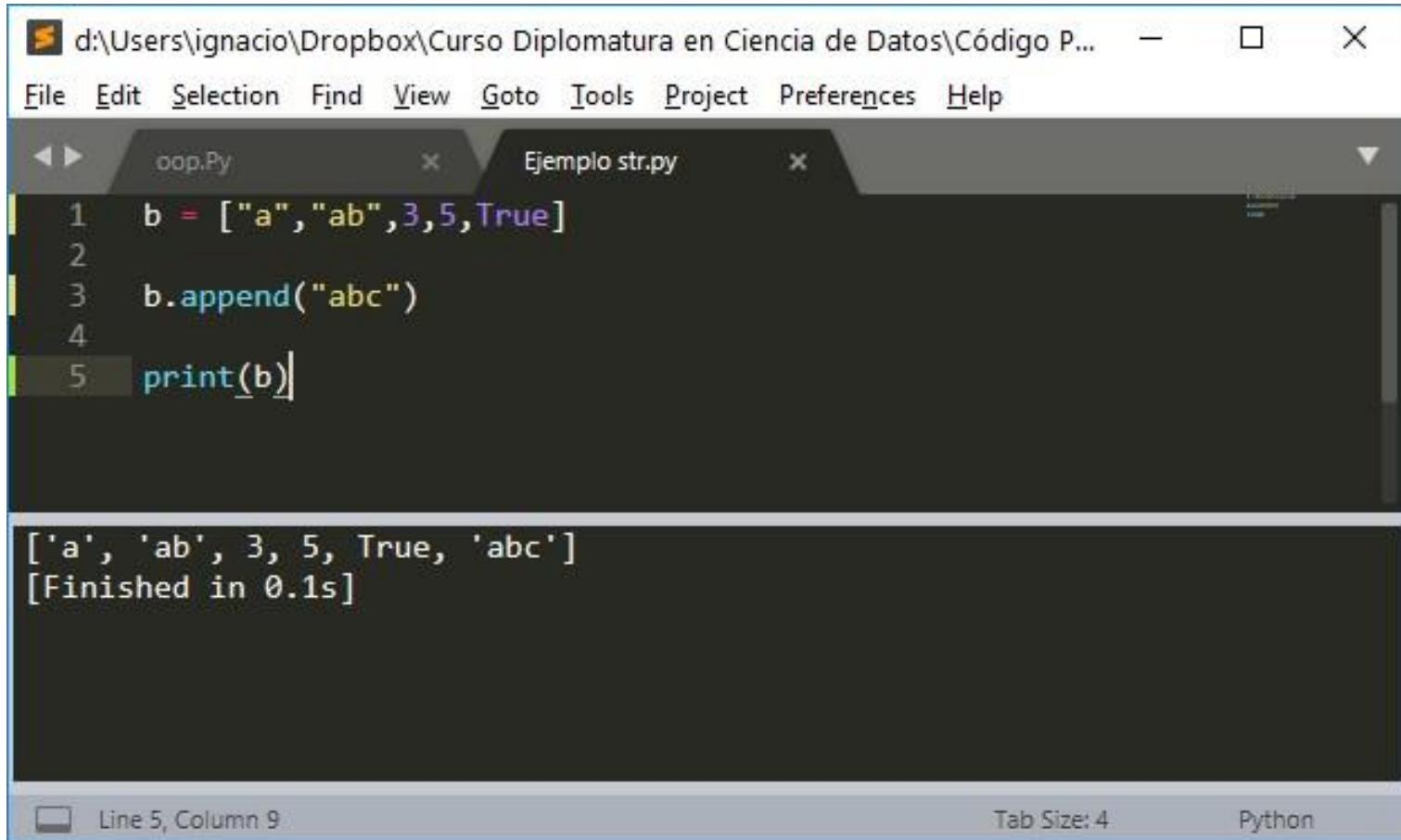
extend

in

insert(posición, elemento)

len

Uso de append:



The screenshot shows a Python code editor window with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab contains the following code:

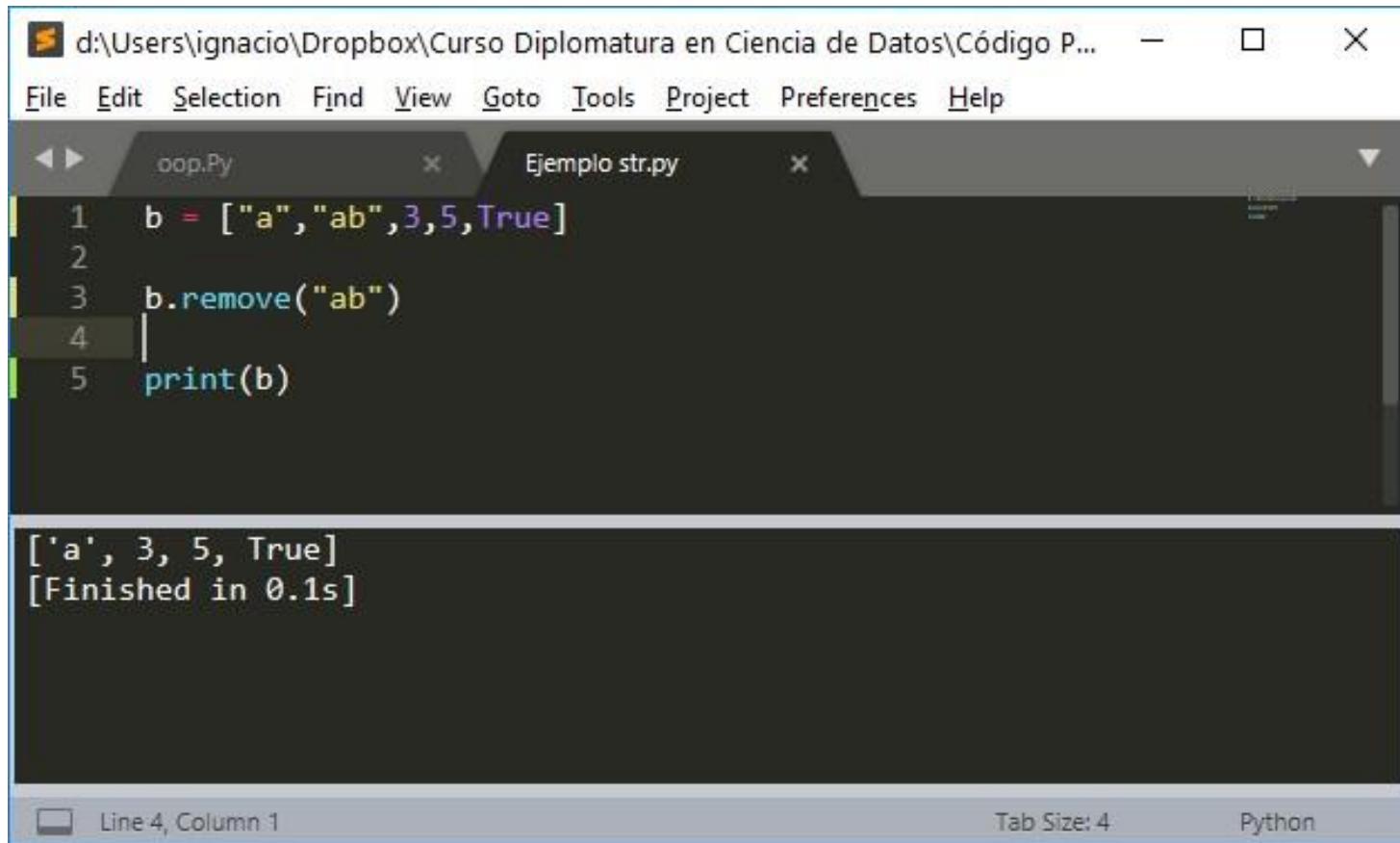
```
1 b = ["a", "ab", 3, 5, True]
2
3 b.append("abc")
4
5 print(b)
```

The output window below shows the result of running the code:

```
['a', 'ab', 3, 5, True, 'abc']
[Finished in 0.1s]
```

At the bottom, status information includes 'Line 5, Column 9', 'Tab Size: 4', and 'Python'.

Uso de remove:



The screenshot shows a Python code editor window with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab contains the following code:

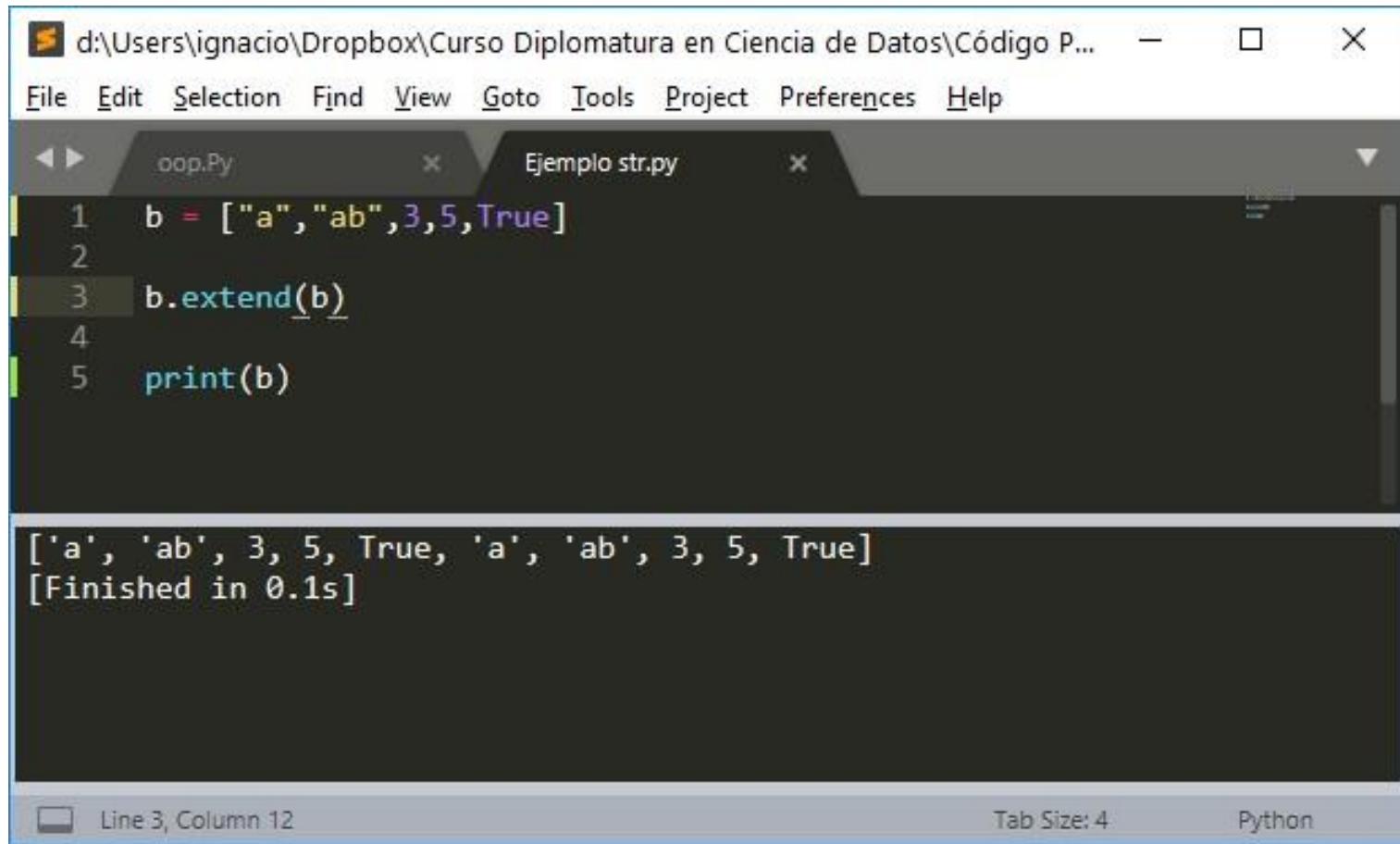
```
1 b = ["a", "ab", 3, 5, True]
2
3 b.remove("ab")
4
5 print(b)
```

The output window below shows the result of running the code:

```
['a', 3, 5, True]
[Finished in 0.1s]
```

At the bottom, status indicators show 'Line 4, Column 1', 'Tab Size: 4', and 'Python'.

Uso de extend:



The screenshot shows a Python code editor interface. At the top, there are two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab is active, displaying the following code:

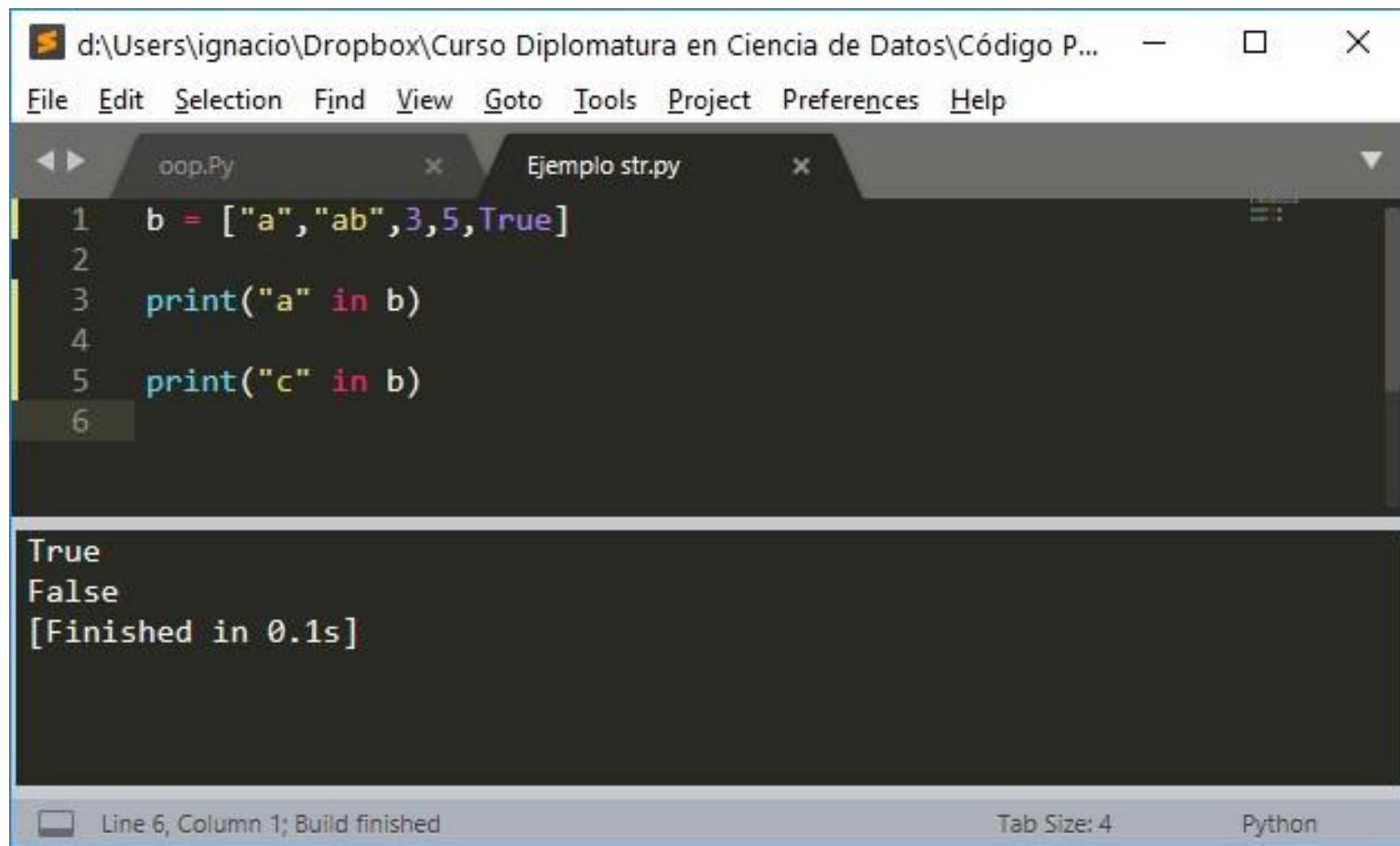
```
1 b = ["a", "ab", 3, 5, True]
2
3 b.extend(b)
4
5 print(b)
```

Below the code editor is a terminal window showing the execution results:

```
['a', 'ab', 3, 5, True, 'a', 'ab', 3, 5, True]
[Finished in 0.1s]
```

At the bottom of the terminal window, there is status information: 'Line 3, Column 12', 'Tab Size: 4', and 'Python'.

Uso de `in`:



The screenshot shows a code editor window with two tabs: "oop.py" and "Ejemplo str.py". The "Ejemplo str.py" tab is active, displaying the following Python code:

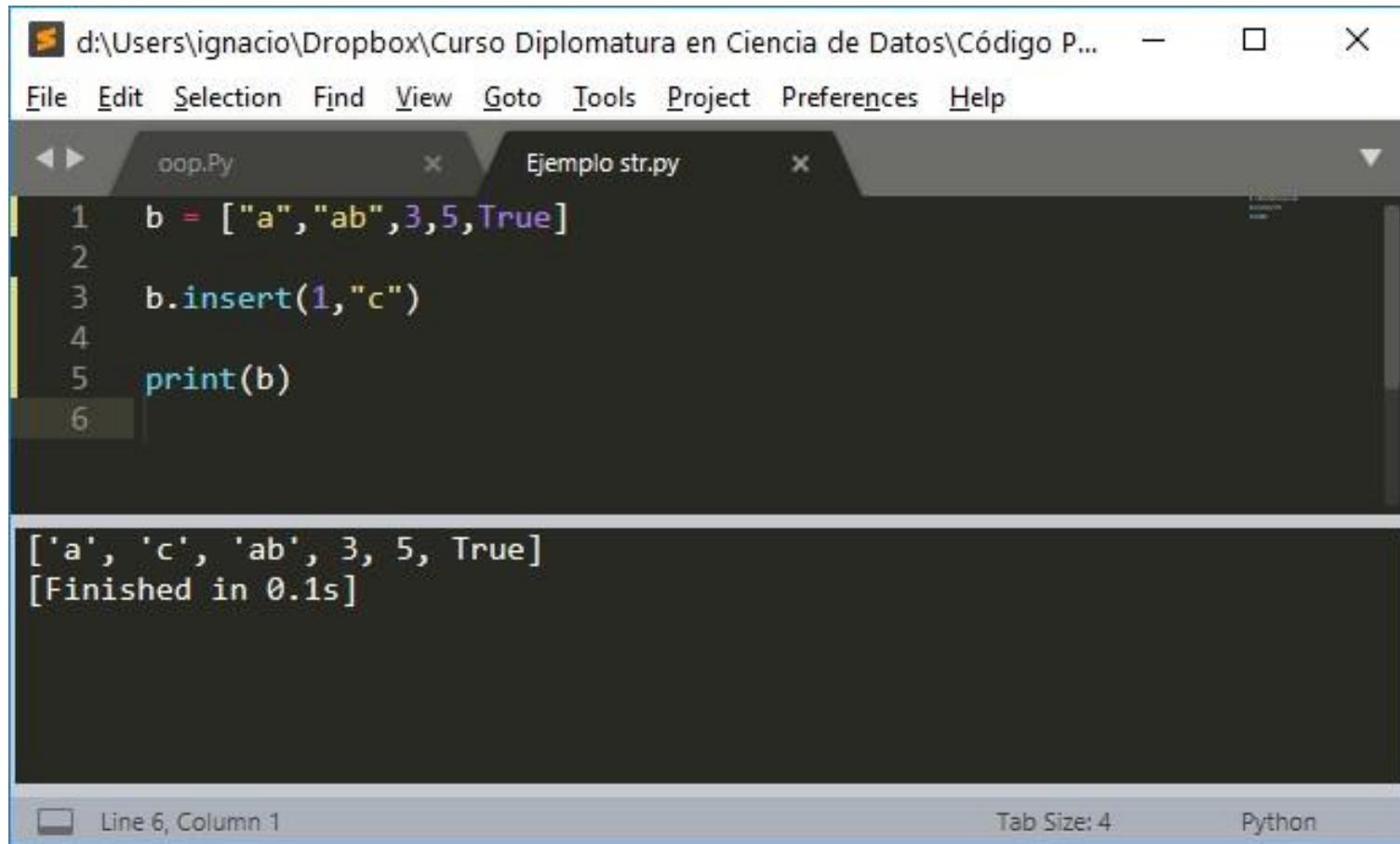
```
1 b = ["a", "ab", 3, 5, True]
2
3 print("a" in b)
4
5 print("c" in b)
6
```

The output window below the editor shows the results of running the code:

```
True
False
[Finished in 0.1s]
```

At the bottom of the editor window, status bars indicate "Line 6, Column 1, Build finished", "Tab Size: 4", and "Python".

Uso de insert:



The screenshot shows a Python code editor interface. At the top, there is a menu bar with File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. Below the menu, there are two tabs: "oop.Py" and "Ejemplo str.py". The "oop.Py" tab contains the following code:

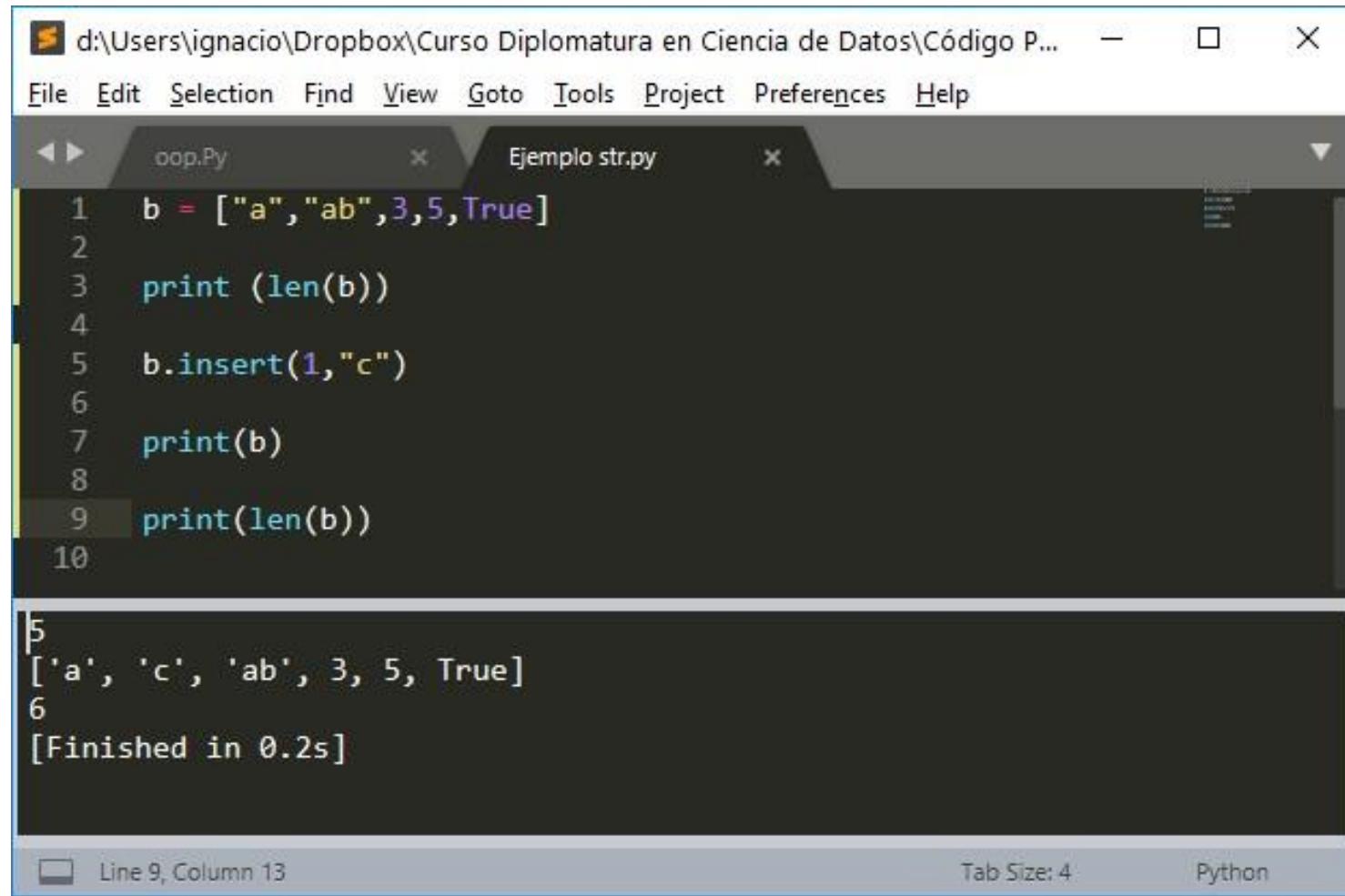
```
1 b = ["a", "ab", 3, 5, True]
2
3 b.insert(1,"c")
4
5 print(b)
6
```

When run, the output is displayed in the terminal window below:

```
['a', 'c', 'ab', 3, 5, True]
[Finished in 0.1s]
```

The status bar at the bottom indicates "Line 6, Column 1", "Tab Size: 4", and "Python".

Uso de len:



The screenshot shows a Python code editor interface with two tabs: "oop.py" and "Ejemplo str.py". The "oop.py" tab contains the following code:

```
1 b = ["a", "ab", 3, 5, True]
2
3 print(len(b))
4
5 b.insert(1, "c")
6
7 print(b)
8
9 print(len(b))
10
```

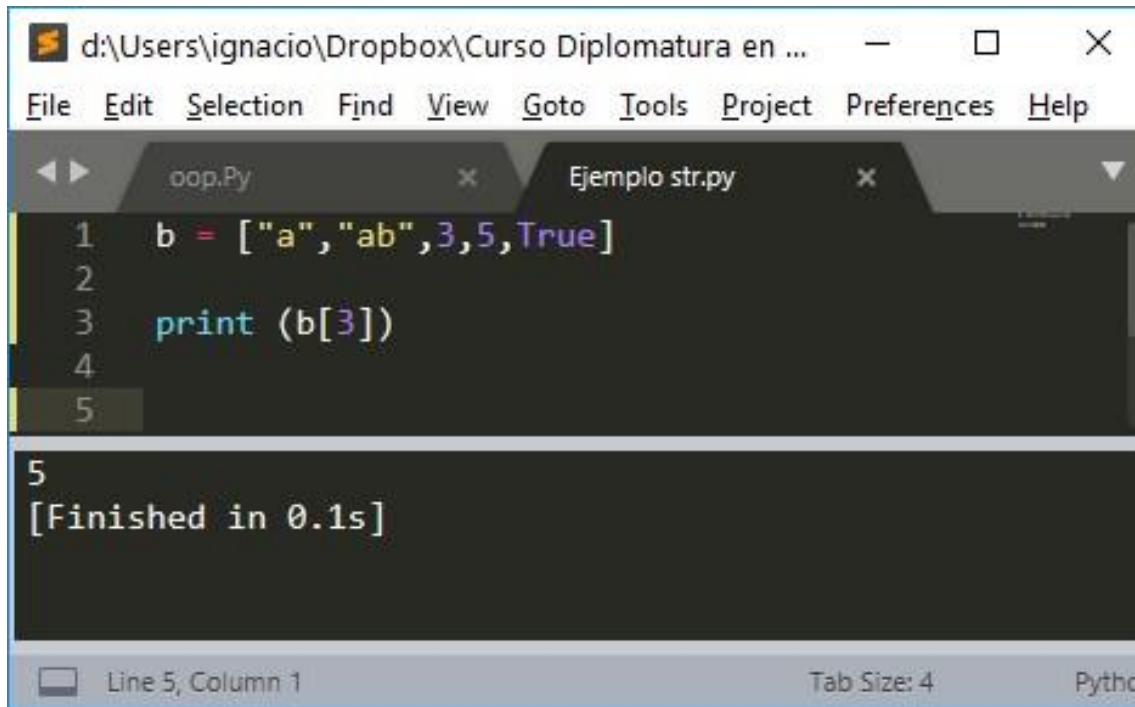
The output window below shows the results of running the code:

```
5
['a', 'c', 'ab', 3, 5, True]
6
[Finished in 0.2s]
```

At the bottom of the editor, it says "Line 9, Column 13". The status bar at the bottom right indicates "Tab Size: 4" and "Python".

Consultar un elemento de una lista:

lista[posición del elemento]



The screenshot shows a code editor window with two tabs: "oop.Py" and "Ejemplo str.py". The "Ejemplo str.py" tab is active, displaying the following Python code:

```
1 b = ["a", "ab", 3, 5, True]
2
3 print (b[3])
4
5
```

Below the code, the output window shows:

```
5
[Finished in 0.1s]
```

At the bottom of the editor, status bars indicate "Line 5, Column 1", "Tab Size: 4", and "Python".

Tuplas:

Son como las listas

Son estáticas: no se pueden modificar

Son más rápidas

Ocupan menos lugar

Usamos () en vez de []

Métodos para las tuplas:

index

x in

len

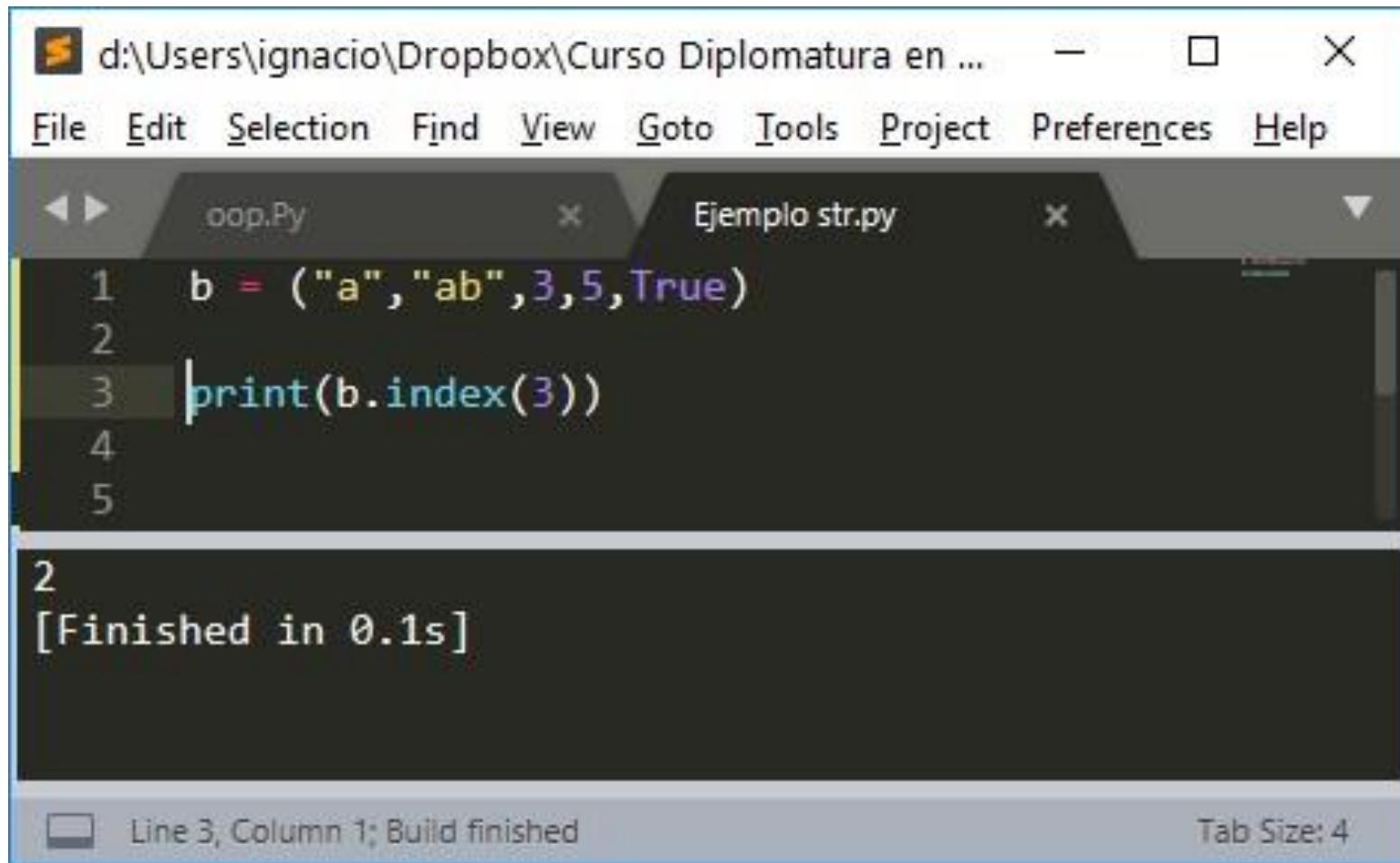
list

tuple

Uso

*de
index*

:



The screenshot shows a code editor window with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab is active, displaying the following code:

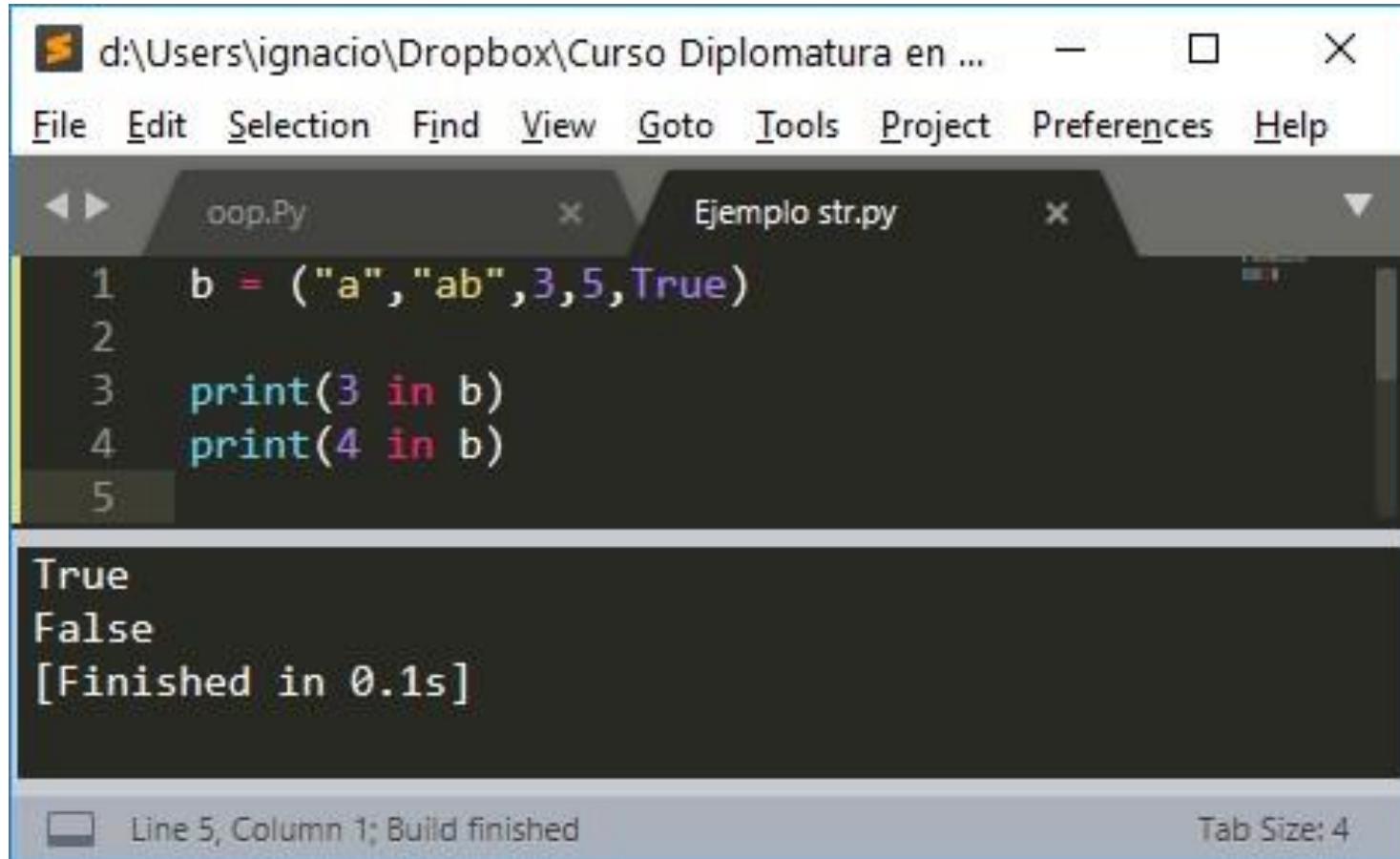
```
1 b = ("a","ab",3,5,True)
2
3 print(b.index(3))
4
5
```

The output window below shows the result of running the code:

```
2
[Finished in 0.1s]
```

At the bottom left, there is a message: "Line 3, Column 1: Build finished". At the bottom right, it says "Tab Size: 4".

Uso de `in:`



The screenshot shows a Python code editor window with two tabs: "oop.py" and "Ejemplo str.py". The "Ejemplo str.py" tab is active, displaying the following code:

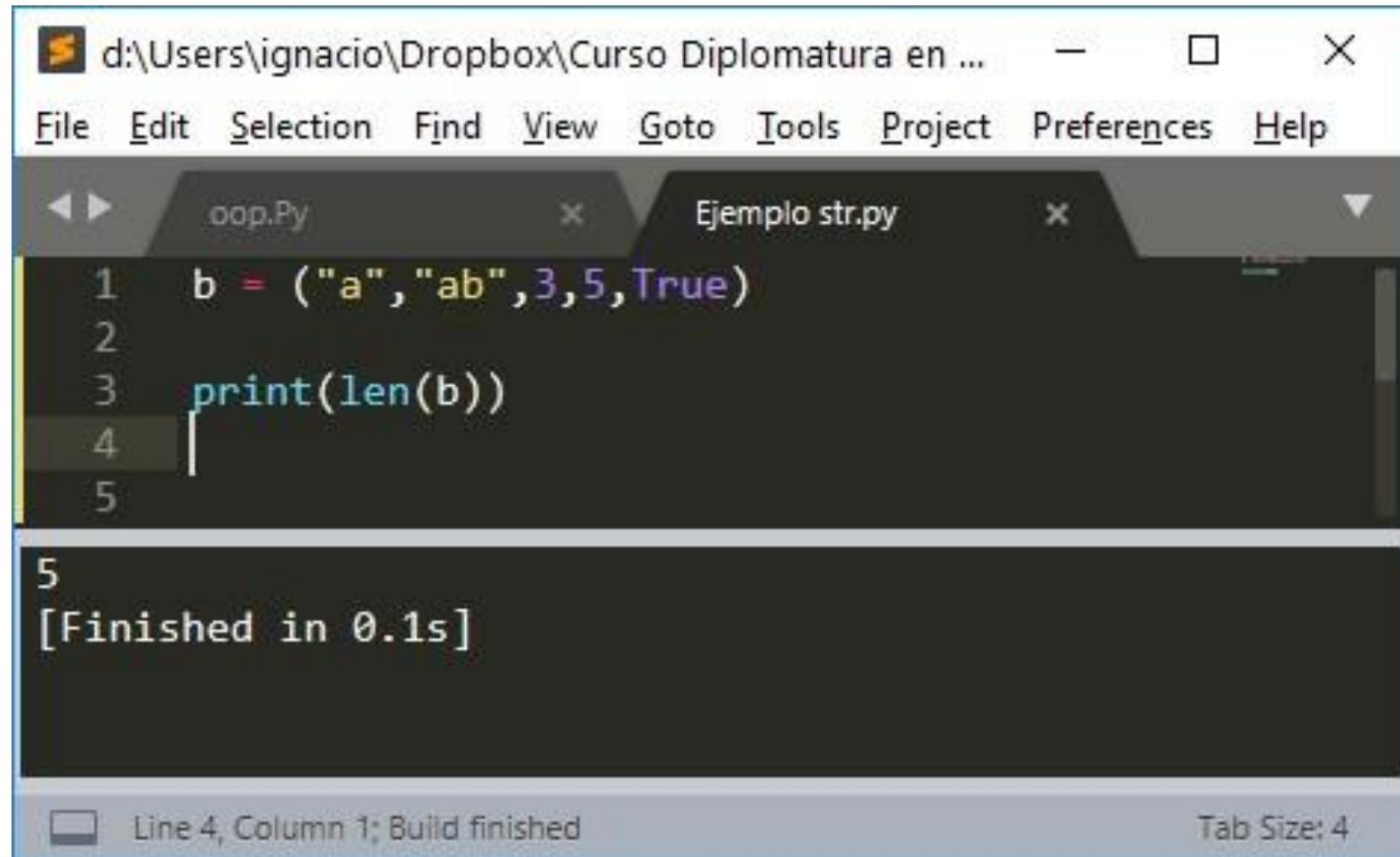
```
1 b = ("a", "ab", 3, 5, True)
2
3 print(3 in b)
4 print(4 in b)
5
```

The output window below shows the results of running the code:

```
True
False
[Finished in 0.1s]
```

At the bottom, status messages indicate "Line 5, Column 1: Build finished" and "Tab Size: 4".

Uso de len:



The screenshot shows a code editor window with two tabs: "oop.Py" and "Ejemplo str.py". The "Ejemplo str.py" tab is active, displaying the following Python code:

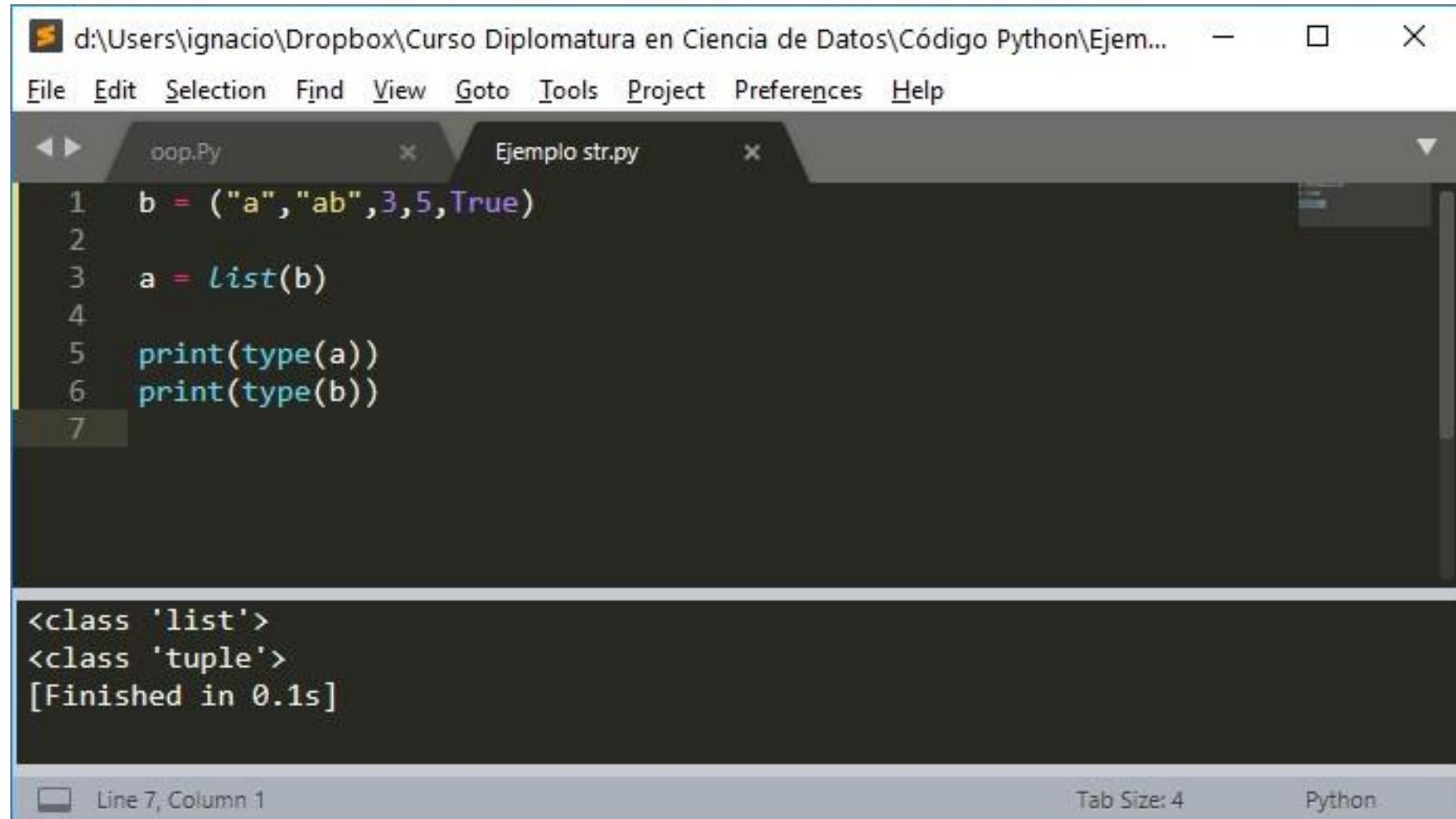
```
1 b = ("a","ab",3,5,True)
2
3 print(len(b))
4
5
```

The output window below shows the result of running the code:

```
5
[Finished in 0.1s]
```

At the bottom left, a status bar indicates "Line 4, Column 1; Build finished". At the bottom right, it says "Tab Size: 4".

Uso de list:



The screenshot shows a Python code editor with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab contains the following code:

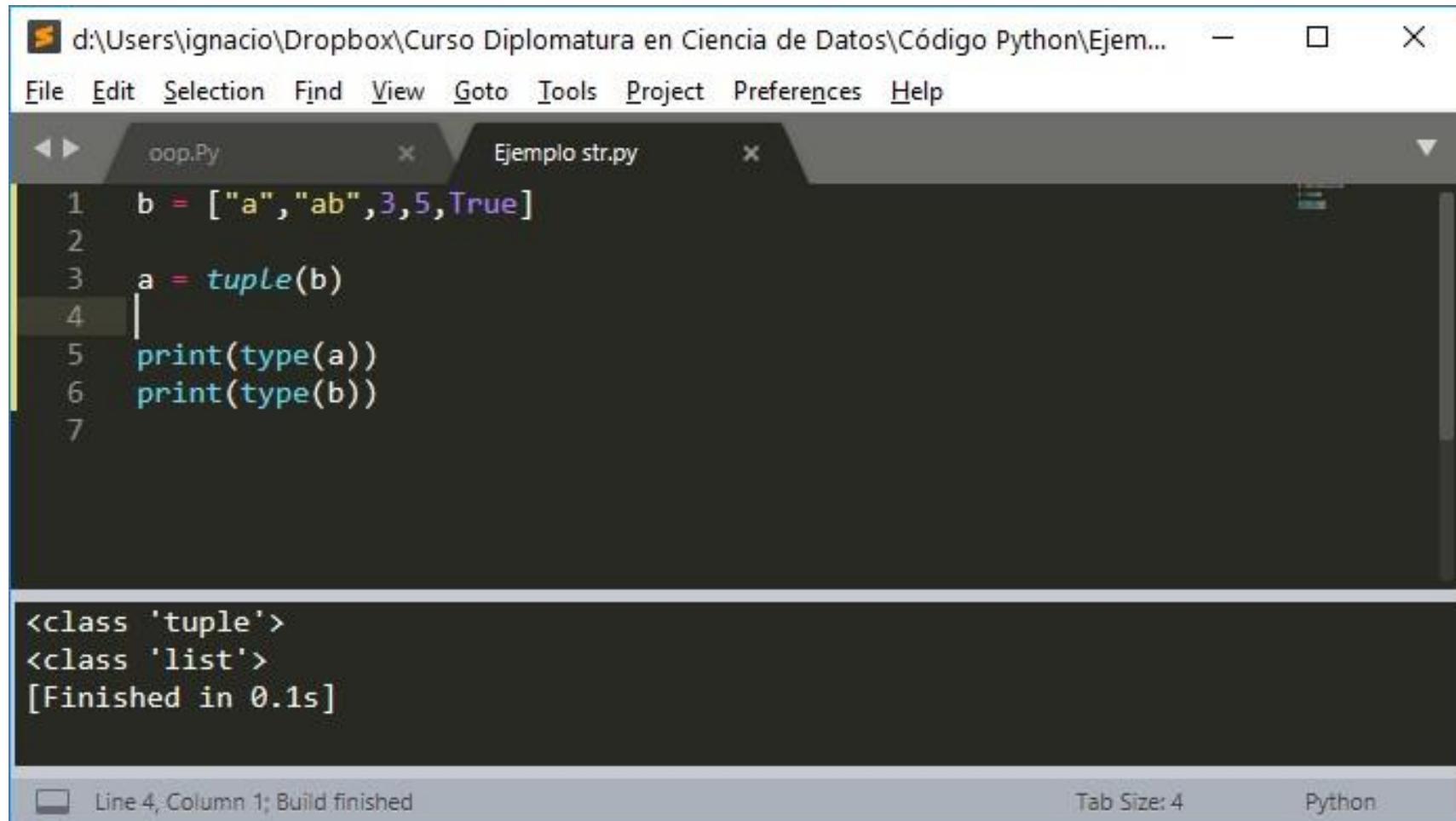
```
1 b = ("a", "ab", 3, 5, True)
2
3 a = list(b)
4
5 print(type(a))
6 print(type(b))
7
```

The 'Ejemplo str.py' tab is currently inactive. Below the code editor is a terminal window displaying the output of the code execution:

```
<class 'list'>
<class 'tuple'>
[Finished in 0.1s]
```

The status bar at the bottom shows 'Line 7, Column 1' and 'Tab Size: 4'.

Uso de tuple:



The screenshot shows a Python code editor with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'Ejemplo str.py' tab is active, displaying the following code:

```
1 b = ["a","ab",3,5,True]
2
3 a = tuple(b)
4
5 print(type(a))
6 print(type(b))
7
```

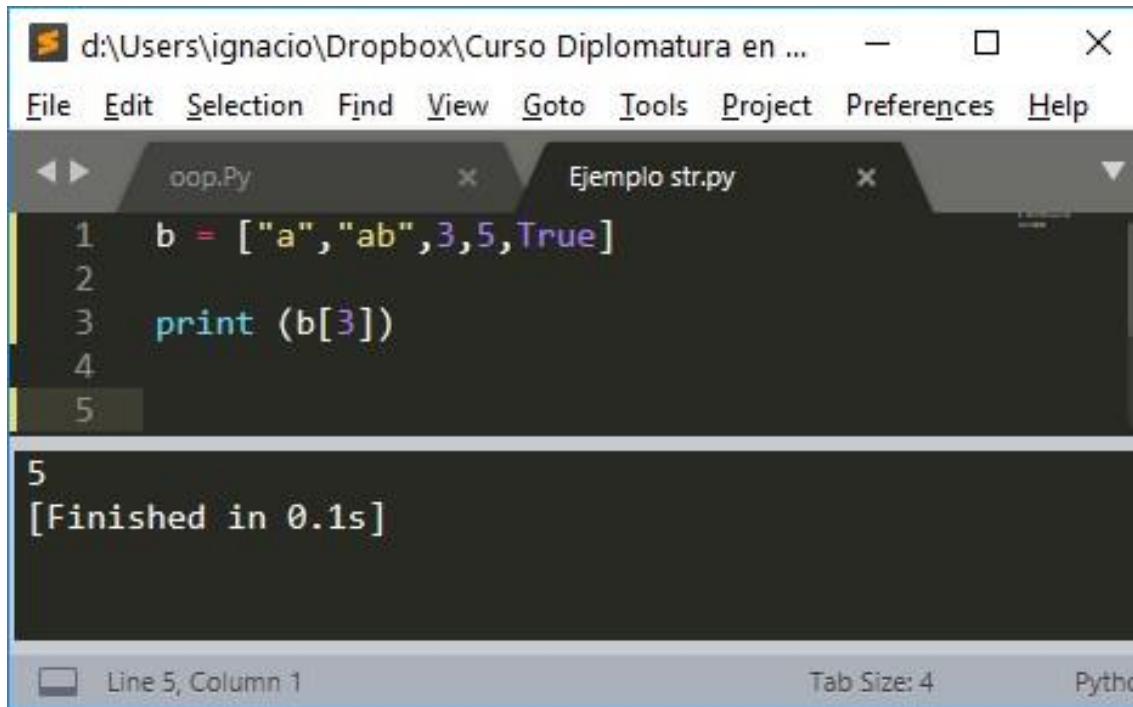
The output window below the editor shows the results of running the code:

```
<class 'tuple'>
<class 'list'>
[Finished in 0.1s]
```

At the bottom of the editor, status bars indicate 'Line 4, Column 1; Build finished', 'Tab Size: 4', and 'Python'.

Consultar un elemento de una lista:

lista[posición del elemento]



The screenshot shows a code editor window with two tabs: "oop.Py" and "Ejemplo str.py". The "Ejemplo str.py" tab is active, displaying the following Python code:

```
1 b = ["a", "ab", 3, 5, True]
2
3 print (b[3])
4
5
```

Below the code, the output window shows:

```
5
[Finished in 0.1s]
```

The status bar at the bottom indicates "Line 5, Column 1", "Tab Size: 4", and "Python".

Diccionarios:

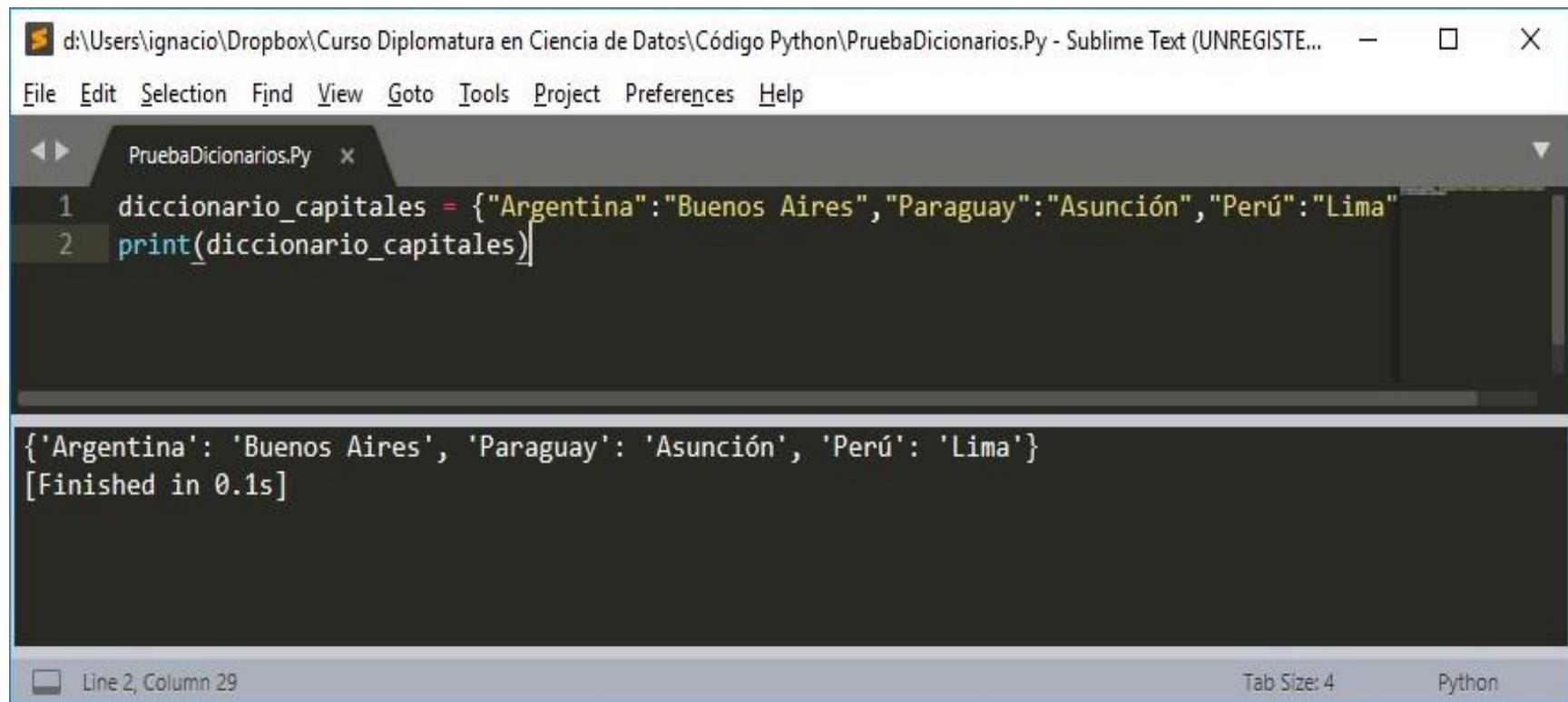
Son colecciones de objetos clave – valor

Permiten buscar el valor a partir de la clave

Las claves deben ser únicas

Usamos {} en vez de [] y ()

Creación de un diccionario:



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\PruebaDicionarios.Py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

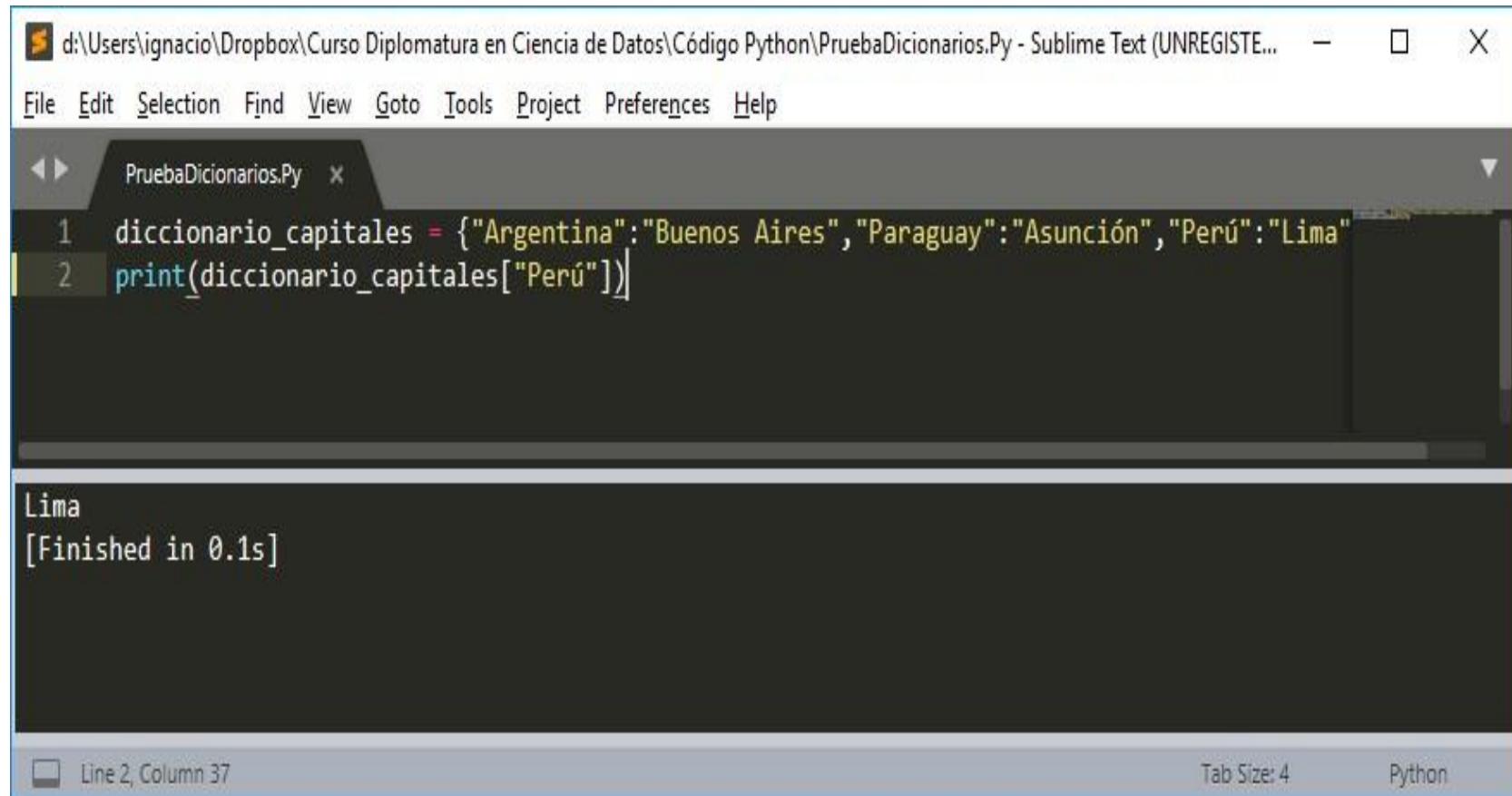
PruebaDicionarios.Py x

```
1 diccionario_capitales = {"Argentina": "Buenos Aires", "Paraguay": "Asunción", "Perú": "Lima"}  
2 print(diccionario_capitales)
```

```
{'Argentina': 'Buenos Aires', 'Paraguay': 'Asunción', 'Perú': 'Lima'}  
[Finished in 0.1s]
```

Line 2, Column 29 Tab Size: 4 Python

Para consultar un valor:



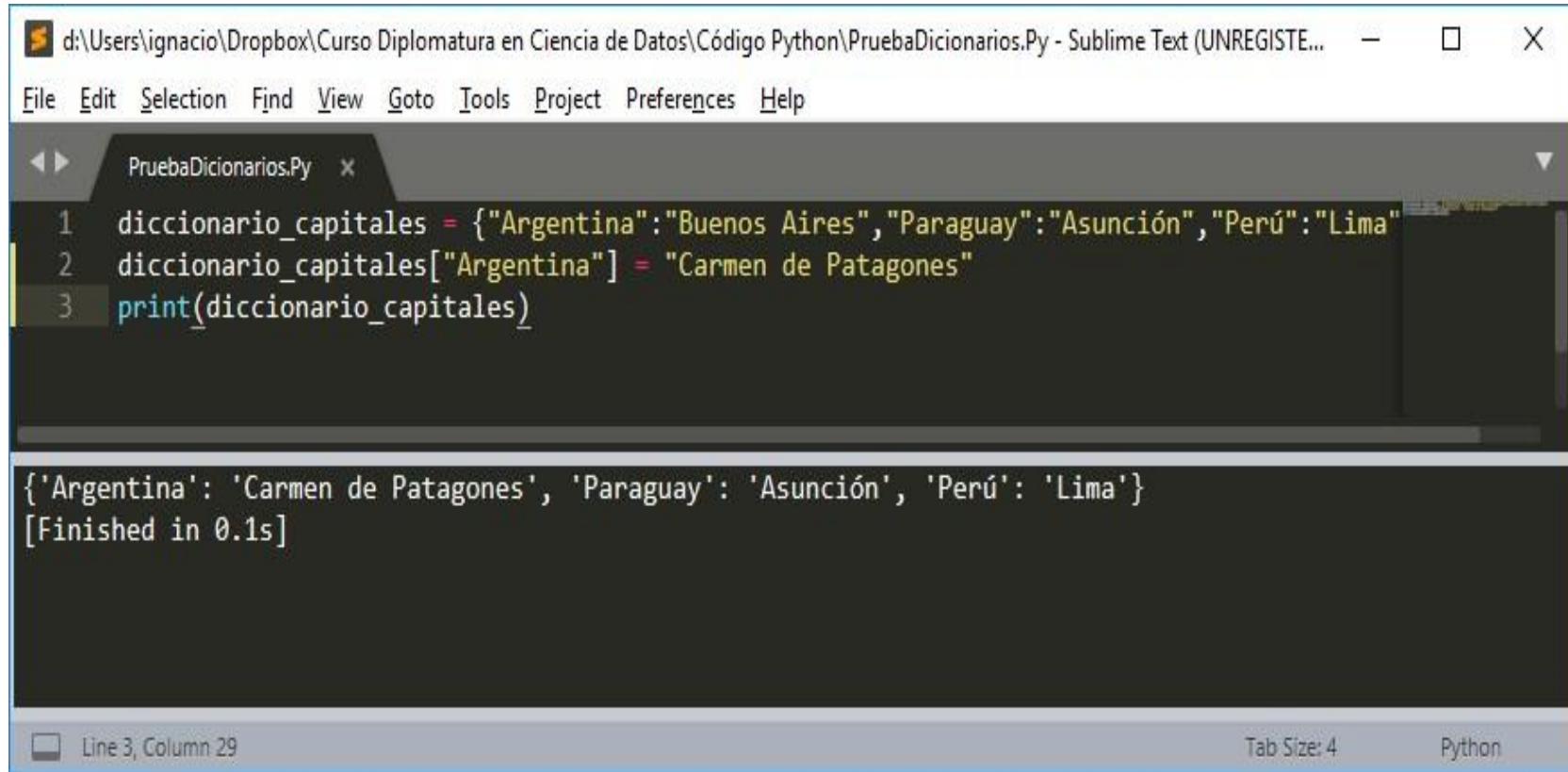
The screenshot shows a Sublime Text window with the following details:

- File Path: d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\PruebaDicionarios.Py - Sublime Text (UNREGISTERED)
- Menu Bar: File Edit Selection Find View Goto Tools Project Preferences Help
- Text Editor Content:

```
1 diccionario_capitales = {"Argentina": "Buenos Aires", "Paraguay": "Asunción", "Perú": "Lima"
2 print(diccionario_capitales["Perú"])]
```
- Output Panel Content:

```
Lima
[Finished in 0.1s]
```
- Status Bar: Line 2, Column 37 | Tab Size: 4 | Python

Para cambiar un valor:



The screenshot shows a Sublime Text window with the following details:

- Title Bar:** d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\PruebaDicionarios.Py - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Text Editor:** The file "PruebaDicionarios.Py" is open, containing the following Python code:

```
1 diccionario_capitales = {"Argentina": "Buenos Aires", "Paraguay": "Asunción", "Perú": "Lima"
2 diccionario_capitales["Argentina"] = "Carmen de Patagones"
3 print(diccionario_capitales)
```
- Output Panel:** The output of the code execution is displayed:

```
{'Argentina': 'Carmen de Patagones', 'Paraguay': 'Asunción', 'Perú': 'Lima'}
[Finished in 0.1s]
```
- Status Bar:** Line 3, Column 29 | Tab Size: 4 | Python

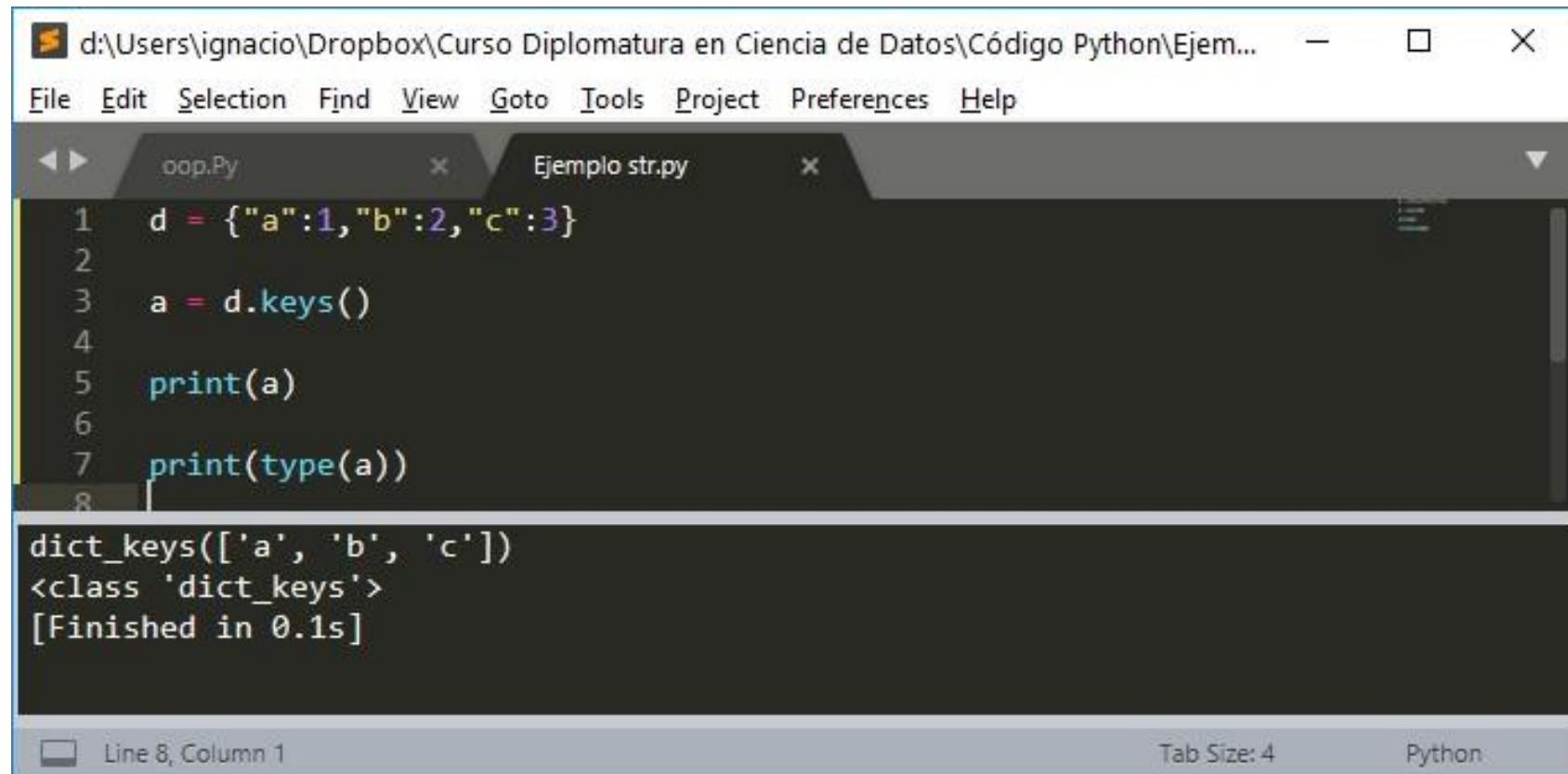
Métodos de los diccionarios:

keys

value

s len

Uso de keys:



The screenshot shows a Python code editor with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'Ejemplo str.py' tab is active, displaying the following code:

```
1 d = {"a":1,"b":2,"c":3}
2
3 a = d.keys()
4
5 print(a)
6
7 print(type(a))
8 [
```

Below the code, the output of the execution is shown:

```
dict_keys(['a', 'b', 'c'])
<class 'dict_keys'>
[Finished in 0.1s]
```

The status bar at the bottom indicates 'Line 8, Column 1', 'Tab Size: 4', and 'Python'.

Uso de values:

Instituto Data Science Argentina



The screenshot shows a code editor window with two tabs: 'oop.py' and 'Ejemplo str.py'. The 'oop.py' tab is active, displaying the following Python code:

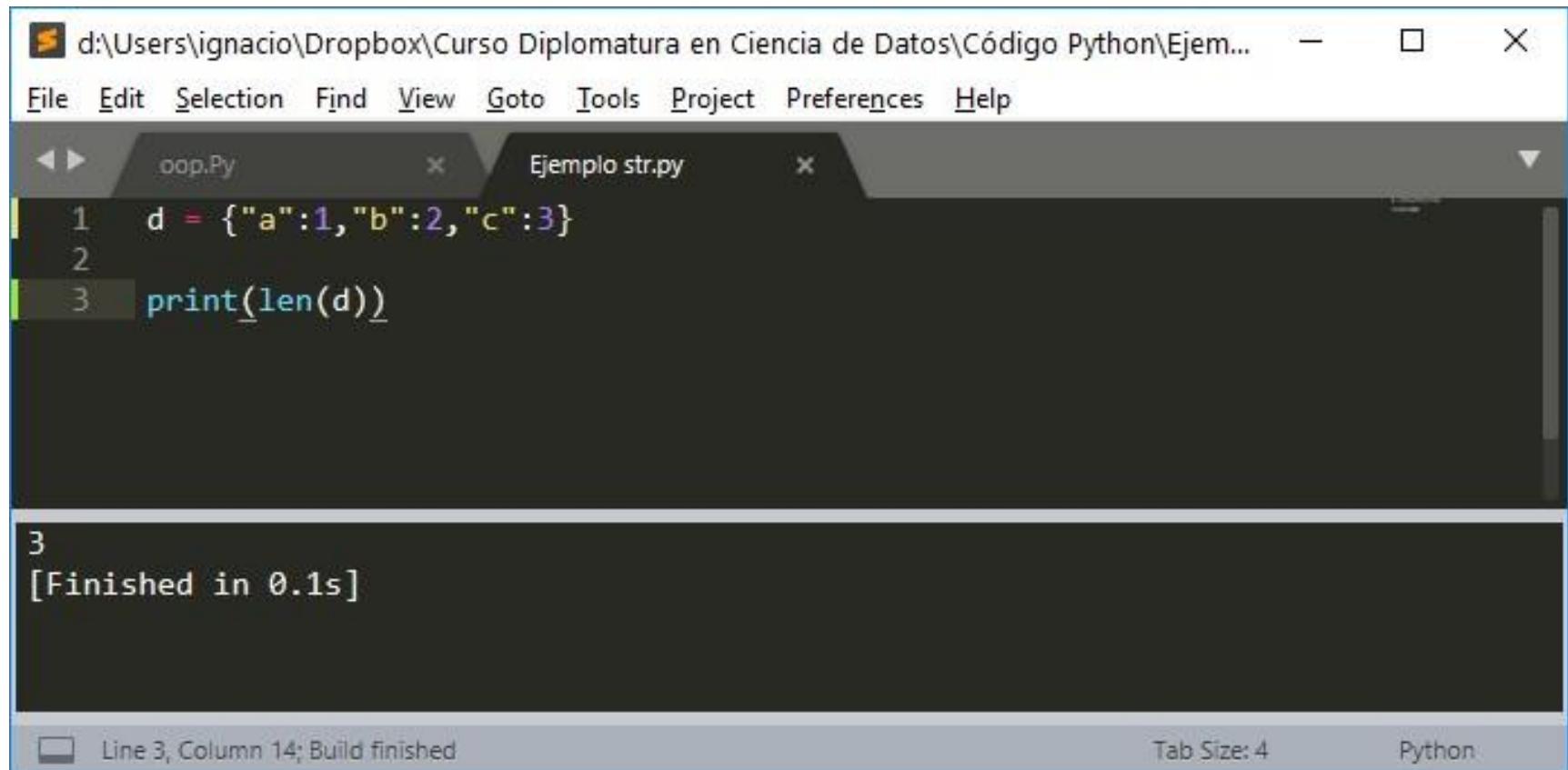
```
1 d = {"a":1,"b":2,"c":3}
2
3 a = d.values()
4
5 print(a)
6
7 print(type(a))
8
```

Below the code, the output of running the script is shown:

```
dict_values([1, 2, 3])
<class 'dict_values'>
[Finished in 0.1s]
```

The status bar at the bottom indicates 'Line 4, Column 1', 'Tab Size: 4', and 'Python'.

Uso de len:



The screenshot shows a code editor window with two tabs: "oop.py" and "Ejemplo str.py". The "Ejemplo str.py" tab is active, displaying the following Python code:

```
1 d = {"a":1,"b":2,"c":3}
2
3 print(len(d))
```

The output window below shows the result of running the code:

```
3
[Finished in 0.1s]
```

At the bottom of the editor, status bars indicate "Line 3; Column 14; Build finished", "Tab Size: 4", and "Python".

Funciones y manejo de archivos:

Instituto Data Science Argentina

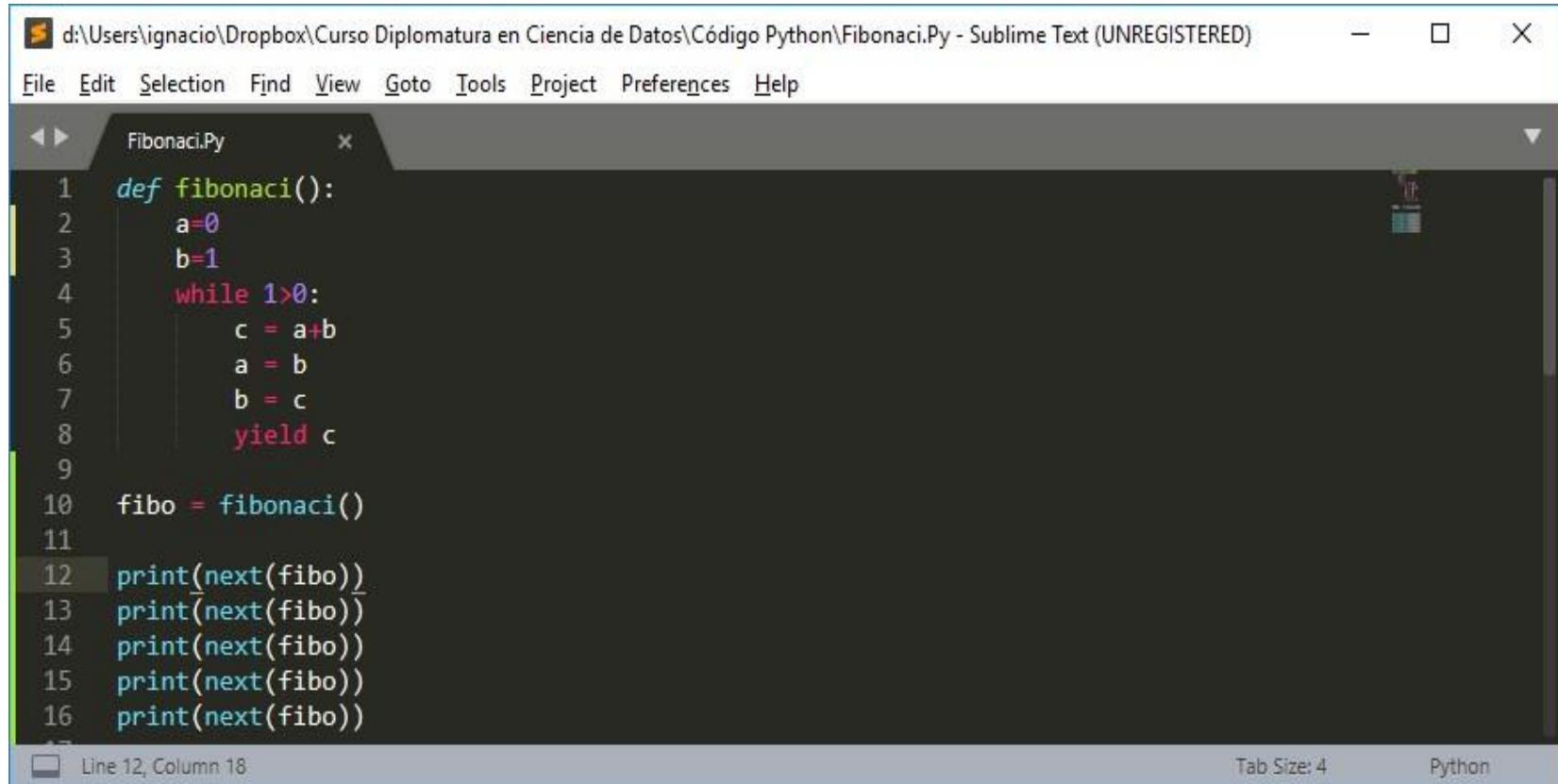
- *Generadores*
- *Excepciones*
- *Ámbito de las variables*
- *Módulo so*
- *Módulo fs*
- *Módulo re*
- *Módulo io*

Generadores:

Instituto Data Science Argentina

- *Producen una serie de valores*
- *Potencialmente infinita*
- *De a uno*

Generadores:

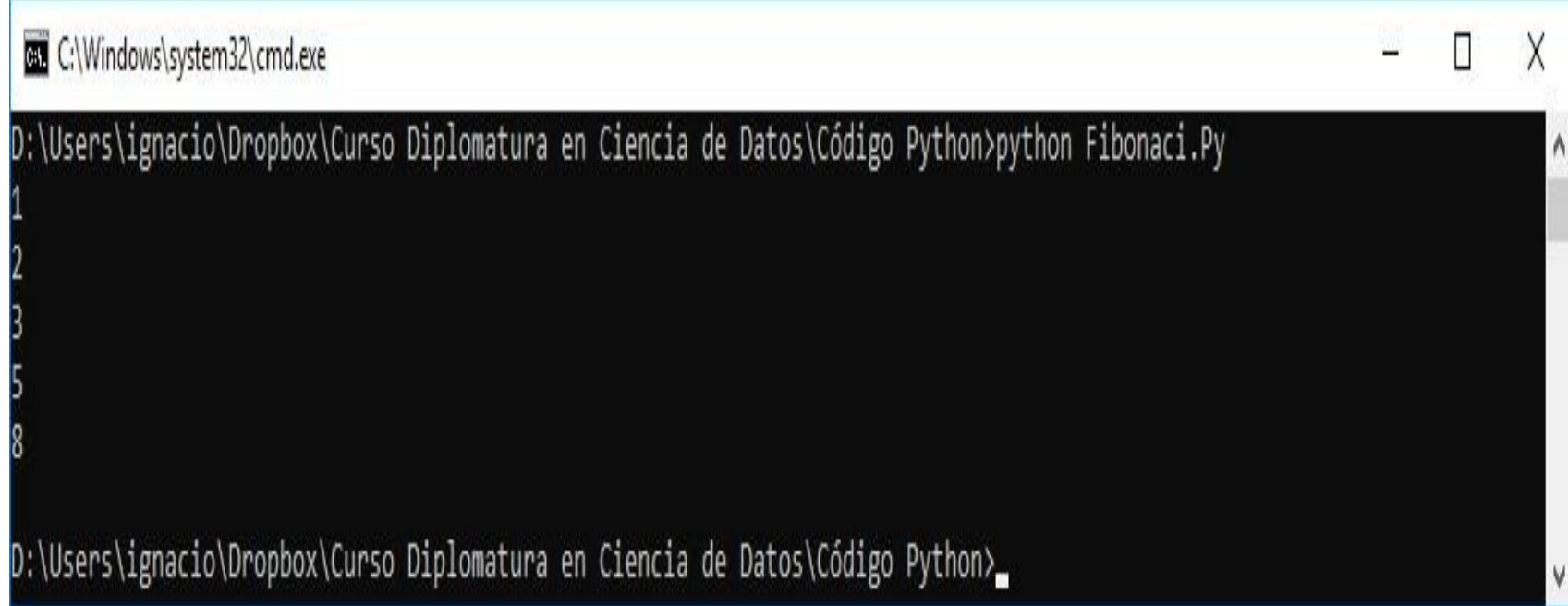


The screenshot shows a Sublime Text window with a dark theme. The title bar reads "d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Fibonaci.Py - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. A tab titled "Fibonaci.Py" is open, displaying the following Python code:

```
1 def fibonacci():
2     a=0
3     b=1
4     while 1>0:
5         c = a+b
6         a = b
7         b = c
8         yield c
9
10 fibo = fibonacci()
11
12 print(next(fibo))
13 print(next(fibo))
14 print(next(fibo))
15 print(next(fibo))
16 print(next(fibo))
```

The status bar at the bottom left shows "Line 12, Column 18". The status bar at the bottom right shows "Tab Size: 4" and "Python".

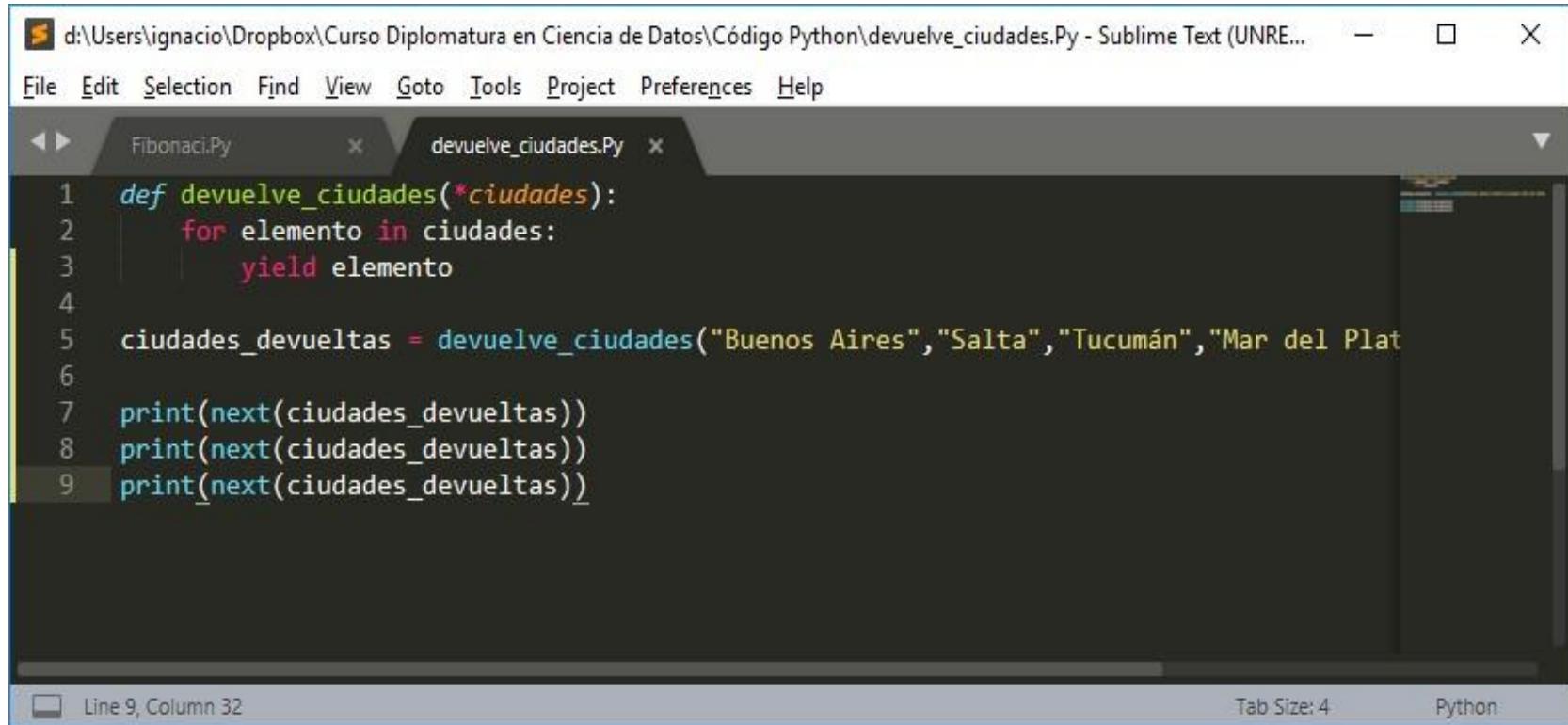
Generadores:



A screenshot of a Windows Command Prompt window titled "cmd C:\Windows\system32\cmd.exe". The window shows the command "D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>python Fibonacci.py" and its output, which are the first eight numbers of the Fibonacci sequence: 1, 1, 2, 3, 5, 8.

```
D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>python Fibonacci.py
1
1
2
3
5
8
D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>
```

Cantidad variable de argumentos

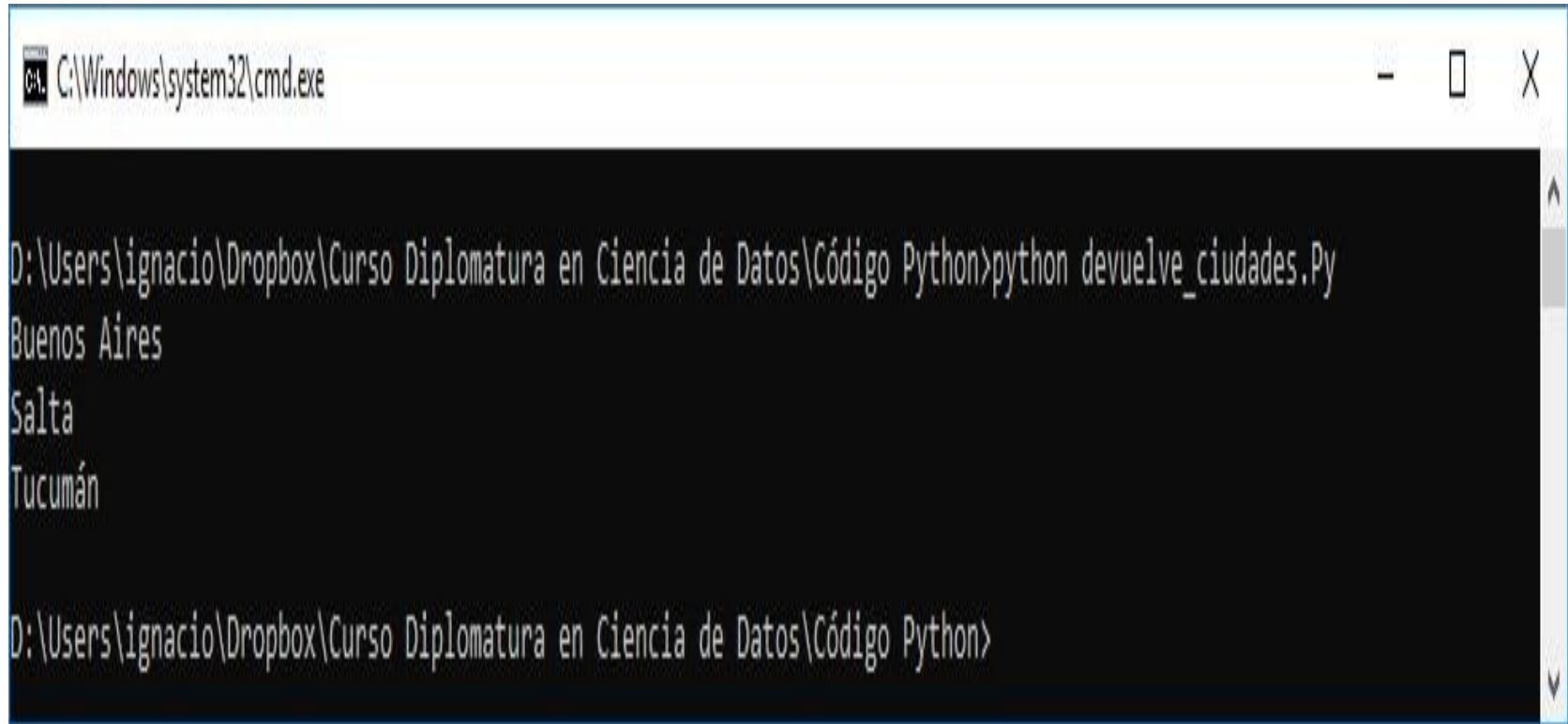


The screenshot shows a Sublime Text window with two tabs: 'Fibonaci.Py' and 'devuelve_ciudades.Py'. The 'devuelve_ciudades.Py' tab is active and displays the following Python code:

```
1 def devuelve_ciudades(*ciudades):
2     for elemento in ciudades:
3         yield elemento
4
5 ciudades_devueltas = devuelve_ciudades("Buenos Aires", "Salta", "Tucumán", "Mar del Plat"
6
7 print(next(ciudades_devueltas))
8 print(next(ciudades_devueltas))
9 print(next(ciudades_devueltas))
```

The code defines a generator function 'devuelve_ciudades' that takes a variable number of city arguments and yields each one. It then creates a variable 'ciudades_devueltas' by calling this function with four city names. Finally, it prints the next three values from this generator using 'next()'.

Cantidad variable de argumentos



A screenshot of a Windows Command Prompt window titled 'cmd C:\Windows\system32\cmd.exe'. The window shows the command 'python devuelve_ciudades.py' being run, followed by three city names: 'Buenos Aires', 'Salta', and 'Tucumán'. The prompt then returns to the directory 'D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python'.

```
D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>python devuelve_ciudades.py
Buenos Aires
Salta
Tucumán

D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>
```

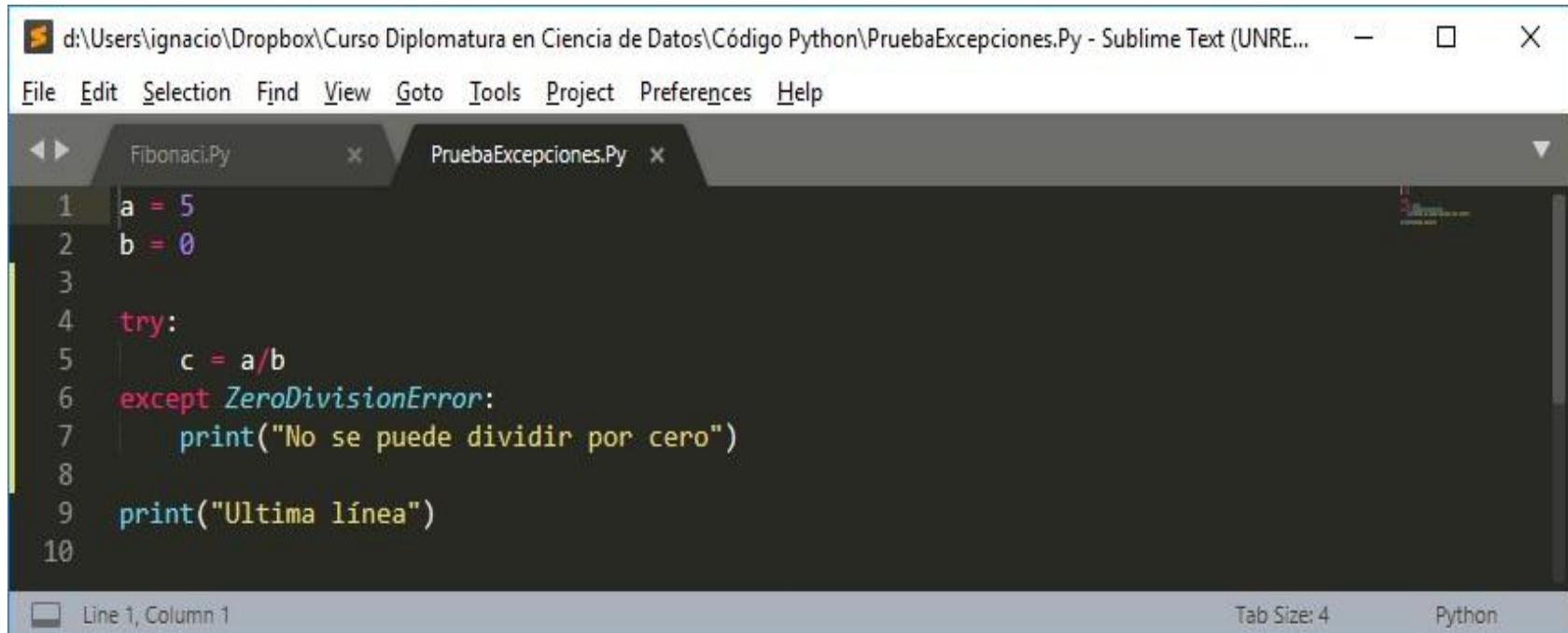
Excepciones:

- Ocurren en programas con sintaxis correcta
- Aparecen cuando hay una situación inesperada durante la ejecución
- Normalmente detendrían la ejecución del programa
- Pueden dejar las tareas en un estado inconsistente

Excepciones:

Instituto Data Science Argentina

- Sintaxis:



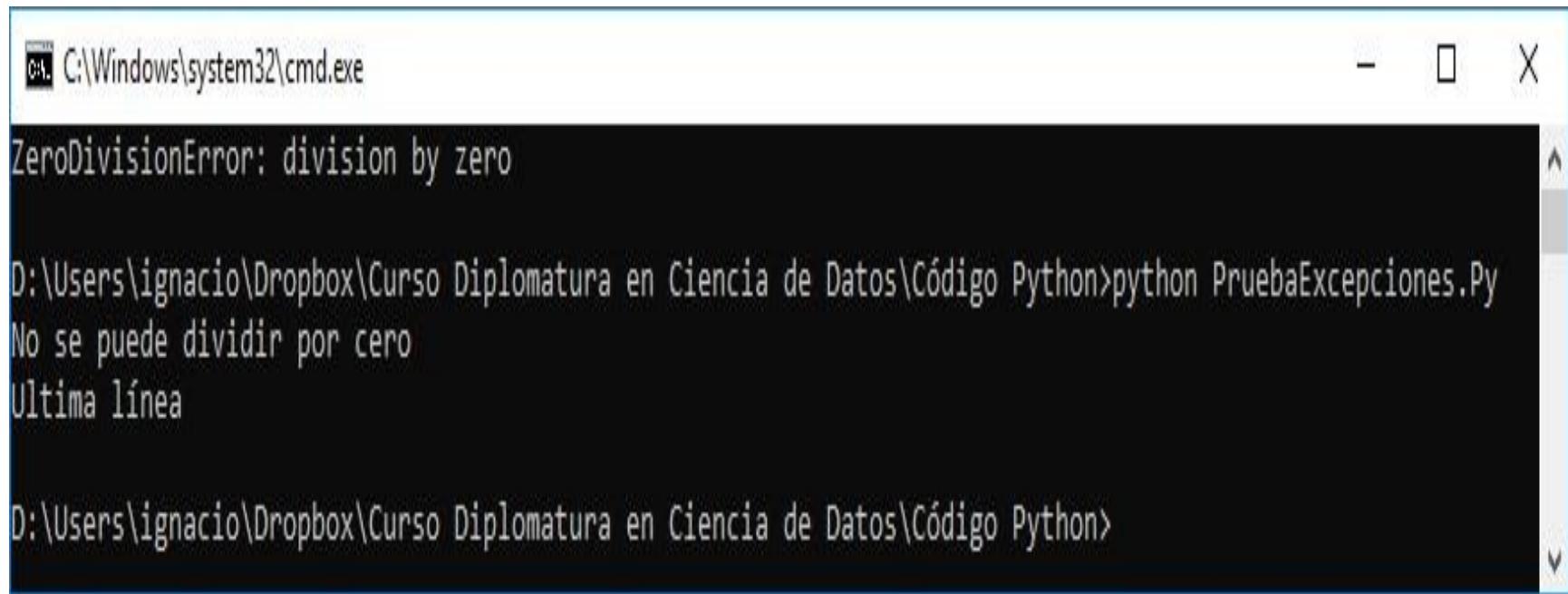
The screenshot shows a Sublime Text window with two tabs: 'Fibonaci.Py' and 'PruebaExcepciones.Py'. The 'PruebaExcepciones.Py' tab is active and displays the following Python code:

```
1 a = 5
2 b = 0
3
4 try:
5     c = a/b
6 except ZeroDivisionError:
7     print("No se puede dividir por cero")
8
9 print("Ultima linea")
10
```

The code uses color-coded syntax highlighting: numbers are blue, strings are green, and keywords like 'try' and 'except' are red. The Sublime Text interface includes a menu bar with File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. At the bottom, status bars show 'Line 1, Column 1', 'Tab Size: 4', and 'Python'.

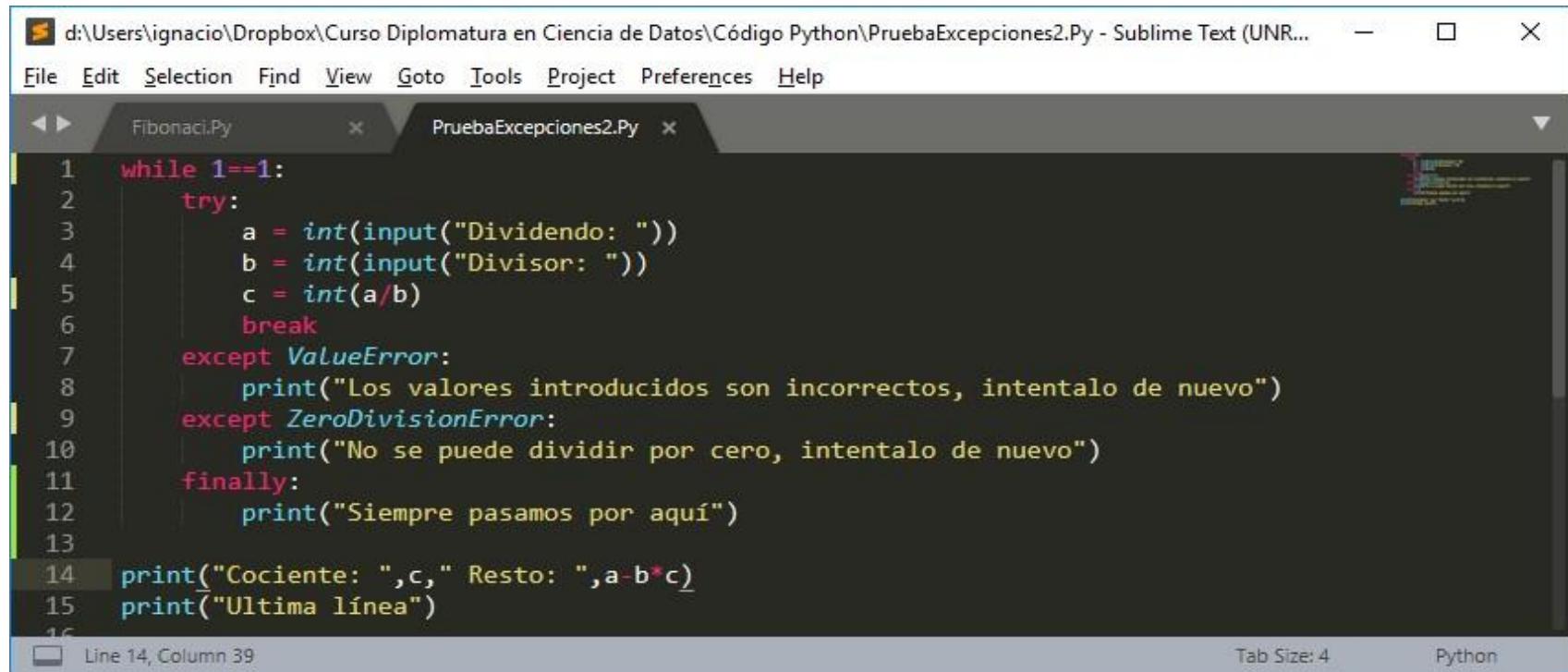
Excepciones:

- Ejecución:



```
C:\Windows\system32\cmd.exe
ZeroDivisionError: division by zero
D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>python PruebaExcepciones.Py
No se puede dividir por cero
Ultima linea
D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>
```

Excepciones: finally



The screenshot shows a Sublime Text window with two tabs: 'Fibonaci.Py' and 'PruebaExcepciones2.Py'. The 'PruebaExcepciones2.Py' tab is active and displays the following Python code:

```
1 while 1==1:
2     try:
3         a = int(input("Dividendo: "))
4         b = int(input("Divisor: "))
5         c = int(a/b)
6         break
7     except ValueError:
8         print("Los valores introducidos son incorrectos, intentalo de nuevo")
9     except ZeroDivisionError:
10        print("No se puede dividir por cero, intentalo de nuevo")
11    finally:
12        print("Siempre pasamos por aquí")
13
14 print("Cociente: ",c," Resto: ",a-b*c)
15 print("Ultima linea")
```

The status bar at the bottom indicates 'Line 14, Column 39', 'Tab Size: 4', and 'Python'.

Excepciones: *finally*

```
C:\Windows\system32\cmd.exe

D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>python PruebaExcepciones2.py
Dividendo: 5
Divisor: 1
Siempre pasamos por aquí
Cociente: 5 Resto: 0
Última línea

D:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python>
```

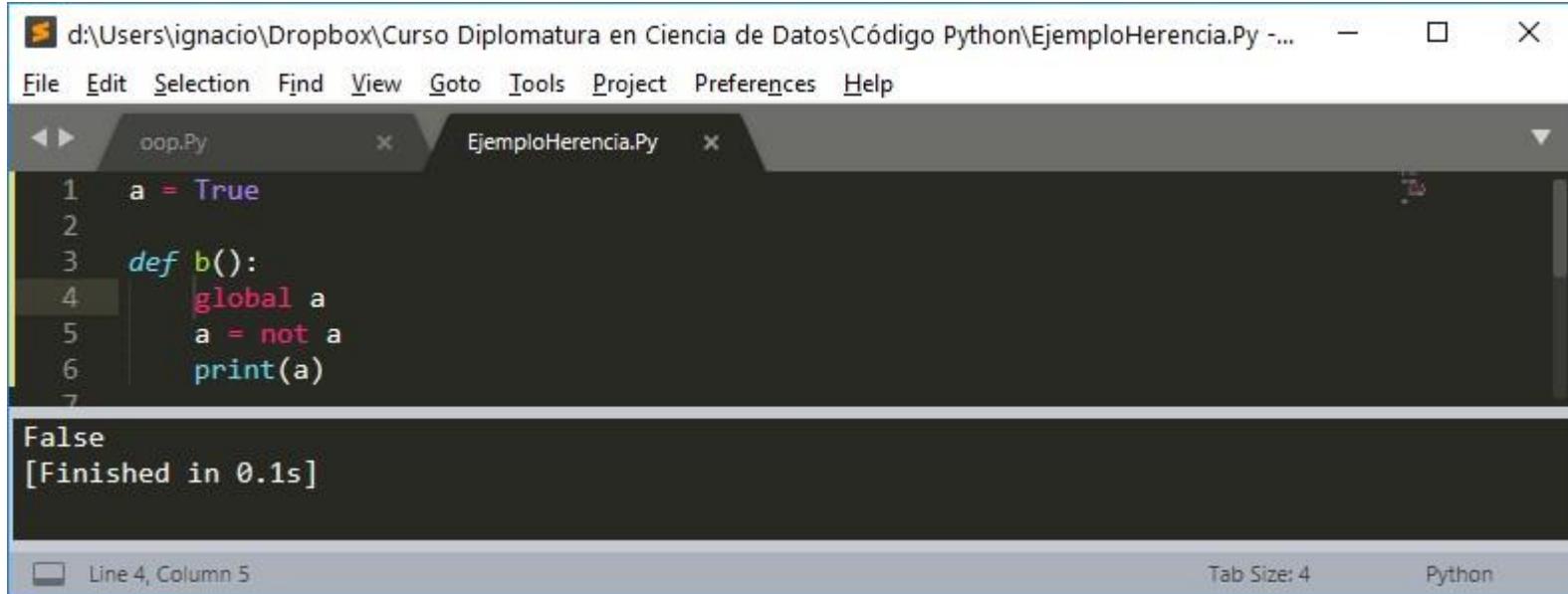
Paquete math

- sqrt()
- floor()
- ceil()
- trunc()
- fsum() • pow() *Ámbito de las variables:*

- Si defino una variable dentro de una función no es visible desde afuera.
- Si defino una variable dentro del cuerpo principal puedo verla dentro de una función
- Si defino una variable en el cuerpo principal y luego también la referencia dentro de una función

Ámbito de las variables:

- Si quiero usar una variable dentro de una función que viene de afuera puedo declararla global:



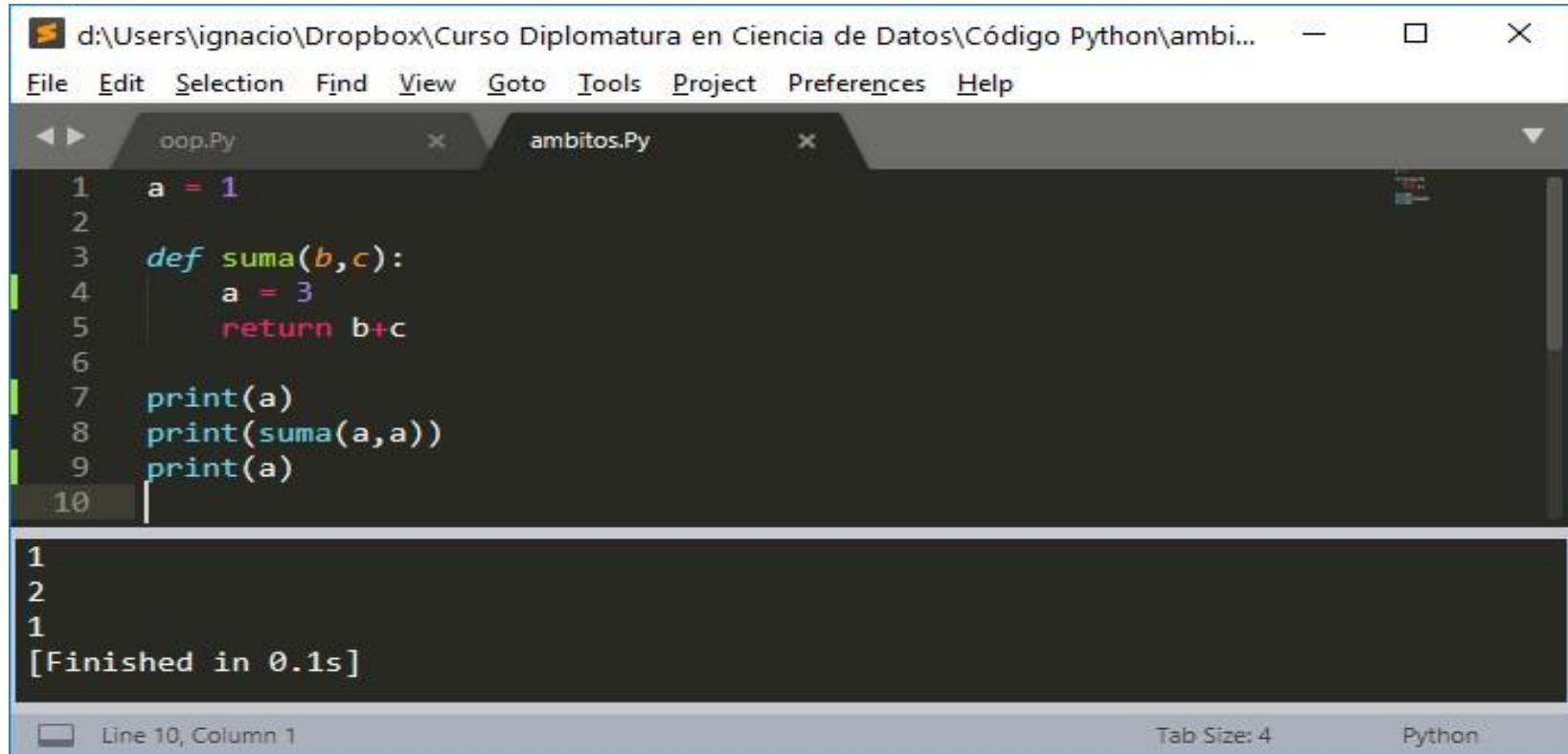
The screenshot shows a Python code editor with two tabs: 'oop.py' and 'EjemploHerencia.py'. The 'EjemploHerencia.py' tab is active, displaying the following code:

```
1 a = True
2
3 def b():
4     global a
5     a = not a
6     print(a)
7
8
9 False
[Finished in 0.1s]
```

The code defines a variable 'a' as True at line 1. It then defines a function 'b' at line 3. Inside 'b', it declares 'a' as global (line 4) and changes its value to not a (line 5). It then prints the value of 'a' (line 6). The output window below shows the result as 'False' and indicates the process finished in 0.1s.

Ámbito de las variables:

- Si no la declaro global es otra variable:



The screenshot shows a code editor window with two tabs: 'oop.py' and 'ambitos.py'. The 'ambitos.py' tab is active, displaying the following Python code:

```
1 a = 1
2
3 def suma(b,c):
4     a = 3
5     return b+c
6
7 print(a)
8 print(suma(1,2))
9 print(a)
10
```

The output window below shows the execution results:

```
1
2
1
[Finished in 0.1s]
```

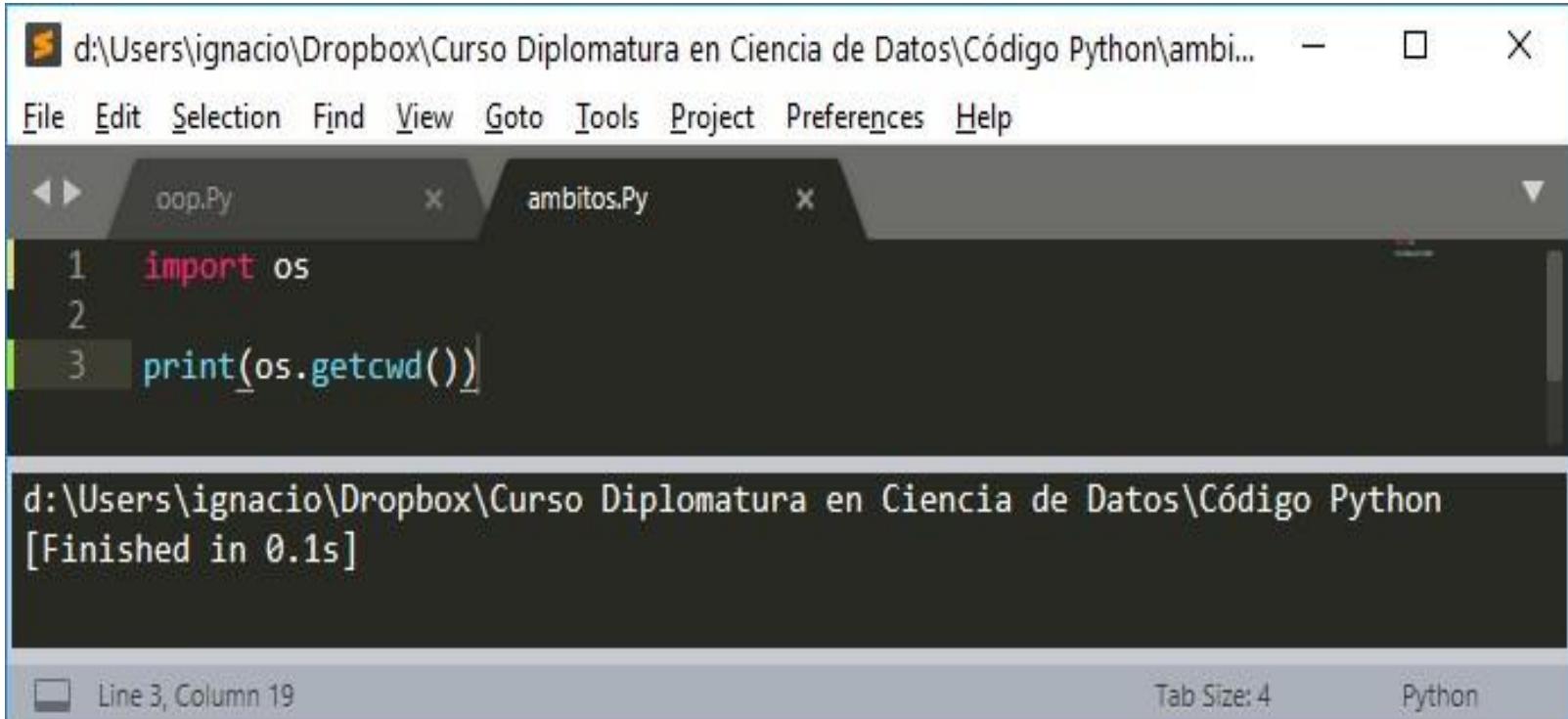
At the bottom, status bars indicate 'Line 10, Column 1', 'Tab Size: 4', and 'Python'.

Módulo os:

- Se usa para manejar archivos y directorios
 - getcwd()
 - listdir()
 - walk()
 - chdir()
 - makedirs()
 - remove()

getcwd:

- Devuelve el directorio por defecto



The screenshot shows a Python code editor interface. At the top, there are two tabs: "oop.py" and "ambitos.py". The "oop.py" tab is active, displaying the following code:

```
1 import os
2
3 print(os.getcwd())
```

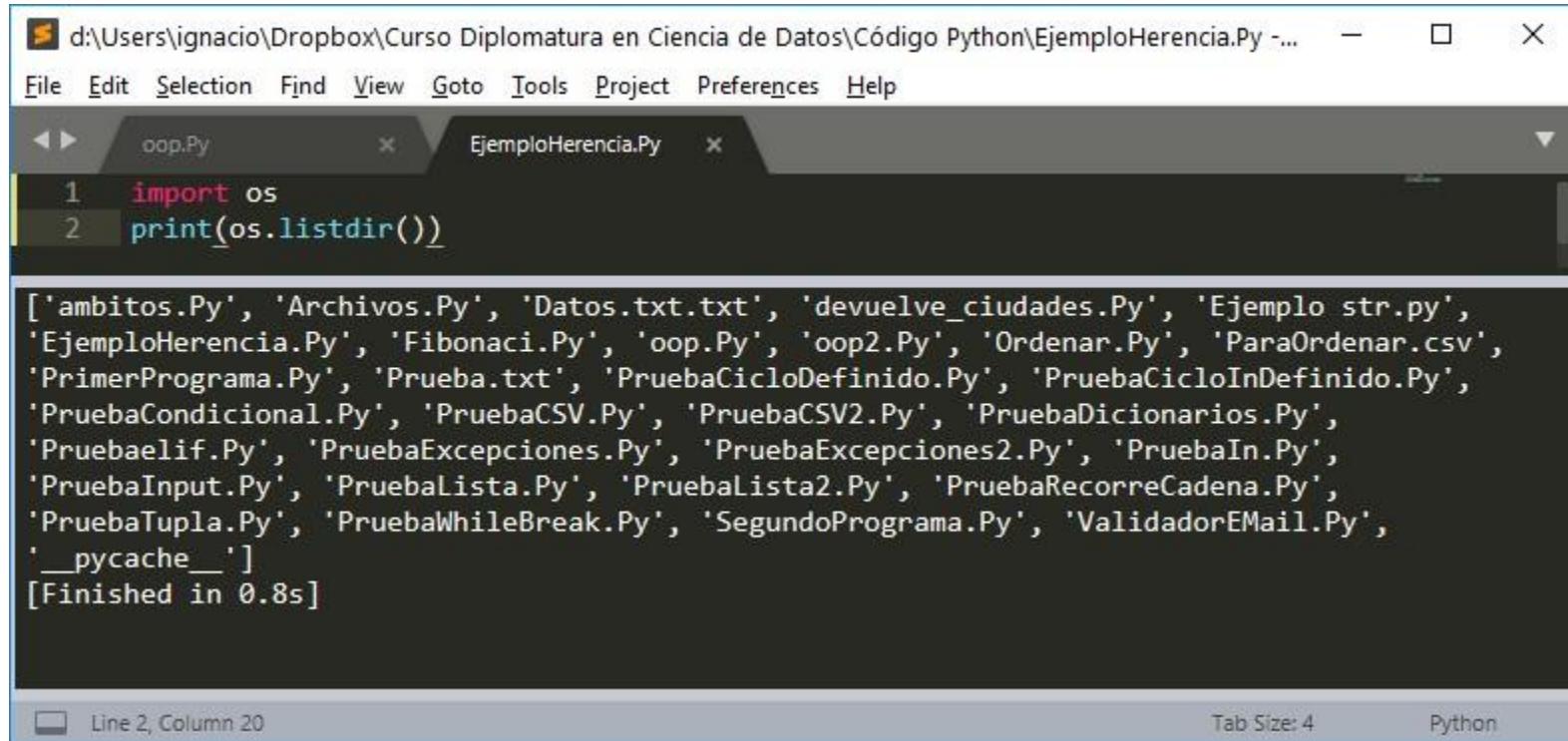
Below the code editor is a terminal window showing the output of the script:

```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python
[Finished in 0.1s]
```

At the bottom of the terminal window, it says "Python".

listdir:

- Devuelve los archivos y directorios



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\EjemploHerencia.Py ... - □ X

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py EjemploHerencia.py

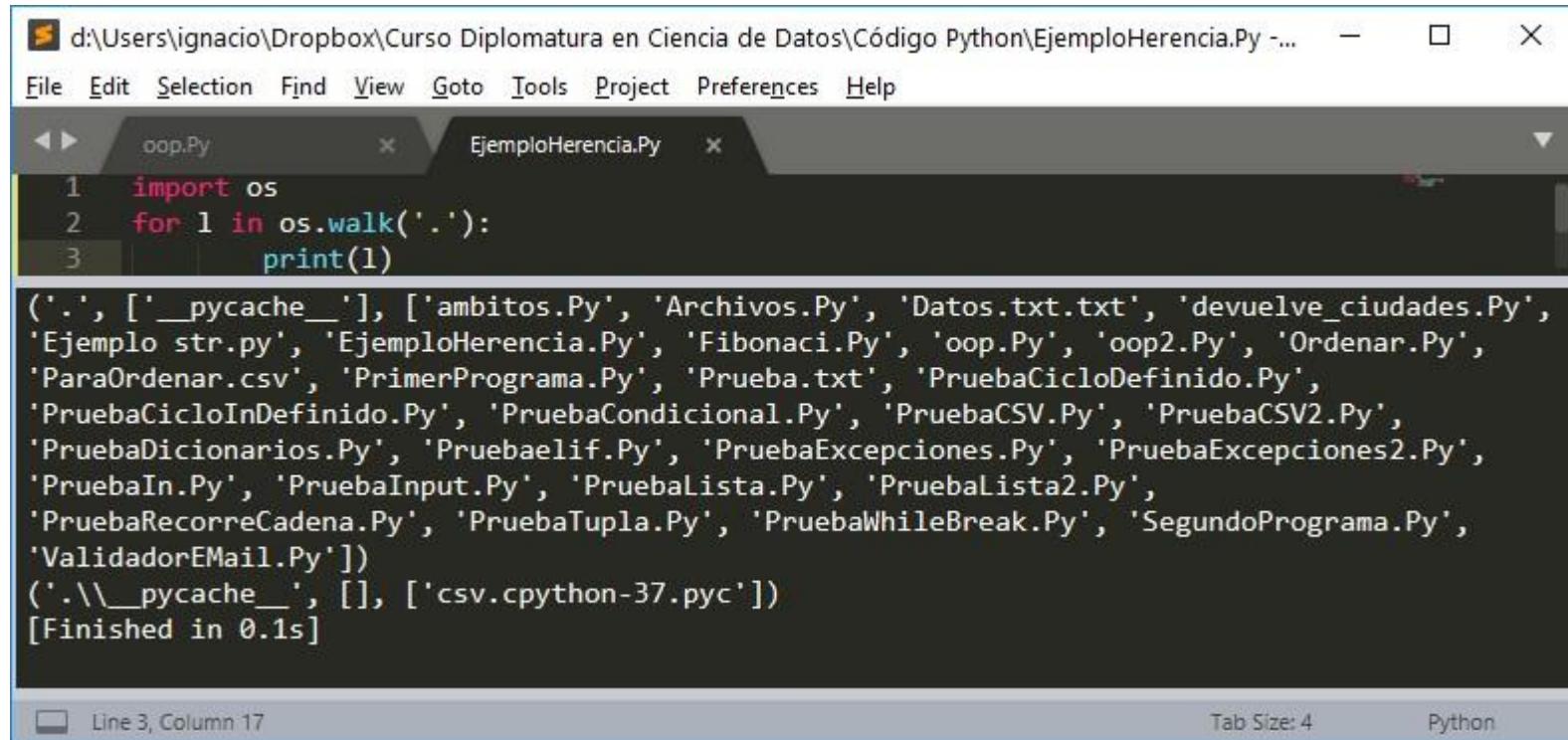
```
1 import os
2 print(os.listdir())
```

```
['ambitos.Py', 'Archivos.Py', 'Datos.txt.txt', 'devuelve_ciudades.Py', 'Ejemplo str.py',
'EjemploHerencia.Py', 'Fibonaci.Py', 'oop.Py', 'oop2.Py', 'Ordenar.Py', 'ParaOrdenar.csv',
'PrimerPrograma.Py', 'Prueba.txt', 'PruebaCicloDefinido.Py', 'PruebaCicloInDefinido.Py',
'PruebaCondicional.Py', 'PruebaCSV.Py', 'PruebaCSV2.Py', 'PruebaDicionarios.Py',
'Prueba elif.Py', 'PruebaExcepciones.Py', 'PruebaExcepciones2.Py', 'PruebaIn.Py',
'PruebaInput.Py', 'PruebaLista.Py', 'PruebaLista2.Py', 'PruebaRecorreCadena.Py',
'PruebaTupla.Py', 'PruebaWhileBreak.Py', 'SegundoPrograma.Py', 'ValidadorEMail.Py',
'__pycache__']
[Finished in 0.8s]
```

Line 2, Column 20 Tab Size: 4 Python

walk:

- Devuelve los archivos y directorios a partir de un directorio entrando recursivamente en los subdirectorios:



The screenshot shows a code editor window with two tabs: 'oop.py' and 'EjemploHerencia.py'. The code in 'oop.py' is as follows:

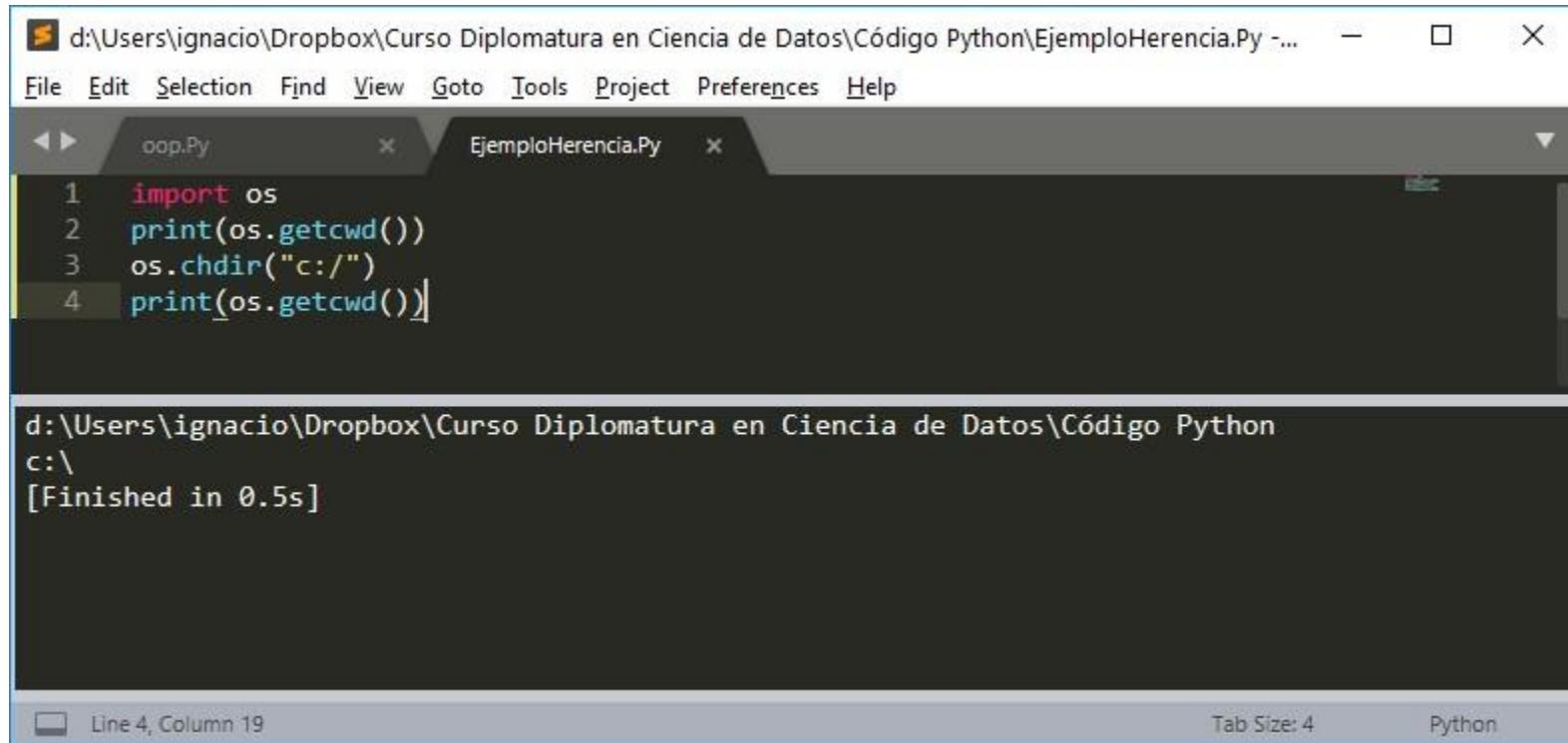
```
1 import os
2 for l in os.walk('.'):
3     print(l)
```

The output of the code is displayed in the terminal area of the editor. It lists all files and subdirectories in the current directory and its subdirectories. The output starts with the current directory ('.') and includes several hidden directories like '__pycache__' and files like 'ambitos.py', 'Archivos.py', etc. The output ends with '[Finished in 0.1s]'. The status bar at the bottom shows 'Line 3, Column 17', 'Tab Size: 4', and 'Python'.

```
('.', ['__pycache__'], ['ambitos.py', 'Archivos.py', 'Datos.txt.txt', 'devuelve_ciudades.py', 'Ejemplo str.py', 'EjemploHerencia.py', 'Fibonaci.py', 'oop.py', 'oop2.py', 'Ordenar.py', 'ParaOrdenar.csv', 'PrimerPrograma.py', 'Prueba.txt', 'PruebaCicloDefinido.py', 'PruebaCicloInDefinido.py', 'PruebaCondicional.py', 'PruebaCSV.py', 'PruebaCSV2.py', 'PruebaDicionarios.py', 'Pruebaelif.py', 'PruebaExcepciones.py', 'PruebaExcepciones2.py', 'PruebaIn.py', 'PruebaInput.py', 'PruebaLista.py', 'PruebaLista2.py', 'PruebaRecorreCadena.py', 'PruebaTupla.py', 'PruebaWhileBreak.py', 'SegundoPrograma.py', 'ValidadorEMail.py'])
('..\__pycache__', [], ['csv.cpython-37.pyc'])
[Finished in 0.1s]
```

chdir:

- Cambia el directorio actual:



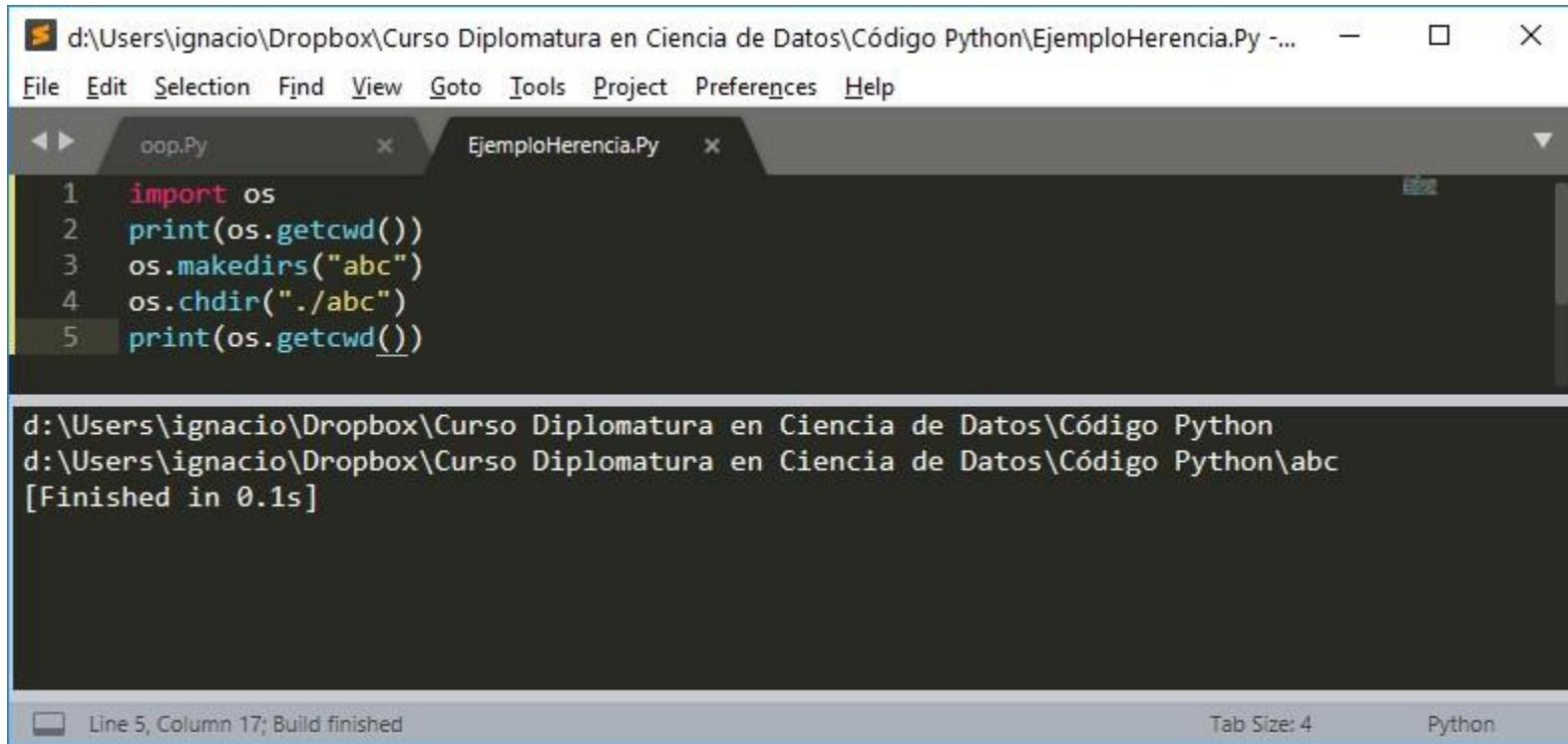
```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\EjemploHerencia.Py ... - □ X
File Edit Selection Find View Goto Tools Project Preferences Help
◀ ▶ oop.Py × EjemploHerencia.Py ×
1 import os
2 print(os.getcwd())
3 os.chdir("c:/")
4 print(os.getcwd())

d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python
c:\
[Finished in 0.5s]

Line 4, Column 19 Tab Size: 4 Python
```

makedirs:

- Crea un directorio:



The screenshot shows a Python code editor window with two tabs: 'oop.py' and 'EjemploHerencia.py'. The 'EjemploHerencia.py' tab is active, displaying the following code:

```
1 import os
2 print(os.getcwd())
3 os.makedirs("abc")
4 os.chdir("./abc")
5 print(os.getcwd())
```

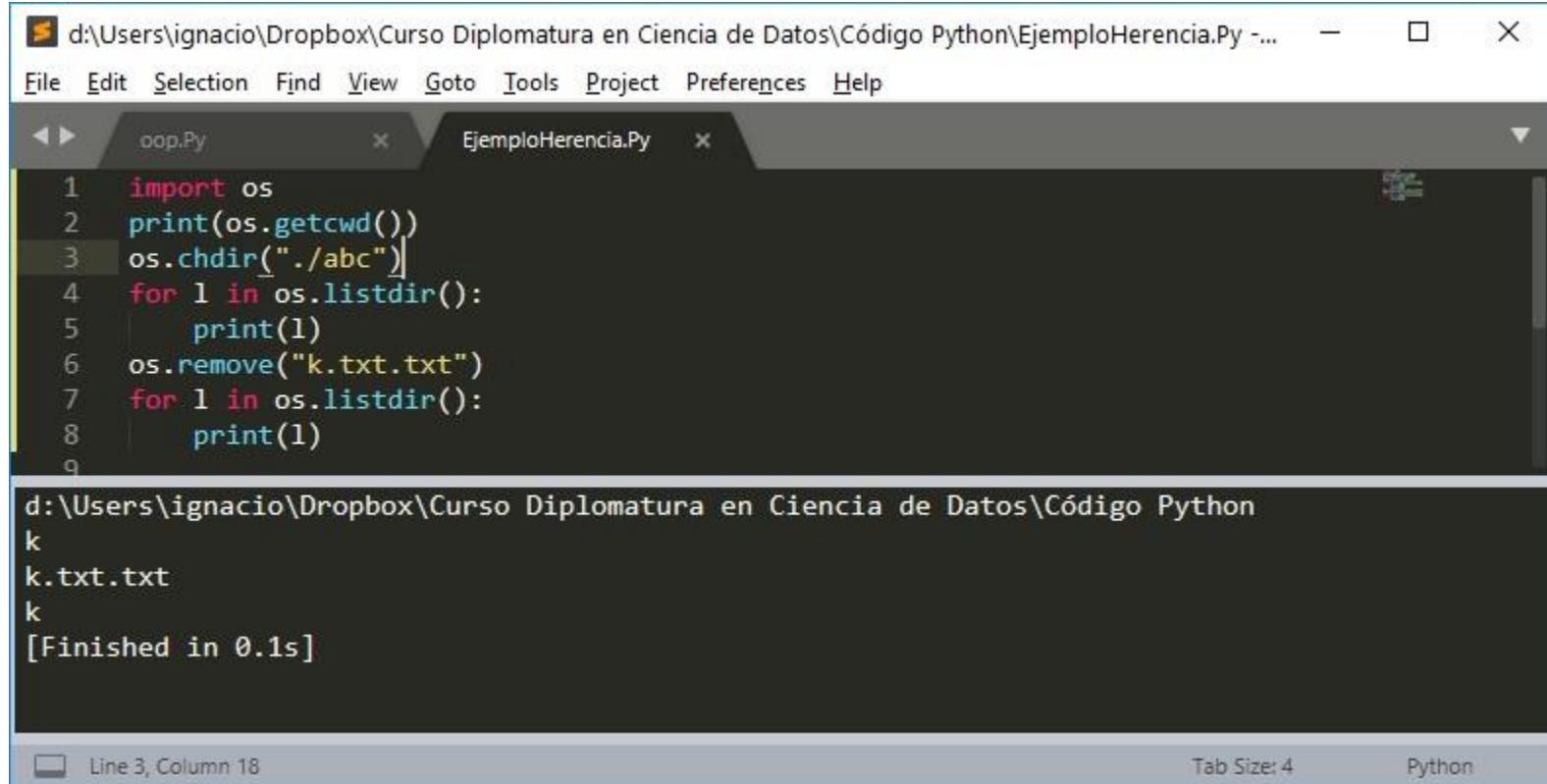
Below the code editor is a terminal window showing the execution results:

```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\abc
[Finished in 0.1s]
```

At the bottom of the terminal window, it says 'Line 5, Column 17; Build finished'.

remove:

- Borra un archivo:



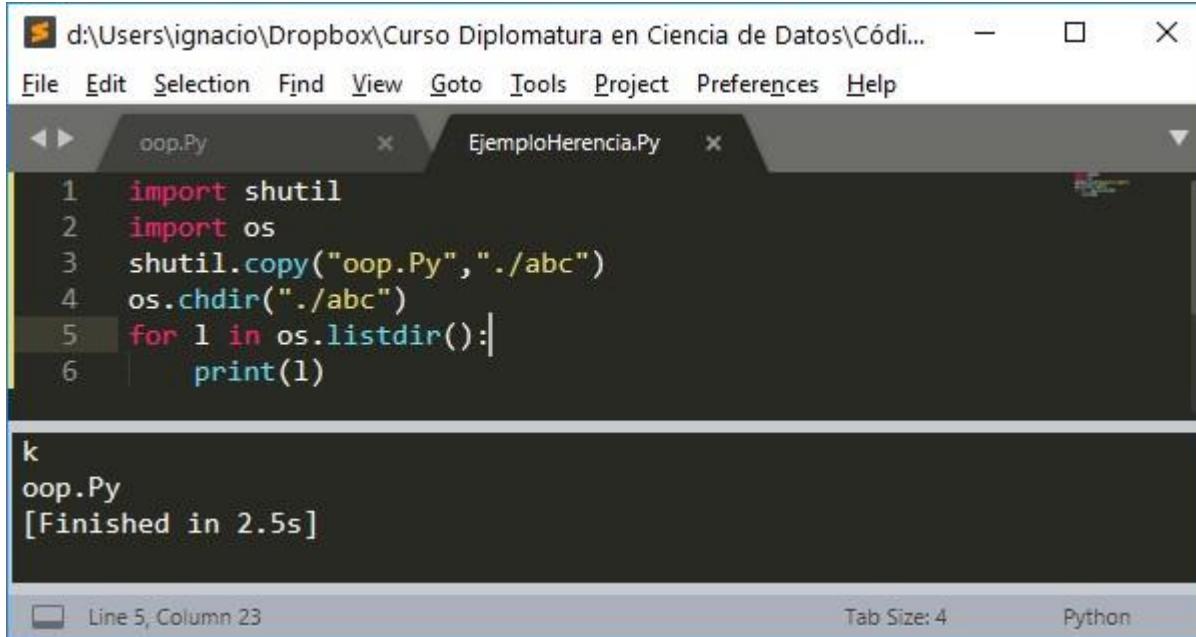
```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\EjemploHerencia.Py ... - □ X
File Edit Selection Find View Goto Tools Project Preferences Help
oop.Py x EjemploHerencia.Py x
1 import os
2 print(os.getcwd())
3 os.chdir("./abc")
4 for l in os.listdir():
5     print(l)
6 os.remove("k.txt.txt")
7 for l in os.listdir():
8     print(l)
9
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python
k
k.txt.txt
k
[Finished in 0.1s]

Line 3, Column 18 Tab Size: 4 Python
```

Módulo *shutil*:

copy()
copy2()
rmtree()
movedir(
)
copydir()
shutil.co
py():

Copia sin metadatos copy(archivo origen, archivo destino)



The screenshot shows a Python code editor interface. The title bar indicates the path: d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código... The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. There are two tabs open: 'oop.py' and 'EjemploHerencia.py'. The 'oop.py' tab contains the following code:

```
1 import shutil
2 import os
3 shutil.copy("oop.py","./abc")
4 os.chdir("./abc")
5 for l in os.listdir():
6     print(l)
```

In the terminal window below, the command 'k' is entered, followed by the output: 'oop.py [Finished in 2.5s]'. The status bar at the bottom shows 'Line 5, Column 23', 'Tab Size: 4', and 'Python'.

shutil.copy2():

Copia con metadatos copy2(archivo origen, archivo destino)



```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código... - □ X
File Edit Selection Find View Goto Tools Project Preferences Help
oop.py x EjemploHerencia.Py x
1 import shutil
2 import os
3 shutil.copy2("oop.py","./abc")
4 os.chdir("./abc")
5 for l in os.listdir():
6     print(l)

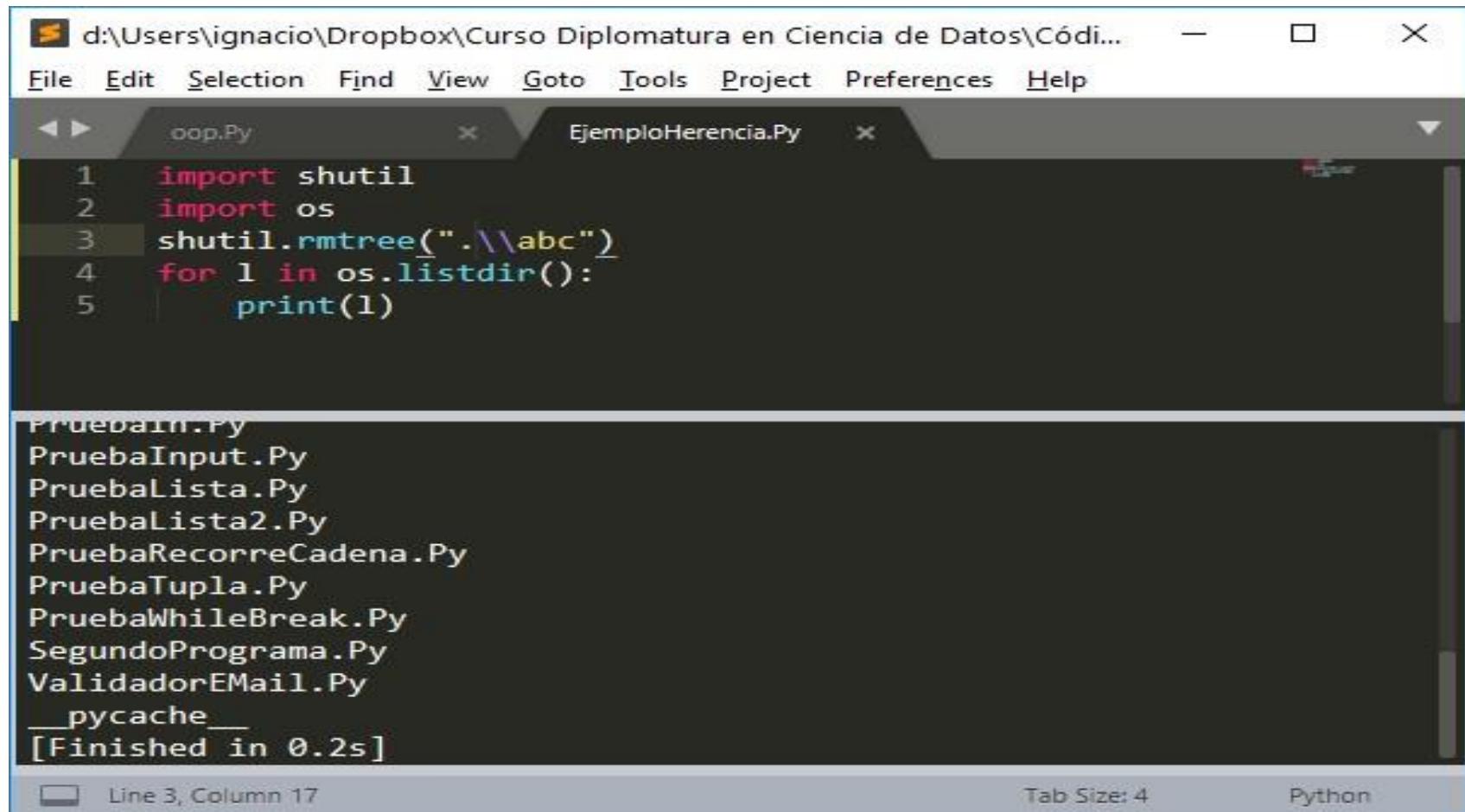
k
oop.py
[Finished in 0.2s]

Line 3, Column 13: Build finished      Tab Size: 4      Python
```

shutil.rmtree():

*Borra un directorio y su contenido
`rmtree(directorio)`*

`shutil.rmtree():`



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Códi... — □ ×

File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py x EjemploHerencia.Py x

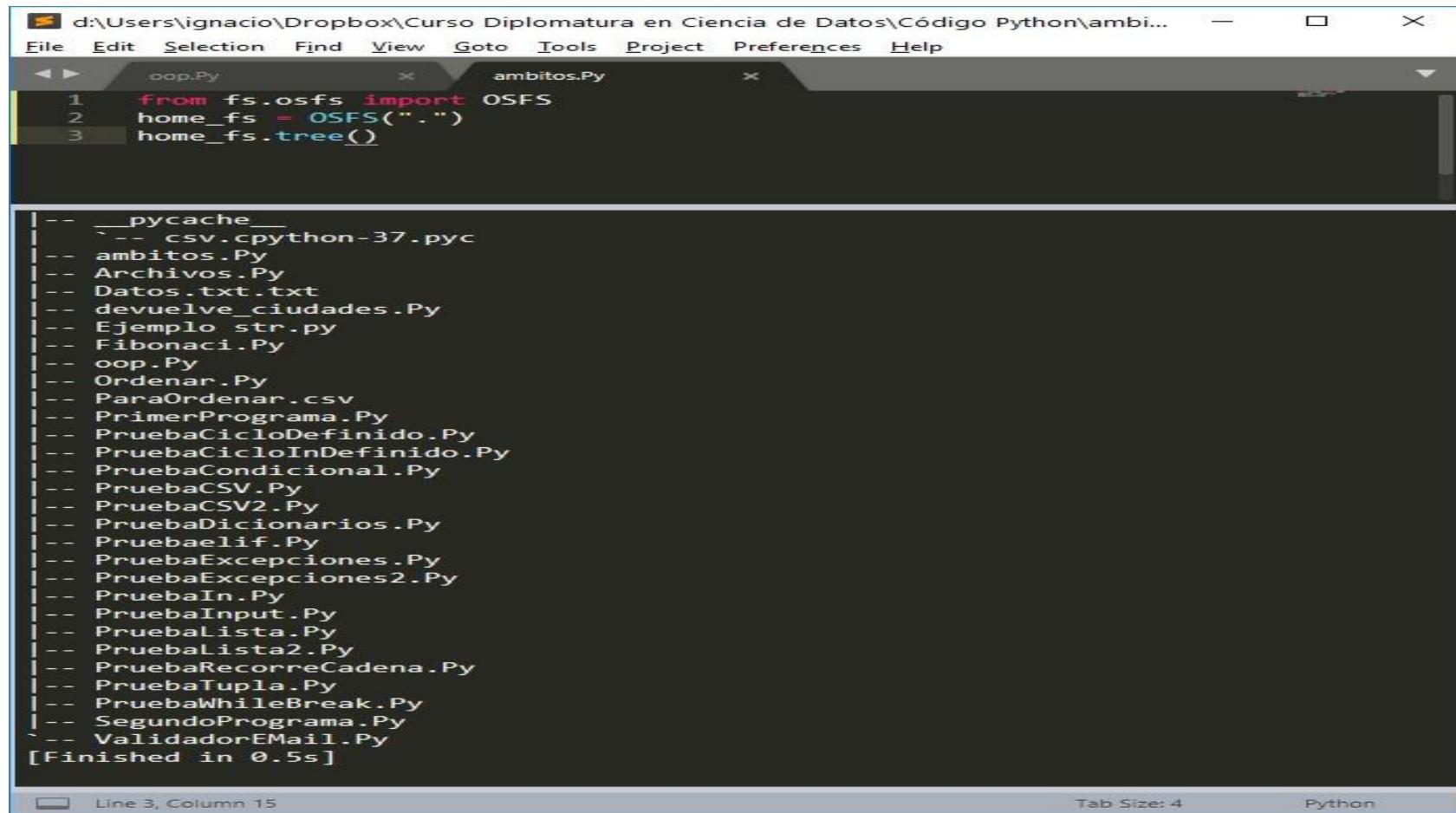
```
1 import shutil
2 import os
3 shutil.rmtree(".\\abc")
4 for l in os.listdir():
5     print(l)
```

PruebaIn.Py
PruebaInput.Py
PruebaLista.Py
PruebaLista2.Py
PruebaRecorreCadena.Py
PruebaTupla.Py
PruebaWhileBreak.Py
SegundoPrograma.Py
ValidadorEMail.Py
__pycache__
[Finished in 0.2s]

Line 3, Column 17 Tab Size: 4 Python

Módulo fs

```
tree()  
close()  
scandir()  
) is_dir()  
is_file()  
getsize()  
tree()
```



The screenshot shows a Python code editor interface. At the top, there are two tabs: 'oop.py' and 'ambitos.py'. The 'ambitos.py' tab is active, displaying the following code:

```
1 from fs.osfs import OSFS
2 home_fs = OSFS(".")
3 home_fs.tree()
```

Below the code editor, a terminal window displays the output of the 'tree' command:

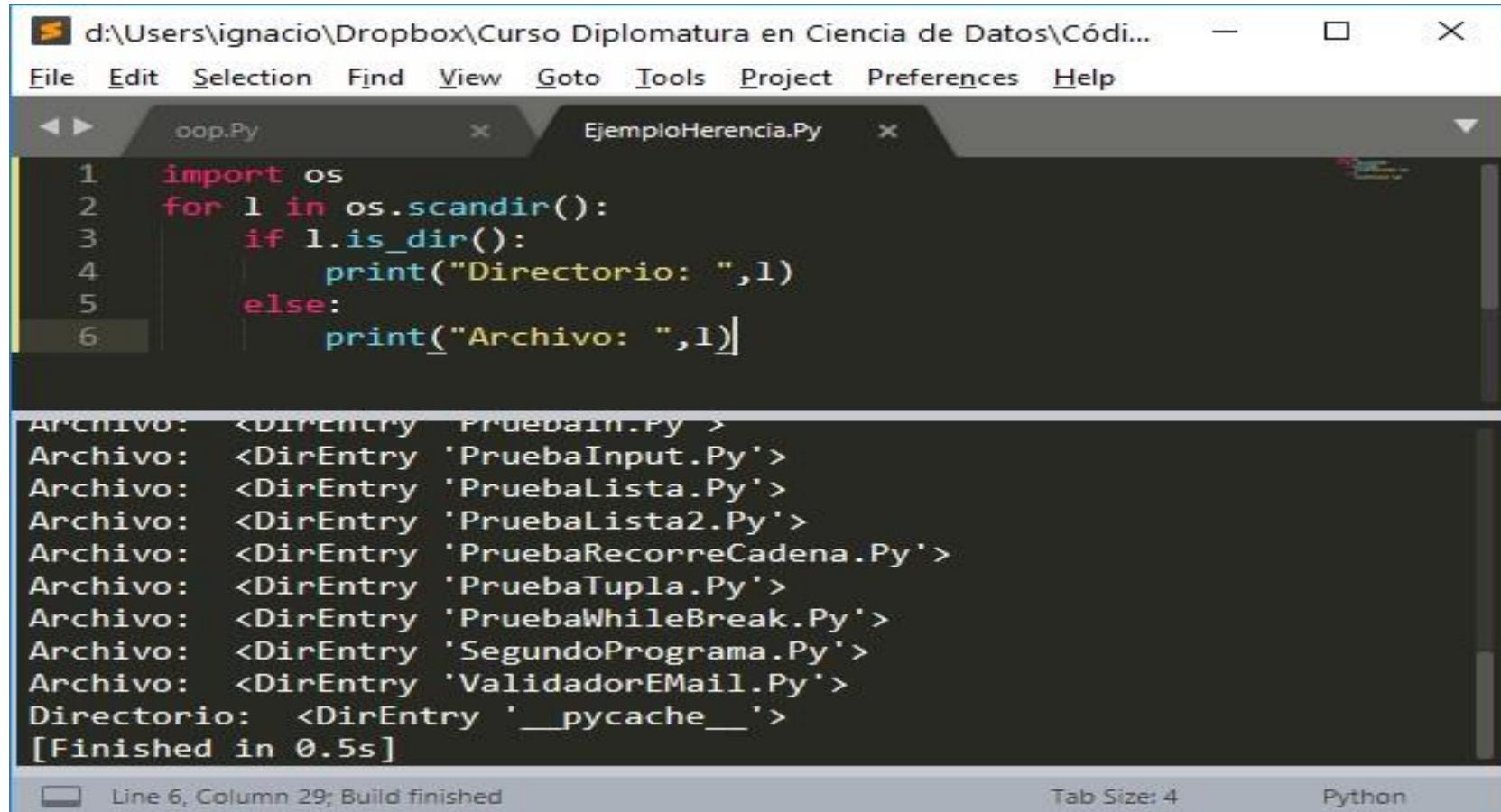
```
-- __pycache__
|-- csv.cpython-37.pyc
|-- ambitos.py
|-- Archivos.py
|-- Datos.txt
|-- devuelve_ciudades.py
|-- Ejemplo_str.py
|-- Fibonaci.py
|-- oop.py
|-- Ordenar.py
|-- ParaOrdenar.csv
|-- PrimerPrograma.py
|-- PruebaCicloDefinido.py
|-- PruebaCicloIndefinido.py
|-- PruebaCondicional.py
|-- PruebaCSV.py
|-- PruebaCSV2.py
|-- PruebaDiccionarios.py
|-- PruebaElif.py
|-- PruebaExcepciones.py
|-- PruebaExcepciones2.py
|-- PruebaIn.py
|-- PruebaInput.py
|-- PruebaLista.py
|-- PruebaLista2.py
|-- PruebaRecorreCadena.py
|-- PruebaTupla.py
|-- PruebaWhileBreak.py
`-- SegundoPrograma.py
`-- ValidadorEMail.py
[Finished in 0.5s]
```

At the bottom of the terminal window, it says '[Finished in 0.5s]'. The status bar at the bottom of the editor shows 'Line 3, Column 15', 'Tab Size: 4', and 'Python'.

close()

No agrega valor en un sistema de archivos local.

*Cierra la conexión y libera recursos con sistemas de archivos remotos como FTP
`scandir()`*



The screenshot shows a code editor window with two tabs: 'oop.py' and 'EjemploHerencia.Py'. The 'oop.py' tab contains the following Python code:

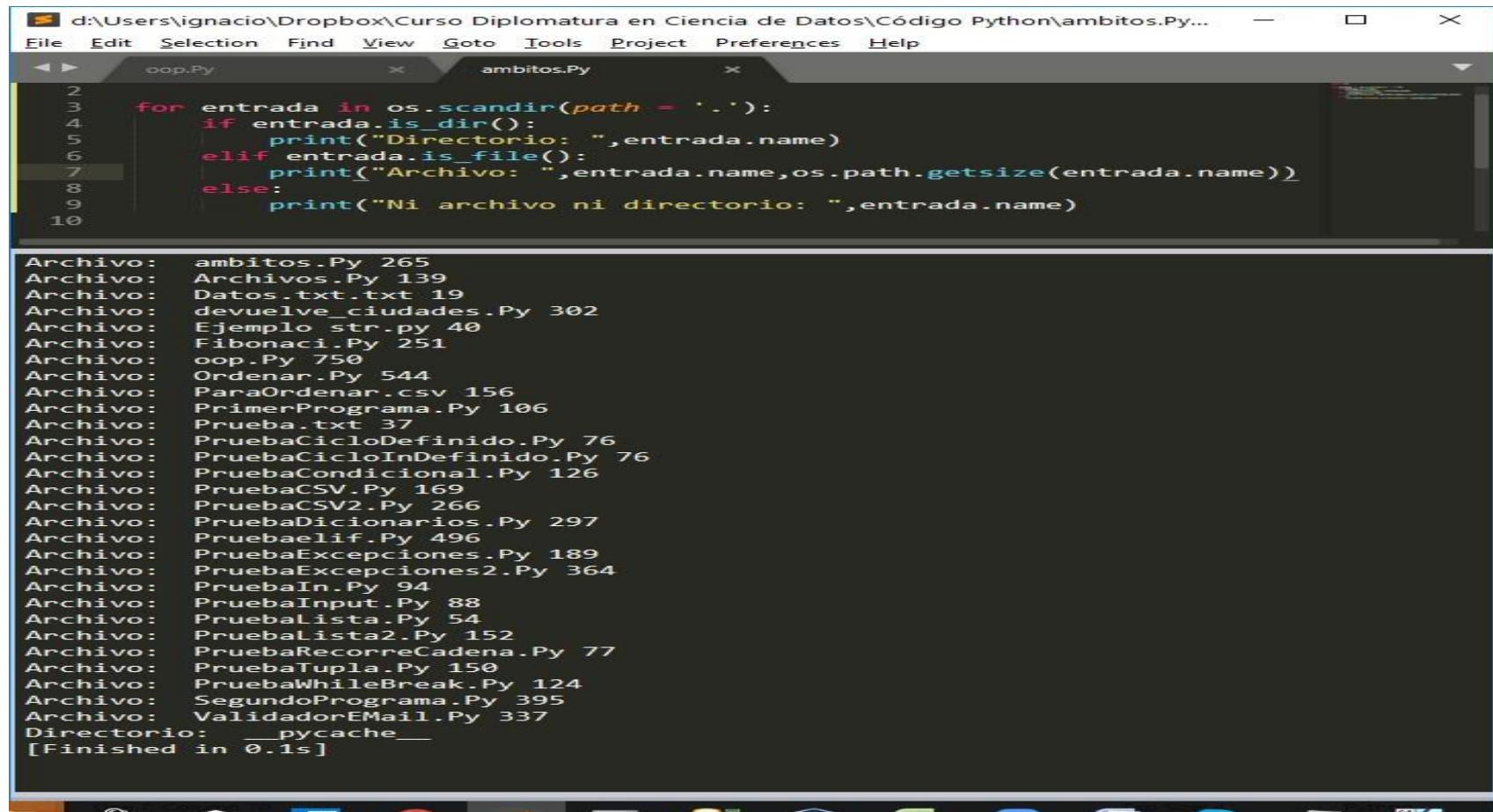
```
1 import os
2 for l in os.scandir():
3     if l.is_dir():
4         print("Directorio: ",l)
5     else:
6         print("Archivo: ",l)
```

The terminal window below displays the output of running this script, listing various files and directories in the current directory:

```
Archivo: <DirEntry 'PruebaInPy'>
Archivo: <DirEntry 'PruebaInput.Py'>
Archivo: <DirEntry 'PruebaLista.Py'>
Archivo: <DirEntry 'PruebaLista2.Py'>
Archivo: <DirEntry 'PruebaRecorreCadena.Py'>
Archivo: <DirEntry 'PruebaTupla.Py'>
Archivo: <DirEntry 'PruebaWhileBreak.Py'>
Archivo: <DirEntry 'SegundoPrograma.Py'>
Archivo: <DirEntry 'ValidadorEMail.Py'>
Directorio: <DirEntry '__pycache__'>
[Finished in 0.5s]
```

At the bottom of the terminal window, status information is shown: 'Line 6, Column 29; Build finished', 'Tab Size: 4', and 'Python'.

getsize()



```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\ambitos.Py...
File Edit Selection Find View Goto Tools Project Preferences Help
oop.py ambitos.py
2
3     for entrada in os.scandir(path = '..'):
4         if entrada.is_dir():
5             print("Directorio: ",entrada.name)
6         elif entrada.is_file():
7             print("Archivo: ",entrada.name,os.path.getsize(entrada.name))
8         else:
9             print("Ni archivo ni directorio: ",entrada.name)
10

Archivo: ambitos.py 265
Archivo: Archivos.py 139
Archivo: Datos.txt.txt 19
Archivo: devuelve_ciudades.py 302
Archivo: Ejemplo_str.py 40
Archivo: Fibonaci.py 251
Archivo: oop.py 750
Archivo: Ordenar.py 544
Archivo: ParaOrdenar.csv 156
Archivo: PrimerPrograma.py 106
Archivo: Prueba.txt 37
Archivo: PruebaCicloDefinido.py 76
Archivo: PruebaCicloIndefinido.py 76
Archivo: PruebaCondicional.py 126
Archivo: PruebaCSV.py 169
Archivo: PruebaCSV2.py 266
Archivo: PruebaDiccionarios.py 297
Archivo: PruebaElif.py 496
Archivo: PruebaExcepciones.py 189
Archivo: PruebaExcepciones2.py 364
Archivo: PruebaIn.py 94
Archivo: PruebaInput.py 88
Archivo: PruebaLista.py 54
Archivo: PruebaLista2.py 152
Archivo: PruebaRecorreCadena.py 77
Archivo: PruebaTupla.py 150
Archivo: PruebaWhileBreak.py 124
Archivo: SegundoPrograma.py 395
Archivo: ValidadorEMail.py 337
Directorio: __pycache__
[Finished in 0.1s]
```

walk()

Permite recorrer archivos y directorios a partir de una ruta definida de forma recursiva:

Devuelve:

- La raíz de la entrada que estoy recorriendo
- Una tupla con los nombres de los archivos que cuelgan de esa entrada
- Una tupla con los nombres de los directorios que cuelgan de esa entrada

walk()

Instituto Data Science Argentina



```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Códi... ━ ━ X
```

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py EjemploHerencia.py

```
2     k=0
3     for l in os.walk("c:/"):
4         k=k+1
5         print(k,l)
6         if k>2:
7             break
8
```

```
1 ('c:/', ['$GetCurrent', '$Recycle.Bin', '$WINDOWS.~BT', 'ESD',
'Intel', 'MSOCache', 'PerfLogs', 'Program Files', 'Program Files (
x86)', 'ProgramData', 'Recovery', 'System Volume Information', 'Users',
'Windows', 'Windows10Upgrade'], ['hiberfil.sys', 'pagefile.sys',
'swapfile.sys'])
2 ('c:/$GetCurrent', ['Logs', 'media', 'SafeOS'], [])
3 ('c:/$GetCurrent\\Logs', [],
['downlevel_2019_09_13_14_37_23_220.log',
'rollback_2019_09_13_16_10_17_486.log'])
[Finished in 0.1s]
```

Line 5, Column 13 Tab Size: 4 Python

Archivos planos

open()

read()

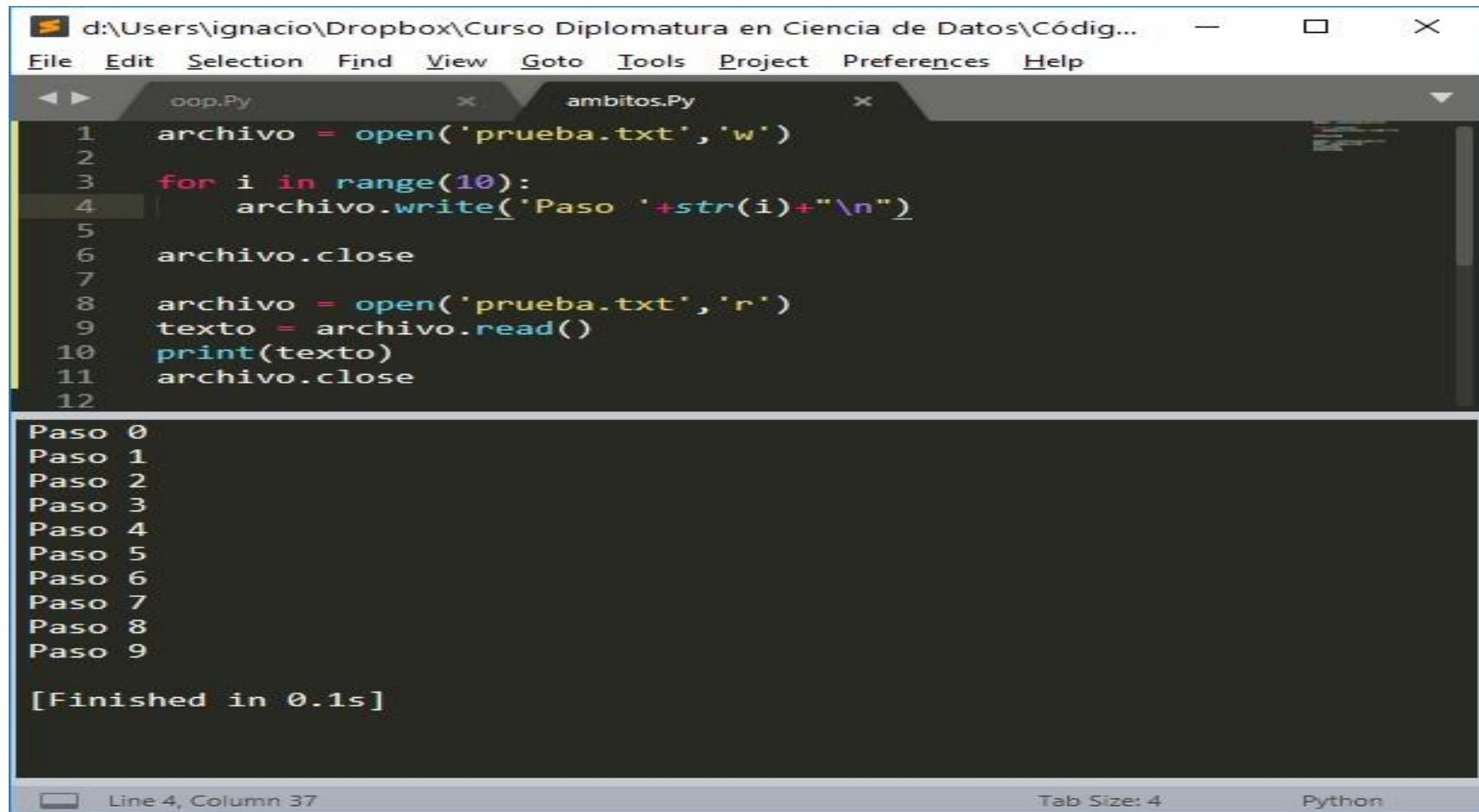
write()

close()

Archiv

os

planos



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código...

File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py ambitos.Py

```
1 archivo = open('prueba.txt','w')
2
3 for i in range(10):
4     archivo.write('Paso '+str(i)+"\n")
5
6 archivo.close
7
8 archivo = open('prueba.txt','r')
9 texto = archivo.read()
10 print(texto)
11 archivo.close
12
```

Paso 0
Paso 1
Paso 2
Paso 3
Paso 4
Paso 5
Paso 6
Paso 7
Paso 8
Paso 9

[Finished in 0.1s]

Line 4, Column 37 Tab Size: 4 Python



The screenshot shows a code editor window with two tabs: 'oop.py' and 'ColectaDirectorio.py'. The 'ColectaDirectorio.py' tab is active, displaying the following Python code:

```
1 import os
2
3 a = open("directorios.txt","w")
4 for d,ds,fs in os.walk("c:/"):
5     for drs in ds:
6         a.write(d+'/'+drs+'\n')
7     for fil in fs:
8         a.write(d+'/'+fil+'\n')
9 a.close()
10 print("Terminé!")
```

The status bar at the bottom indicates 'Line 8, Column 30'. The bottom right corner of the status bar shows 'Python'.

Búsqueda de patrones de texto:

Instituto Data Science Argentina

Módulo *re*

Expresiones regulares:

1. *Compilar el patrón*
2. *Buscarlo*

Expresiones regulares: match()

Instituto Data Science Argentina



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código...

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py ambitos.py

```
1 import re
2
3 patron = re.compile('a')
4
5 print(patron.match('amigo'))
6 print(patron.match('pepa'))
7 print(patron.match('Pepe'))
8
9
10
```

```
<re.Match object; span=(0, 1), match='a'>
None
None
[Finished in 0.1s]
```

Line 8, Column 1 Tab Size: 4 Python

Expresiones regulares: search()



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código...

File Edit Selection Find View Goto Tools Project Preferences Help

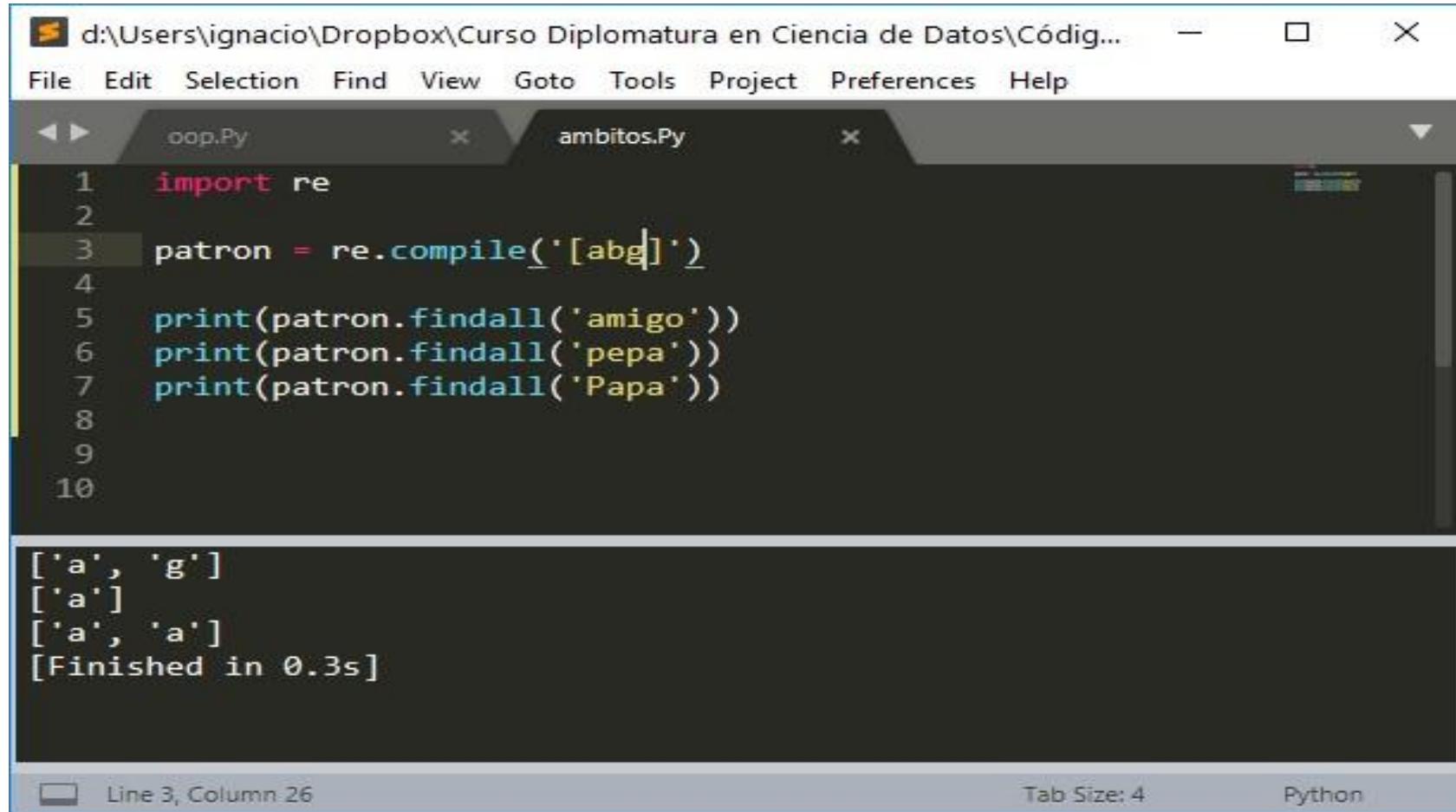
oop.py ambitos.py

```
1 import re
2
3 patron = re.compile('a')
4
5 print(patron.search('amigo'))
6 print(patron.search('pepa'))
7 print(patron.search('Pepe'))
8
9
10
```

```
<re.Match object; span=(0, 1), match='a'>
<re.Match object; span=(3, 4), match='a'>
None
[Finished in 0.1s]
```

Line 8, Column 1 Tab Size: 4 Python

Expresiones regulares: `findall()` []



The screenshot shows a code editor window with two tabs: 'oop.py' and 'ambitos.py'. The 'ambitos.py' tab is active, displaying the following Python code:

```
1 import re
2
3 patron = re.compile('abg')
4
5 print(patron.findall('amigo'))
6 print(patron.findall('pepa'))
7 print(patron.findall('Papa'))
8
9
10
```

Below the code, the output of the script is shown in the terminal window:

```
['a', 'g']
['a']
['a', 'a']
[Finished in 0.3s]
```

At the bottom of the editor window, status bars indicate 'Line 3, Column 26' and 'Tab Size: 4'. The word 'Python' is also visible in the bottom right corner.

Expresiones regulares: `findall()` [^]



The screenshot shows a code editor window with two tabs: 'oop.py' and 'ambitos.py'. The 'ambitos.py' tab is active, displaying the following Python code:

```
1 import re
2
3 patron = re.compile('^[abcdefghijklmnoprstuvwxyz]')
4
5 print(patron.findall('amigo'))
6 print(patron.findall('pepa'))
7 print(patron.findall('Papa'))
8
9
10
```

The output window below the code shows the results of running the script:

```
[]  
[]  
['P']  
[Finished in 0.3s]
```

At the bottom of the editor window, status bars indicate 'Line 3, Column 50', 'Tab Size: 4', and 'Python'.

Metacaracteres:

- ^ inicio de línea.
- \$ fin de línea.
- \A inicio de texto.
- \Z fin de texto.
- . cualquier carácter en la línea.
- \b encuentra límite de palabra.
- \B encuentra distinto a límite de palabra.

Metacaracteres:

- \w un carácter alfanumérico (incluye "_").
- \W un carácter no alfanumérico.
- \d un carácter numérico.
- \D un carácter no numérico.
- \s cualquier espacio (lo mismo que [\t\n\r\f]).
- \S un no espacio.

Iteradores:

- * cero o más, similar a $\{0,\}$.
- + una o más, similar a $\{1,\}$.
- ? cero o una, similar a $\{0,1\}$.
- $\{n\}$ exactamente n veces.
- $\{n,\}$ por lo menos n veces.
- $\{n,m\}$ por lo menos n pero no más de m veces.

Iteradores (cont.):

- *? cero o más, similar a $\{0,\}^*$.
- +? una o más, similar a $\{1,\}^*$.
- ?? cero o una, similar a $\{0,1\}^?$.
- {n}? exactamente n veces.
- {n,}? por lo menos n veces.
- {n,m}? por lo menos n pero no más de m veces.

Orientación a objetos:

- *Motivación*
- *Clases y objetos*
- *Encapsulamiento*
- *Constructor*
- *Herencia*
- *Herencia multiple*

- *Supercargar métodos*

Motivación:

- *En la realidad física las cosas se agrupan porque comparten:*
 - *Estados*
 - *Comportamientos*

- *Si la estructura de mi programa copia a la realidad estaré reutilizando*
 - Propiedades
 - Métodos

Comparación:

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.

Clase automóvil:



The screenshot shows a code editor window with two tabs: 'oop.py' and 'ColectaDirectorio.py'. The 'oop.py' tab is active and displays the following Python code:

```
1 class Coche():
2     marca = 'Toyota'
3     modelo = 'Corolla'
4     ruedas = 4
5     enMarcha = False
6
7     def arrancar(self):
8         self.enMarcha = True
9
10    def parar(self):
11        self.enMarcha = False
12
```

The status bar at the bottom indicates 'Line 10, Column 21', 'Tab Size: 4', and 'Python'.

Vamos a usar el coche:



The screenshot shows a Python code editor with two tabs: 'oop.Py' and 'ColectaDirectorio.Py'. The 'oop.Py' tab contains the following code:

```
1  class Coche():
2      marca = 'Toyota'
3      modelo = 'Corolla'
4      ruedas = 4
5      enMarcha = False
6
7      def arrancar(self):
8          self.enMarcha = True
9
10     def parar(self):
11         self.enMarcha = False
12
13 miCoche = Coche()
14 print("Marca ",miCoche.marca)
```

The output window below the editor shows the execution results:

```
Marca Toyota
[Finished in 0.3s]
```

At the bottom left, it says 'Line 14, Column 30'. At the bottom right, it says 'Tab Size: 4' and 'Python'.

Cambio el estado del coche:



```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Códi...
File Edit Selection Find View Goto Tools Project Preferences Help
oop.py ColectaDirectorio.Py
1 class Coche():
2     marca = 'Toyota'
3     modelo = 'Corolla'
4     ruedas = 4
5     enMarcha = False
6
7     def arrancar(self):
8         self.enMarcha = True
9
10    def parar(self):
11        self.enMarcha = False
12
13 miCoche = Coche()
14 print("Estado ",miCoche.enMarcha)
15 miCoche.arrancar()
16 print("Estado ",miCoche.enMarcha)
17
18
Estado False
Estado True
[Finished in 0.1s]
```

Line 17, Column 1 Tab Size: 4 Python

Forma de cambiar los estados

1. *Tocando directamente las propiedades*
miCoche.marca = “Volkswagen”

2. *Mediante algún método dedicado*
miCoche.arrancar()

Problemas de consistencia

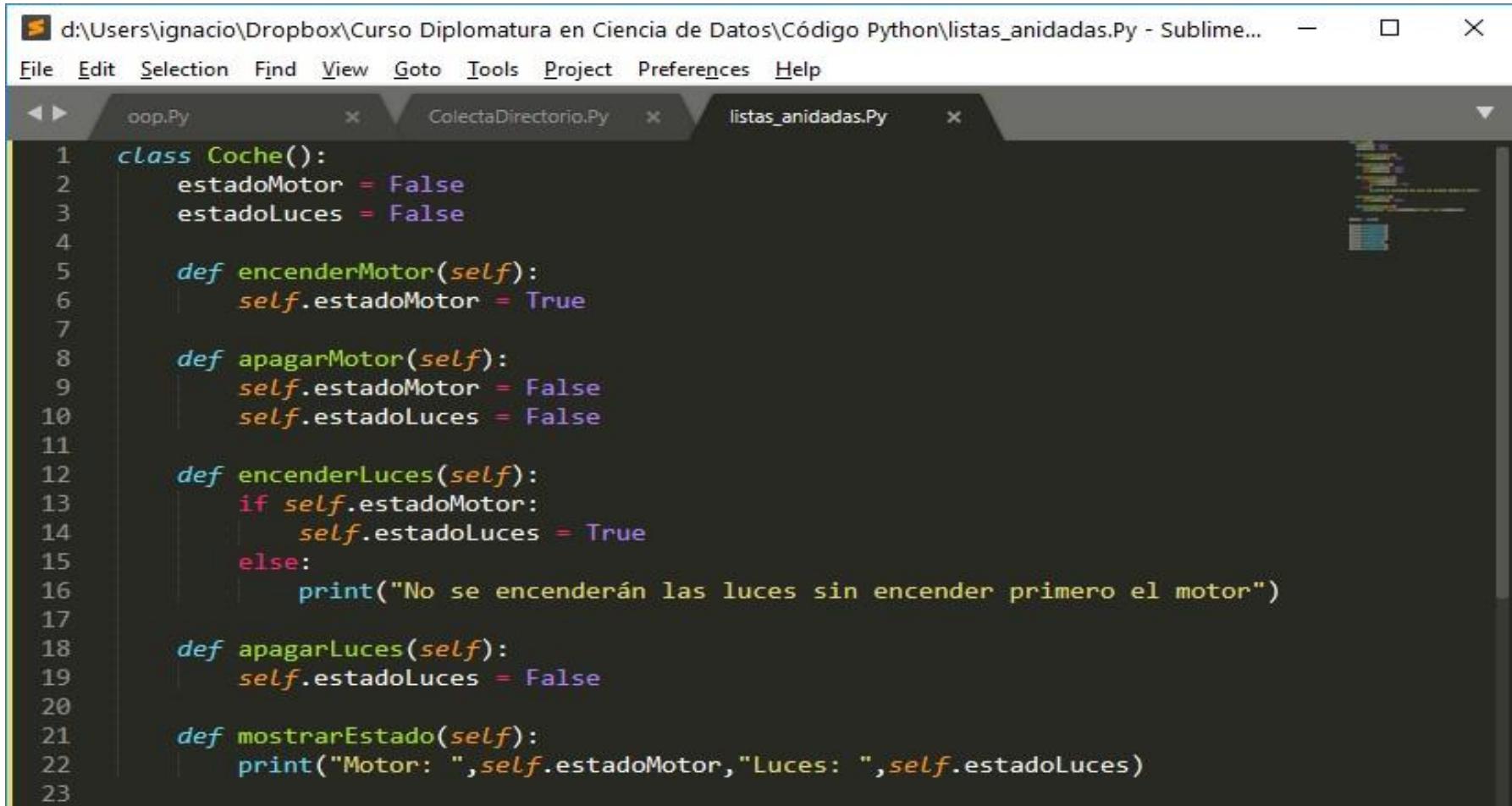
Clase auto

Propiedades:

- *estadoMotor*
- *estadoLuces*

*No queremos dejar encendidas las luces
con el motor apagado!!!*

Defino la clase:



The screenshot shows a Sublime Text window with three tabs: 'oop.py', 'ColectaDirectorio.py', and 'listas_anidadas.py'. The 'listas_anidadas.py' tab is active, displaying the following Python code:

```
1 class Coche():
2     estadoMotor = False
3     estadoLuces = False
4
5     def encenderMotor(self):
6         self.estadoMotor = True
7
8     def apagarMotor(self):
9         self.estadoMotor = False
10    self.estadoLuces = False
11
12    def encenderLuces(self):
13        if self.estadoMotor:
14            self.estadoLuces = True
15        else:
16            print("No se encenderán las luces sin encender primero el motor")
17
18    def apagarLuces(self):
19        self.estadoLuces = False
20
21    def mostrarEstado(self):
22        print("Motor: ",self.estadoMotor,"Luces: ",self.estadoLuces)
23
```

Ejecuto:

```
24
25  miCoche = Coche()
26
27  miCoche.mostrarEstado()
28  miCoche.encenderMotor()
29  miCoche.mostrarEstado()
30  miCoche.encenderLuces()
31  miCoche.mostrarEstado()
32  miCoche.apagarMotor()
33  miCoche.mostrarEstado()
34  miCoche.encenderLuces()
35
Motor: False Luces: False
Motor: True Luces: False
Motor: True Luces: True
Motor: False Luces: False
No se encenderán las luces sin encender primero el motor
[Finished in 0.2s]
```

Ejecuto:

```
24  
25  miCoche = Coche()  
26  
27  miCoche.mostrarEstado()  
28  miCoche.encenderMotor()  
29  miCoche.mostrarEstado()  
30  miCoche.encenderLuces()  
31  miCoche.mostrarEstado()  
32  miCoche.apagarMotor()  
33  miCoche.mostrarEstado()  
34  miCoche.encenderLuces()  
35
```

¿Parece seguro?

```
Motor: False Luces: False  
Motor: True Luces: False  
Motor: True Luces: True  
Motor: False Luces: False  
No se encenderán las luces sin encender primero el motor  
[Finished in 0.2s]
```

Ejecuto:

The screenshot shows a Python code editor with the following details:

- Title Bar:** d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\listas_anidadas.Py - ...
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Code Editor Tabs:** oop.py, ColectaDirectorio.py, listas_anidadas.py
- Code Content (oop.py):**

```
1  class Coche():
2      estadoMotor = False
3      estadoLuces = False
4
5      def encenderMotor(self):
6          self.estadoMotor = True
7
8      def apagarMotor(self):
9          self.estadoMotor = False
10     self.estadoLuces = False
11
12     def encenderLuces(self):
13         if self.estadoMotor:
14             self.estadoLuces = True
15         else:
16             print("No se encenderán las luces sin encender primero el motor")
17
18     def apagarLuces(self):
19         self.estadoLuces = False
20
21     def mostrarEstado(self):
22         print("Motor: ",self.estadoMotor,"Luces: ",self.estadoLuces)
23
24
25 miCoche = Coche()
26
27 miCoche.mostrarEstado()
28 miCoche.estadoLuces = True
29 miCoche.mostrarEstado()
```

- Output Window:**

```
Motor:  False Luces:  False
Motor:  False Luces:  True
[Finished in 0.1s]
```

- Status Bar:** Line 30, Column 1 | Tab Size: 4 | Python

Ejecuto:

Problemas de consistencia

Si quiero proteger la consistencia no puedo permitir que se toquen las propiedades por fuera de los métodos:

Encapsulamiento

Encapsulamiento:

Si quiero proteger una propiedad para que no pueda ser alterada por fuera de los métodos debo comenzarla por __:

Encapsulamiento:

Instituto Data Science Argentina



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\listas_anidadas.Py -

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py ColectaDirectorio.py listas_anidadas.py

```
1 class Coche():
2     __estadoMotor = False
3     __estadoLuces = False
4
5     def encenderMotor(self):
6         self.__estadoMotor = True
7
8     def apagarMotor(self):
9         self.__estadoMotor = False
10    self.__estadoLuces = False
11
12    def encenderLuces(self):
13        if self.__estadoMotor:
14            self.__estadoLuces = True
15        else:
16            print("No se encenderán las luces sin encender primero el motor")
17
18    def apagarLuces(self):
19        self.__estadoLuces = False
20
21    def mostrarEstado(self):
22        print("Motor: ",self.__estadoMotor,"Luces: ",self.__estadoLuces)
```

Encapsulamiento:



The screenshot shows a Python code editor with three tabs: 'oop.py', 'ColectaDirectorio.py', and 'listas_anidadas.py'. The 'listas_anidadas.py' tab is active, displaying the following code:

```
25 miCoche = Coche()
26
27 miCoche.mostrarEstado()
28 miCoche.encenderMotor()
29 miCoche.mostrarEstado()
30 miCoche.encenderLuces()
31 miCoche.mostrarEstado()
32 miCoche.apagarMotor()
33 miCoche.mostrarEstado()
34 miCoche.encenderLuces()
35 miCoche.__estadoLuces = True
36 miCoche.mostrarEstado()
```

The output window below shows the execution results:

```
Motor: False Luces: False
Motor: True Luces: False
Motor: True Luces: True
Motor: False Luces: False
No se encenderán las luces sin encender primero el motor
Motor: False Luces: False
[Finished in 0.1s]
```

At the bottom, status bar indicators show 'Line 37, Column 1', 'Tab Size: 4', and 'Python'.

Encapsulamiento:

Instituto Data Science Argentina



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\listas_anidadas.Py - ... ━ ━ X

File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py ColectaDirectorio.Py listas_anidadas.Py

```
25 miCoche = Coche()
26
27 miCoche.mostrarEstado()
28 miCoche.encenderMotor()
29 miCoche.mostrarEstado()
30 miCoche.encenderLuces()
31 miCoche.mostrarEstado()
32 miCoche.prenderMotor()
33 miCoche.mostrarEstado()
34 miCoche.encenderLuces()
35 miCoche._estadoLuces = True
36 miCoche.mostrarEstado()
```

No da error pero
no enciende las luces

```
Motor: False Luces: False
Motor: True Luces: False
Motor: True Luces: True
Motor: False Luces: False
No se encenderán las luces sin encender primero el motor
Motor: False Luces: False
[Finished in 0.1s]
```

Line 37, Column 1 Tab Size: 4 Python

Constructor:

Instituto Data Science Argentina

Sirve para especificar el estado inicial de un objeto.

Cada vez que se genere un nuevo objeto de esa clase se ejecutará el método constructor.

Debe llamarse `__init__`

Constructor: lo definimos...



The screenshot shows a code editor window with three tabs at the top: 'oop.py', 'ColectaDirectorio.py', and 'listas_anidadas.py'. The 'listas_anidadas.py' tab is active, displaying the following Python code:

```
1  class Coche():
2
3      def __init__(self):
4          self.__estadoMotor = False
5          self.__estadoluces = False
6
7      def encenderMotor(self):
8          self.__estadoMotor = True
9
10     def apagarMotor(self):
11         self.__estadoMotor = False
12         self.__estadoluces = False
13
14     def encenderLuces(self):
15         if self.__estadoMotor:
16             self.__estadoluces = True
17         else:
18             print("No se encenderán las luces sin encender primero el motor")
19
20     def apagarLuces(self):
21         self.__estadoluces = False
22
23     def mostrarEstado(self):
24         print("Motor: ",self.__estadoMotor,"Luces: ",self.__estadoluces)
```

Constructor: lo ejecutamos...

Instituto Data Science Argentina



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\listas_anidadas.Py - ... ━ ━ X

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py ColectaDirectorio.py listas_anidadas.py

```
25
26
27 miCoche = Coche()
28
29 miCoche.mostrarEstado()
30 miCoche.encenderMotor()
31 miCoche.mostrarEstado()
32 miCoche.encenderLuces()
33 miCoche.mostrarEstado()
34 miCoche.apagarMotor()
35 miCoche.mostrarEstado()
36 miCoche.encenderLuces()
37 miCoche.__estadoluces = True
38 miCoche.mostrarEstado()
```

```
Motor: False Luces: False
Motor: True Luces: False
Motor: True Luces: True
Motor: False Luces: False
No se encenderán las luces sin encender primero el motor
Motor: False Luces: False
[Finished in 0.2s]
```

Line 3, Column 5 Tab Size: 4 Python

Constructor:

Instituto Data Science Argentina

¿Qué gané?

Ya lo vamos a ver en acción cuando implementemos HERENCIA...

¿Y si queremos tener dos coches?:



The screenshot shows a code editor window with three tabs: 'oop.py', 'ColectaDirectorio.py', and 'listas_anidadas.py'. The 'oop.py' tab is active, displaying the following Python code:

```
26
27     miCoche = Coche()
28
29     miCoche2 = miCoche
30
31     miCoche.encenderMotor()
32
33     miCoche.mostrarEstado()
34     miCoche2.mostrarEstado()
35
```

Below the code editor, the terminal output shows the execution results:

```
Motor: True Luces: False
Motor: True Luces: False
[Finished in 0.2s]
```

At the bottom of the editor window, status bars indicate 'Line 33, Column 24', 'Tab Size: 4', and 'Python'.



¿Y si queremos tener dos coches?:

The screenshot shows a Python code editor with three tabs: 'oop.py', 'ColectaDirectorio.py', and 'listas_anidadas.py'. The 'oop.py' tab contains the following code:

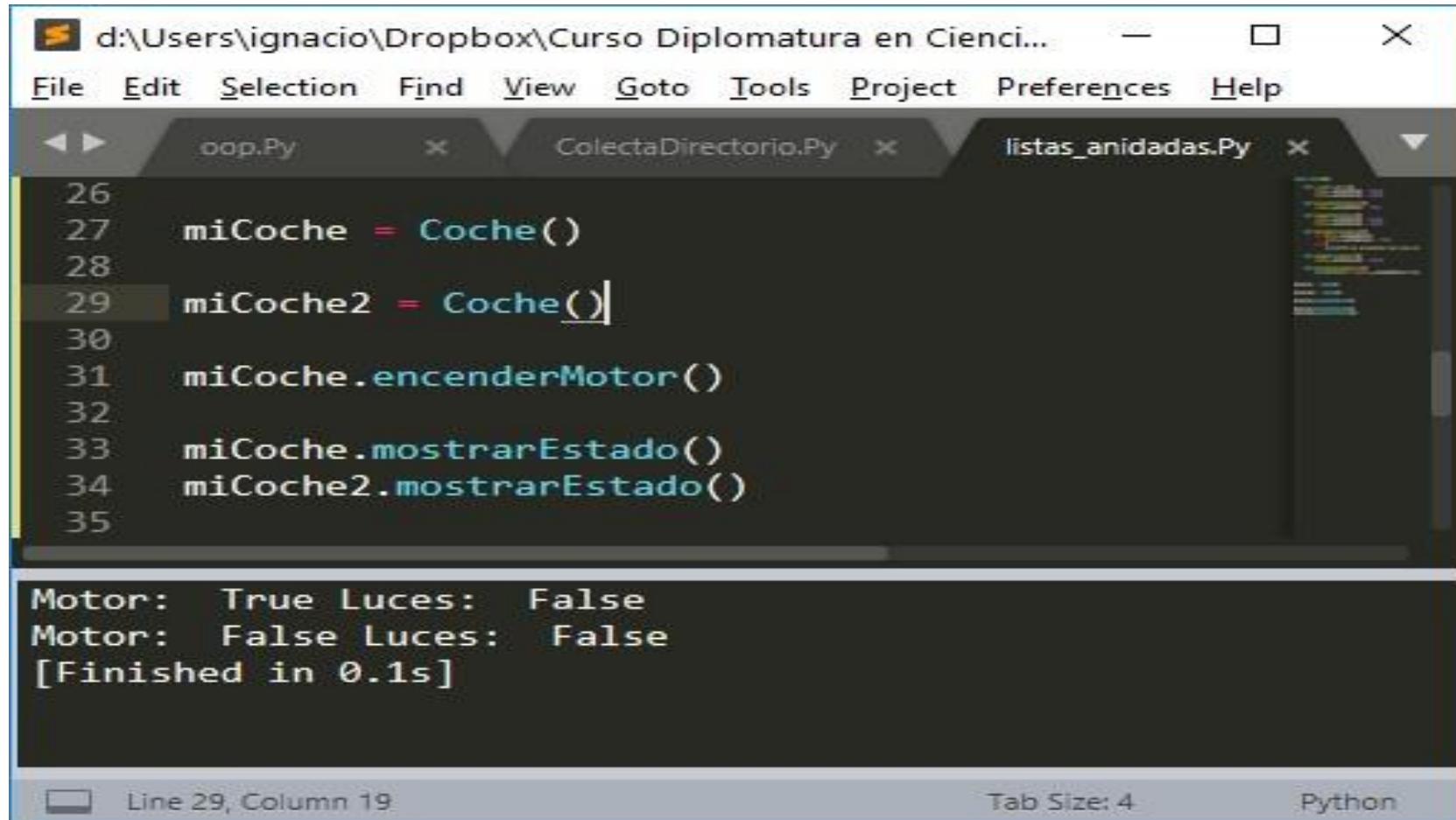
```
26
27     miCoche = Coche()
28
29     miCoche2 = miCoche
30
31     miCoche.encenderMotor()
32
33     miCoche.mostrarEstado()
34     miCoche2.mostrarEstado()
35
```

The output window below shows the results of running the code:

```
Motor: True Luces: False
Motor: True Luces: False
[Finished in 0.2s]
```

A large red watermark diagonal across the slide reads "¿No se puede tener dos coches?".

¿Y si queremos tener dos coches?:



A screenshot of a Python code editor showing a script with two car objects and their methods.

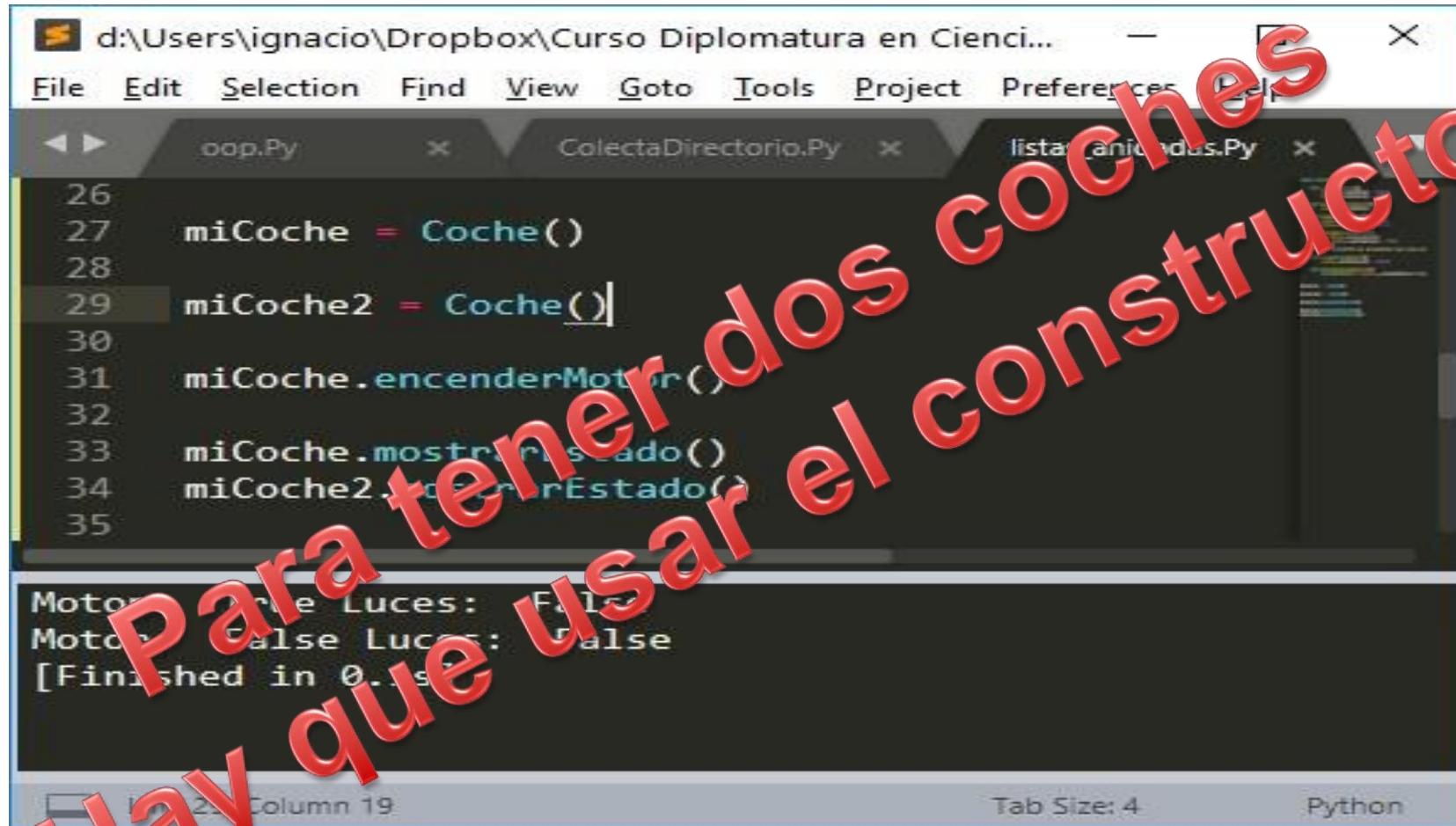
```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Cienci... ━ □ ×
File Edit Selection Find View Goto Tools Project Preferences Help
oop.Py x ColectaDirectorio.Py x listas_anidadadas.Py x
26
27     miCoche = Coche()
28
29     miCoche2 = Coche()
30
31     miCoche.encenderMotor()
32
33     miCoche.mostrarEstado()
34     miCoche2.mostrarEstado()
35

Motor:  True Luces:  False
Motor:  False Luces:  False
[Finished in 0.1s]

Line 29, Column 19   Tab Size: 4   Python
```

The code defines two instances of the `Coche` class, `miCoche` and `miCoche2`. It then calls the `encenderMotor()` method on `miCoche` and the `mostrarEstado()` method on both `miCoche` and `miCoche2`. The output shows the state of each car's motor and lights after the operations.

¿Y si queremos tener dos coches?:



The screenshot shows a Python code editor with three tabs: 'oop.py', 'ColectaDirectorio.py', and 'listarArchivos.py'. The 'oop.py' tab contains the following code:

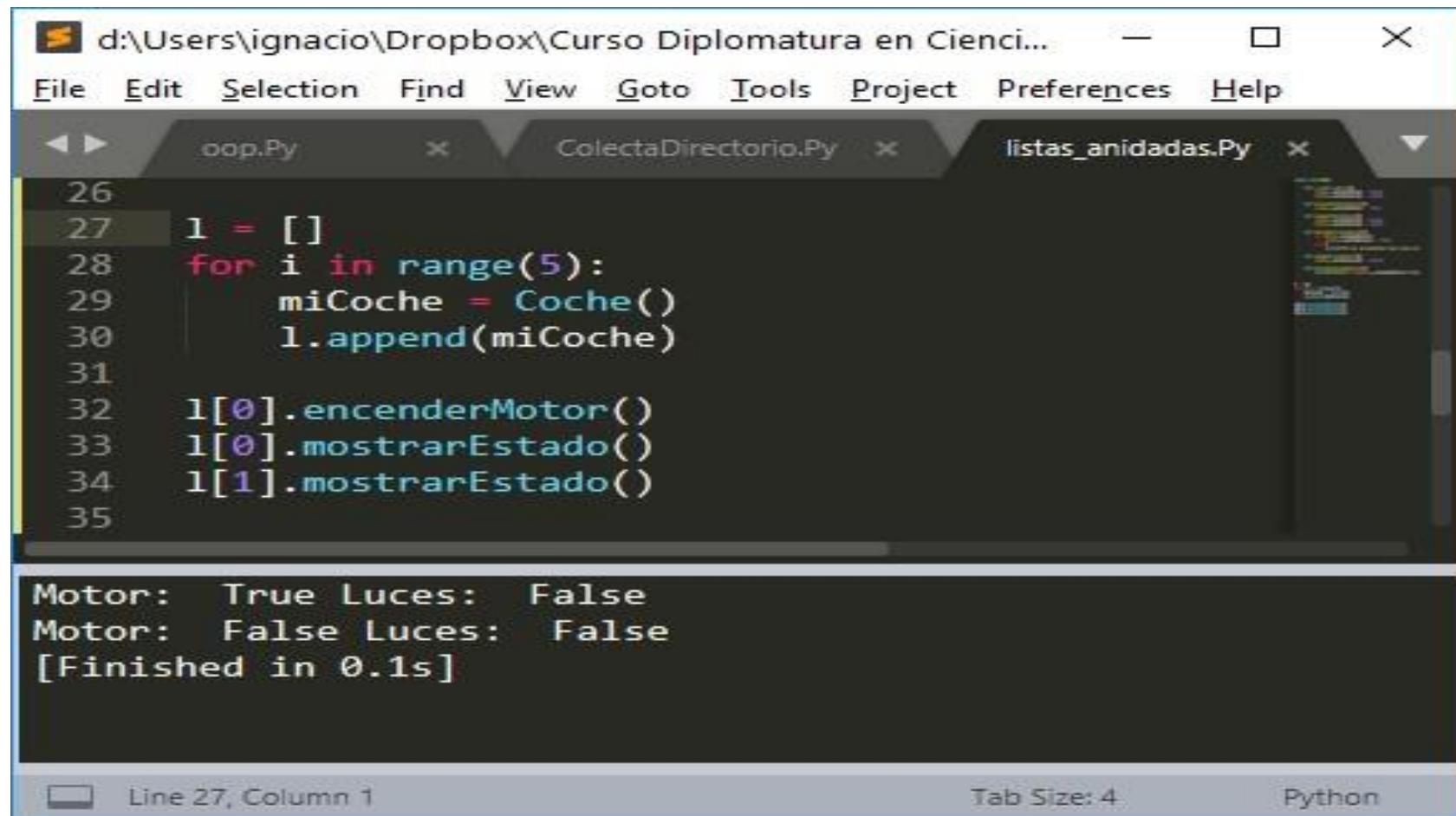
```
26  
27     miCoche = Coche()  
28  
29     miCoche2 = Coche()  
30  
31     miCoche.encenderMotor()  
32  
33     miCoche.mostrarEstado()  
34     miCoche2.mostrarEstado()  
35
```

The output window below shows the results of running the code:

```
Motor: True Luces: False  
Motor: False Luces: False  
[Finished in 0.5s]
```

A large red watermark diagonally across the image reads "Para tener dos coches hay que usar el constructor".

¿Y si queremos una lista?:



The screenshot shows a Python code editor with three tabs: 'oop.Py', 'ColectaDirectorio.Py', and 'listas_anidadadas.Py'. The 'listas_anidadadas.Py' tab is active, displaying the following code:

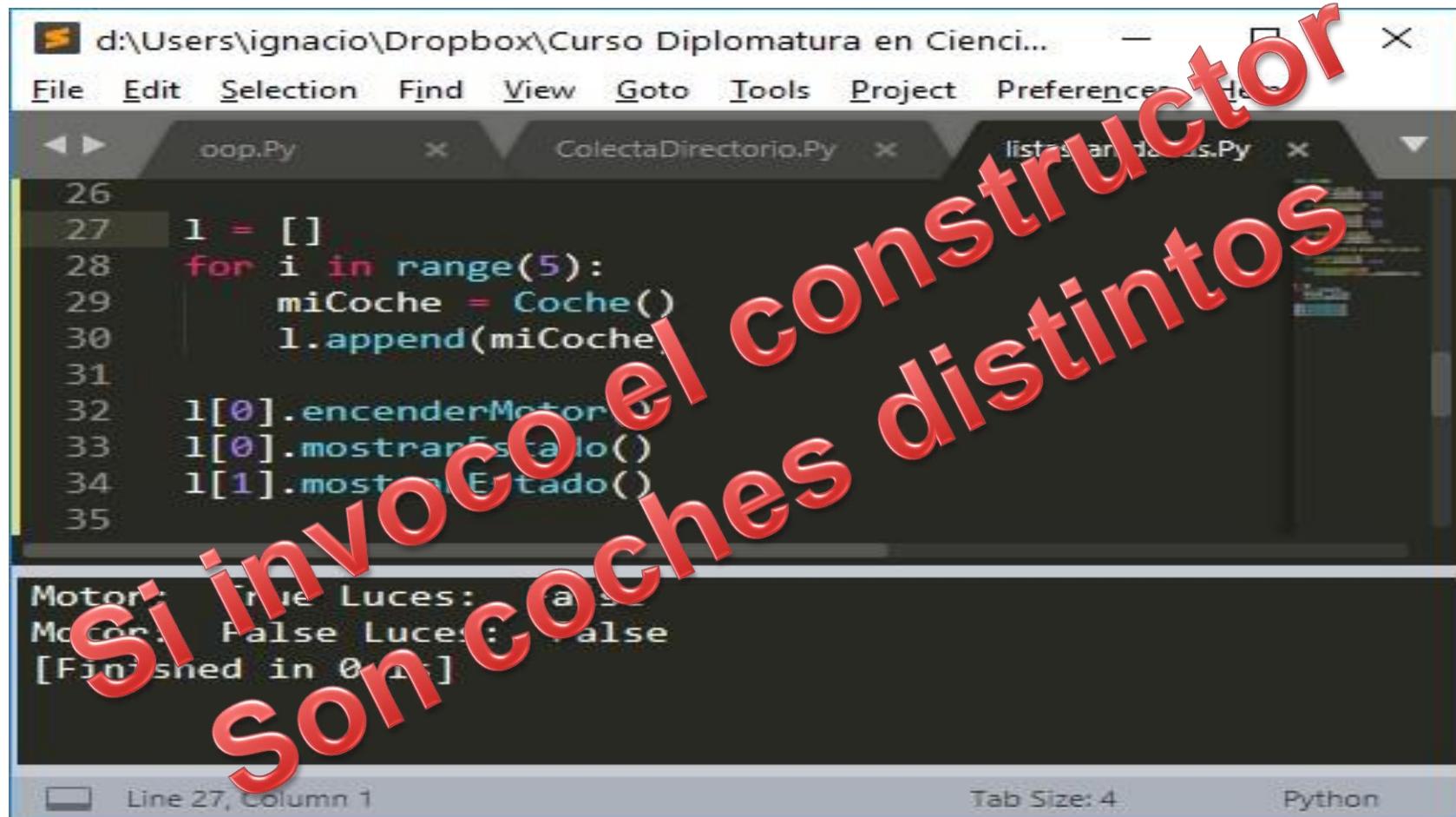
```
26
27 l = []
28 for i in range(5):
29     miCoche = Coche()
30     l.append(miCoche)
31
32 l[0].encenderMotor()
33 l[0].mostrarEstado()
34 l[1].mostrarEstado()
35
```

The output window below the editor shows the execution results:

```
Motor: True Luces: False
Motor: False Luces: False
[Finished in 0.1s]
```

At the bottom of the editor window, status bars indicate 'Line 27, Column 1', 'Tab Size: 4', and 'Python'.

¿Y si queremos una lista?:



The screenshot shows a Python code editor with three tabs: 'oop.py', 'ColectaDirectorio.py', and 'listas.py'. The 'listas.py' tab contains the following code:

```
26
27 l = []
28 for i in range(5):
29     miCoche = Coche()
30     l.append(miCoche)
31
32 l[0].encenderMotor()
33 l[0].mostrarEstado()
34 l[1].mostrarEstado()
35
```

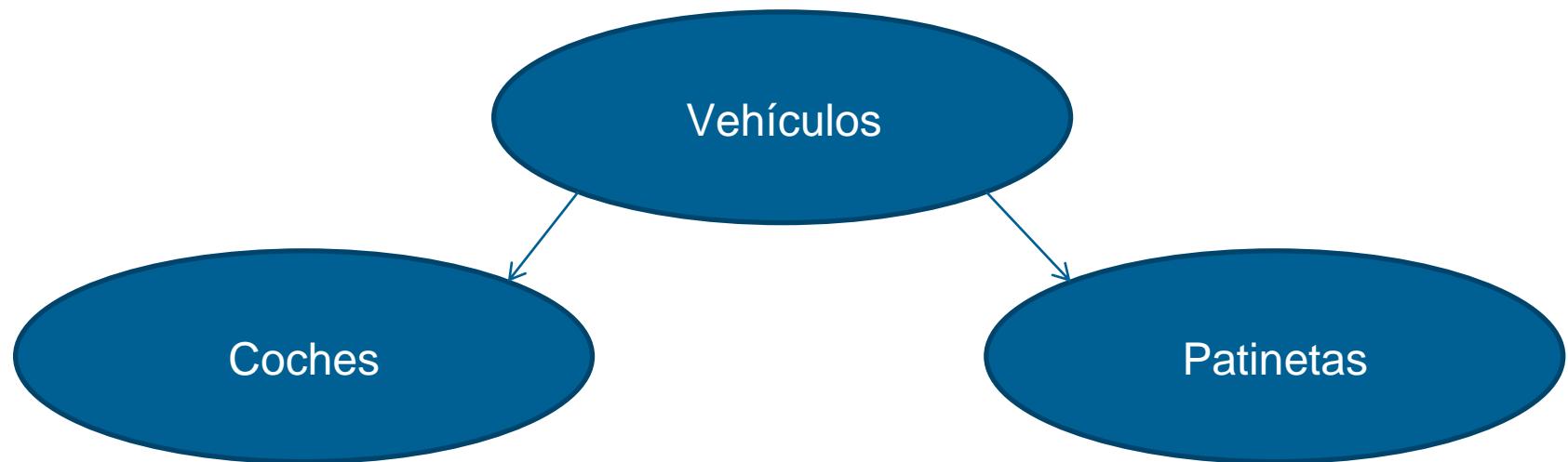
The output window below shows the results of running the code:

```
Motor: True Luces: False
Motor: False Luces: False
[Finished in 0.1s]
```

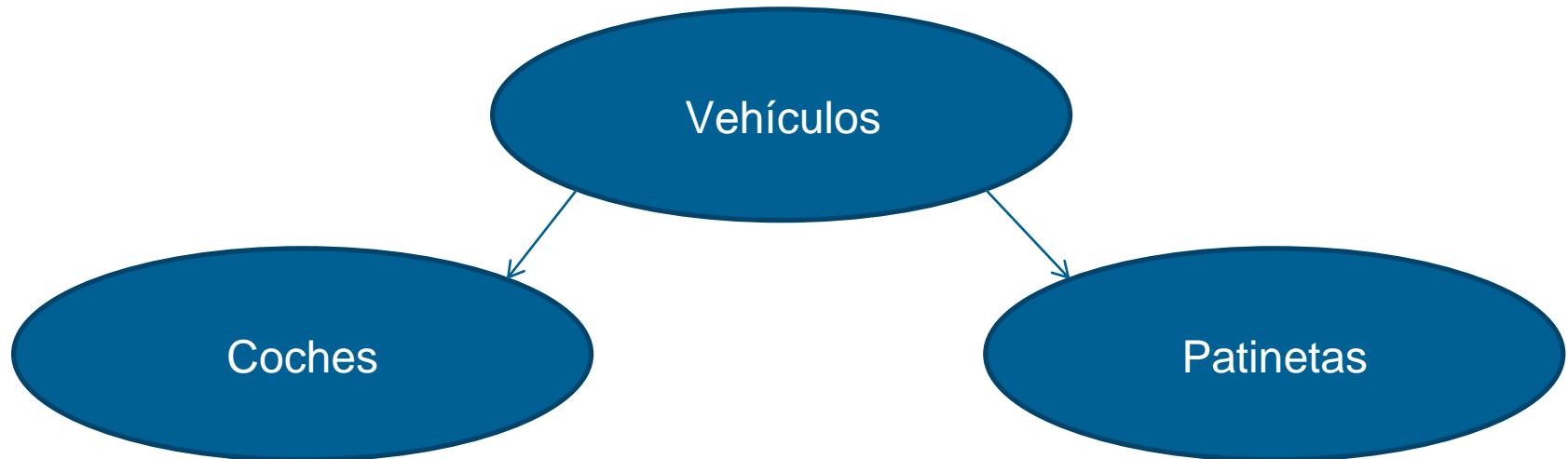
A large red watermark diagonally across the slide reads: "Si invoco el constructor Son coches distintos".

Herencia:

Tenemos una jerarquía de objetos:



¿Qué es una jerarquía?:



Todos tienen cosas en común:

Pero, los coches y patinetas tienen cosas específicas de cada una que los vehículos quizá no tienen

Herencia:

*Veamos por
partes:*

- Marca*
- Modelo*





Herencia:

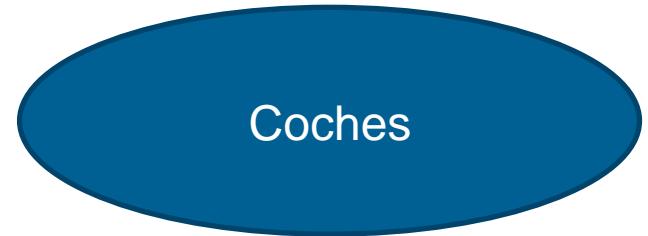
*Veamos por
partes:*

- Marca
- Modelo
- EstadoMotor

Instituto Data Science Argentina

Herencia:

- EstadoLuces*
- EncenderMotor*
- EncenderLuces*
- ApagarMotor*
- ApagarLuces*



*Herencia:
Veamos por
partes:
-Marca
-Modelo
-Estado Rulemanes*





Herencia:

Instituto Data Science Argentina

Herencia:

Vehículos

```
1 class Vehiculo():
2     def __init__(self,marca,modelo):
3         self.__marca = marca
4         self.__modelo = modelo
5
6     def marca(self):
7         return self.__marca
8
9     def modelo(self):
10        return self.__modelo
11
```

Herencia:



Coche

```
12 class Coche(Vehiculo):
13
14     def __init__(self,Marca,Modelo):
15         self.__estadomotor = False
16         self.__estadoluces = False
17         super().__init__(Marca,Modelo)
18
19     def encenderMotor(self):
20         self.__estadomotor = True
21
22     def apagarMotor(self):
23         self.__estadomotor = False
24         self.__estadoluces = False
25
26     def encenderLuces(self):
27         if self.__estadomotor:
28             self.__estadoluces = True
29         else:
30             print("No se encenderán las luces sin encender primero el motor")
31
32     def apagarLuces(self):
33         self.__estadoluces = False
34
35     def mostrarEstado(self):
36
37         print("Marca: ",self.marca()," Modelo: ",self.modelo()," Motor: ",self.__estadomotor," Luces: ",self.__estadoluces)
```

Herencia:

Patineta

```
39 class Patineta(Vehiculo):
40
41     def __init__(self,Marca,Modelo):
42         self.__estadoRulemanes = True
43         super().__init__(Marca,Modelo)
44
45     def romperRulemanes(self):
46         self.__estadoRulemanes = False
47
48     def repararRulemanes(self):
49         self.__estadoRulemanes = True
50
51     def mostrarEstado(self):
52         print("Marca: ",self.marca()," Modelo: ",self.modelo()," Estado Rulemanes: ",self.__estadoRulemanes)
```

Herencia: ejecutamos:

```
54
55     l = []
56
57     miCoche = Coche("Ford","Falcon")
58     l.append(miCoche)
59     miCoche = Coche("Toyota","Corolla")
60     l.append(miCoche)
61     miPatineta = Patineta("Rombal","H10")
62     l.append(miPatineta)
63
64     l[0].encenderMotor()
65     l[0].mostrarEstado()
66     l[1].mostrarEstado()
67
68     for o in l:
69         o.mostrarEstado()
```

```
Marca: Ford Modelo: Falcon Motor: True Luces: False
Marca: Toyota Modelo: Corolla Motor: False Luces: False
Marca: Ford Modelo: Falcon Motor: True Luces: False
Marca: Toyota Modelo: Corolla Motor: False Luces: False
Marca: Rombal Modelo: H10 Estado Rulemanes: True
[Finished in 0.1s]
```

Supercargar:

Es invocar al método del padre de un objeto aunque exista un método para el propio objeto con el mismo nombre:

super().metodo(parámetros)

Polimorfismo:

Es invocar un método distinto para objetos distintos aprovechando que el nombre del método coincide

Ejemplo polimorfismo:

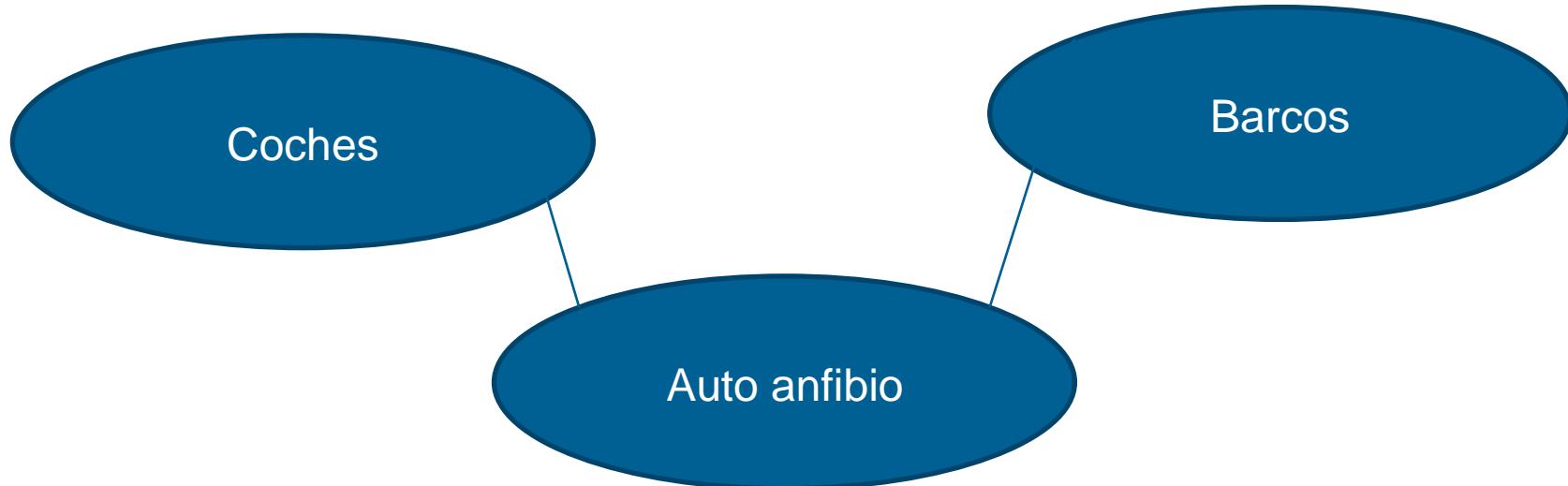
```
54
55     l = []
56
57     miCoche = Coche("Ford","Falcon")
58     l.append(miCoche)
59     miCoche = Coche("Toyota","Corolla")
60     l.append(miCoche)
61     miPatineta = Patineta("Rombal","H10")
62     l.append(miPatineta)
63
64     l[0].encenderMotor()
65     l[0].mostrarEstado()
66     l[1].mostrarEstado()
67
68     for o in l:
69         o.mostrarEstado()
```

```
Marca: Ford Modelo: Falcon Motor: True Luces: False
Marca: Toyota Modelo: Corolla Motor: False Luces: False
Marca: Ford Modelo: Falcon Motor: True Luces: False
Marca: Toyota Modelo: Corolla Motor: False Luces: False
Marca: Rombal Modelo: H10 Estado Rulemanes: True
[Finished in 0.1s]
```

Herencia múltiple:

Tenemos un hijo que va a heredar propiedades y métodos de dos o más padres

class Hijo(Padre1, Padre2,...) Herencia múltiple: Ejemplo:



Vamos a ir clase por clase...

Herencia múltiple:

Primero los padres:

-Profundidad Herencia múltiple:

Veamos por partes:

-EstadoMotor

-EstadoLuces

-EncenderMotor

-EncenderLuces

-ApagarMotor



Barcos



Coches

-ApagarLuces

Herencia múltiple:

Veamos por partes:

- EstadoMotor
- EstadoLuces
- EncenderMotor



- EncenderLuces*
- ApagarMotor*
- ApagarLuces*
- Profundidad*

Herencia Múltiple: Clase Barco

```
21 class Barco():
22
23     def __init__(self):
24         self.__profundidad = 0
25     def ajustarProfundidad(self, profundidad):
26         self.__profundidad = profundidad
27     def consultarProfundidad(self):
28         return self.__profundidad
29     def mostrarEstado(self):
30         print(" Profundidad: ", self.__profundidad)
```

Herencia Múltiple: Clase Coche

```
1 class Coche():
2
3     def __init__(self):
4         self.__estadoMotor = False
5         self.__estadoluces = False
6     def encenderMotor(self):
7         self.__estadoMotor = True
8     def apagarMotor(self):
9         self.__estadoMotor = False
10    self.__estadoluces = False
11    def encenderLuces(self):
12        if self.__estadoMotor:
13            self.__estadoluces = True
14        else:
15            print("No se encenderán las luces sin encender primero el motor")
16    def apagarLuces(self):
17        self.__estadoluces = False
18    def mostrarEstado(self):
19        print(" Motor: ",self.__estadoMotor," Luces: ",self.__estadoluces)
20
```

Herencia Múltiple: Clase Anfibio

```
33 class Anfibio(Coche,Barco):  
34     pass  
35  
36     miAnfibio = Anfibio()  
37  
38     miAnfibio.mostrarEstado()  
39     miAnfibio.ajustarProfundidad(100)  
40     print(miAnfibio.consultarProfundidad())  
41  
42
```

```
Motor: False Luces: False  
100  
[Finished in 0.1s]
```

Manejo de archivos CSV y XLSX

- *Archivos .CSV (Paquete CSV)*
 - Leer
 - Escribir
- *Archivos .XLSX (Paquetes OpenPyxl, XLSXWriter)*
 - Leer
 - Escribir
- *Recomendaciones para la nomenclatura*

Manejo de archivos

CSV:

- *open*
 - *csv.reader* }
 - *csv.writer*
 - *csv.DictReader* }
 - *csv.DictWriter*
- Maneja una lista de filas / columnas
- Maneja un diccionario

- *close*

¿Qué tiene un archivo .CSV?

En Excel lo vemos así:

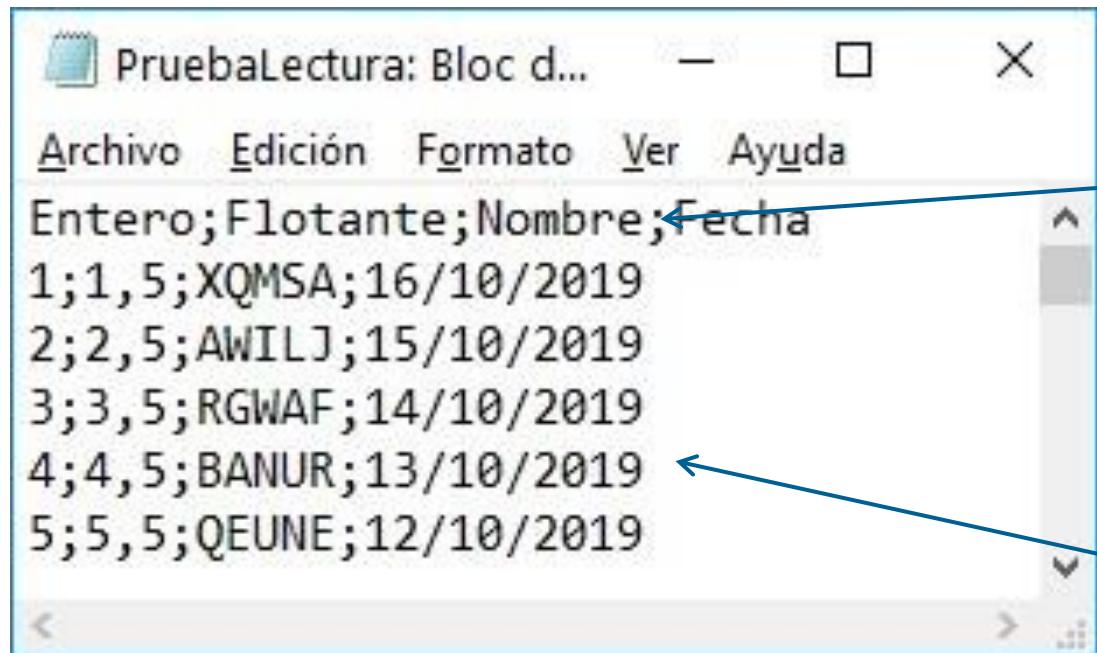
Entero	Flotante	Nombre	Fecha
1	1,5 XQMSA		16/10/2019

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

¿Qué tiene un archivo .CSV?

Pero lo que tiene es:



The screenshot shows a Windows Notepad window titled "PruebaLectura: Bloc d...". The window contains the following text:

```
Archivo Edición Formato Ver Ayuda
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
```

Annotations with arrows point to specific parts of the data:

- A blue arrow points from the label "Separador" to the semicolon character ";" in the header row.
- A blue arrow points from the label "Fin de línea" to the carriage return character "\r" in the footer row of the data.

¿Cómo leo un archivo csv?



The screenshot shows a code editor window with two tabs: 'oop.py' and 'ColectaDirectorio.py'. The 'oop.py' tab is active, displaying the following Python code:

```
1 import csv
2
3 f = open("PruebaLectura.csv","r")
4
5 reader = csv.reader(f,delimiter=";")
6
7 for row in reader:
8     print(row)
9
10 f.close()
11
12
```

¿Qué obtenemos?

```
[ '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' ]
```



Line 2, Column 1

Tab Size: 4



Vamos a escribir un archivo csv:

1. *Primero hay que generar la lista de listas*
 2. *Luego podremos grabarla*
 3. *Finalmente verificar que todo funcionó*
-
1. *Generar la lista de listas:*

Instituto Data Science Argentina

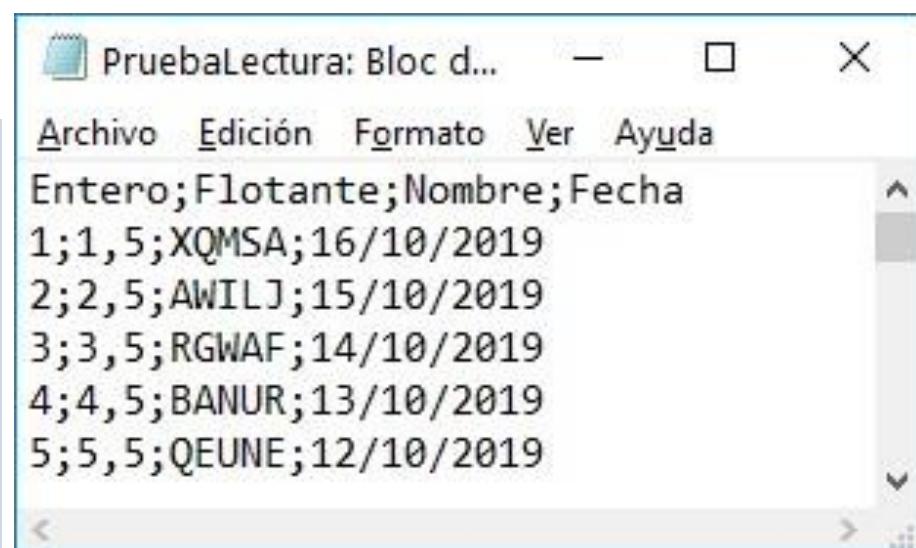
```
1 import csv
2
3 encabezado = ["Entero","Flotante","Texto","Fecha"]
4
5 l = []
6
7 l.append(encabezado)
8
9 for i in range(100):
10     r = []
11     r.append(i)
12     r.append(float(i)+0.5)
13     r.append("abcde")
14     r.append("10/10/2019")
15     l.append(r)
16
```

2. Grabar el archivo csv:

```
18 f2 = open("PruebaEscritura2.csv", "w")
19
20 writer = csv.writer(f2, delimiter=";", lineterminator="\n")
21
22 writer.writerow(1)
23
24 f2.close
```

3. Verificar:

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

Lectura de un archivo csv en diccionarios



```
1 import csv
2
3 f = open("PruebaEscritura2.csv","r")
4
5 reader = csv.DictReader(f, delimiter = ";")
6 print(reader.fieldnames)
7
8 for r in reader:
9     print(r["Entero"],r["Flotante"],r["Texto"],r["Fecha"])
10
11 f.close
12
```

['Entero', 'Flotante', 'Texto', 'Fecha']

0 0.5 abcde 10/10/2019

1 1.5 abcde 10/10/2019

2 2.5 abcde 10/10/2019

3 3.5 abcde 10/10/2019

4 4.5 abcde 10/10/2019

5 5.5 abcde 10/10/2019

Escritura a partir de diccionarios

```
1 import csv
2
3 l = []
4
5 for i in range(100):
6     d = {}
7     d["Entero"] = i
8     d["Flotante"] = i+.5
9     d["Texto"] = "abcdef"
10    d["Fecha"] = "10/10/2019"
11    l.append(d)
12
13
14 f = open("PruebaEscritura3.csv","w")
15
16 writer = csv.DictWriter(f, delimiter = ";", lineterminator = "\n", fieldnames = d.keys())
17 writer.writeheader()
18
19 for r in l:
20     writer.writerow(r)
21
22 f.close()
23
24 print("Escritura completa")
```

Nos toca el Excel

Instalar openpyxl

Modelo de datos de excel

Leer un archivo

Instalar XLSXwriter

Escribir un archivo

Instalar openpyxl
pip install openpyxl

Modelo de datos de Excel

- Workbook
 - Tupla de worksheets
- Tupla de filas
 - Tupla de columnas
 - » Cada columna tiene un valor

Lectura de archivos Excel:



```
1 import openpyxl
2
3 doc = openpyxl.load_workbook('PruebaLecturaExcel.xlsx')
4
5 for s in doc:
6     try:
7         for r in s:
8             try:
9                 for c in r:
10                     print(c.value)
11             except TypeError:
12                 break
13     except TypeError:
14         break
15
16 doc.close()
```

Escritura de archivos excel:

1. *Abro el libro y creo la hoja*
2. *Creo los datos a grabar*
3. *Grabo los datos*
4. *Verifico el resultado*

1. *Abro el libro y creo la hoja:*

Instituto Data Science Argentina



The screenshot shows a code editor window with the following details:

- Title Bar:** d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Escribe...
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Tab Bar:** oop.py (selected), EscribeUnArchivoXLSX.Py
- Code Area:**

```
1 # Programa que genera un archivo xlsx
2
3 import xlsxwriter
4
5 # abro el archivo
6
7 libro = xlsxwriter.Workbook("PruebaEscrituraExcel.xlsx")
8
9 hoja = libro.add_worksheet()
10
```
- Status Bar:** [Finished in 0.3s]
- Bottom Status:** Line 4, Column 1, Tab Size: 4, Python

2. Creo los datos a grabar:



The screenshot shows a code editor window with a dark theme. At the top, there's a menu bar with File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. Below the menu is a tab bar with 'oop.py' and 'EscribeUnArchivoXLSX.Py'. The main area contains the following Python code:

```
11 # Genero la lista de listas con la información a grabar
12
13 s = []
14 l = ['Entero', 'Flotante', 'Texto', 'Fecha']
15 s.append(l)
16 for r in range(100):
17     l = []
18     l.append(r)
19     l.append(float(r)+.5)
20     l.append('abc'+str(r))
21     l.append('10/10/2019')
22     s.append(l)
23
```

At the bottom of the code editor, a status bar displays '[Finished in 0.3s]'. Below the status bar, it says 'Line 4, Column 1' and has tabs for 'Tab Size: 4' and 'Python'.

3. Grabo los datos:

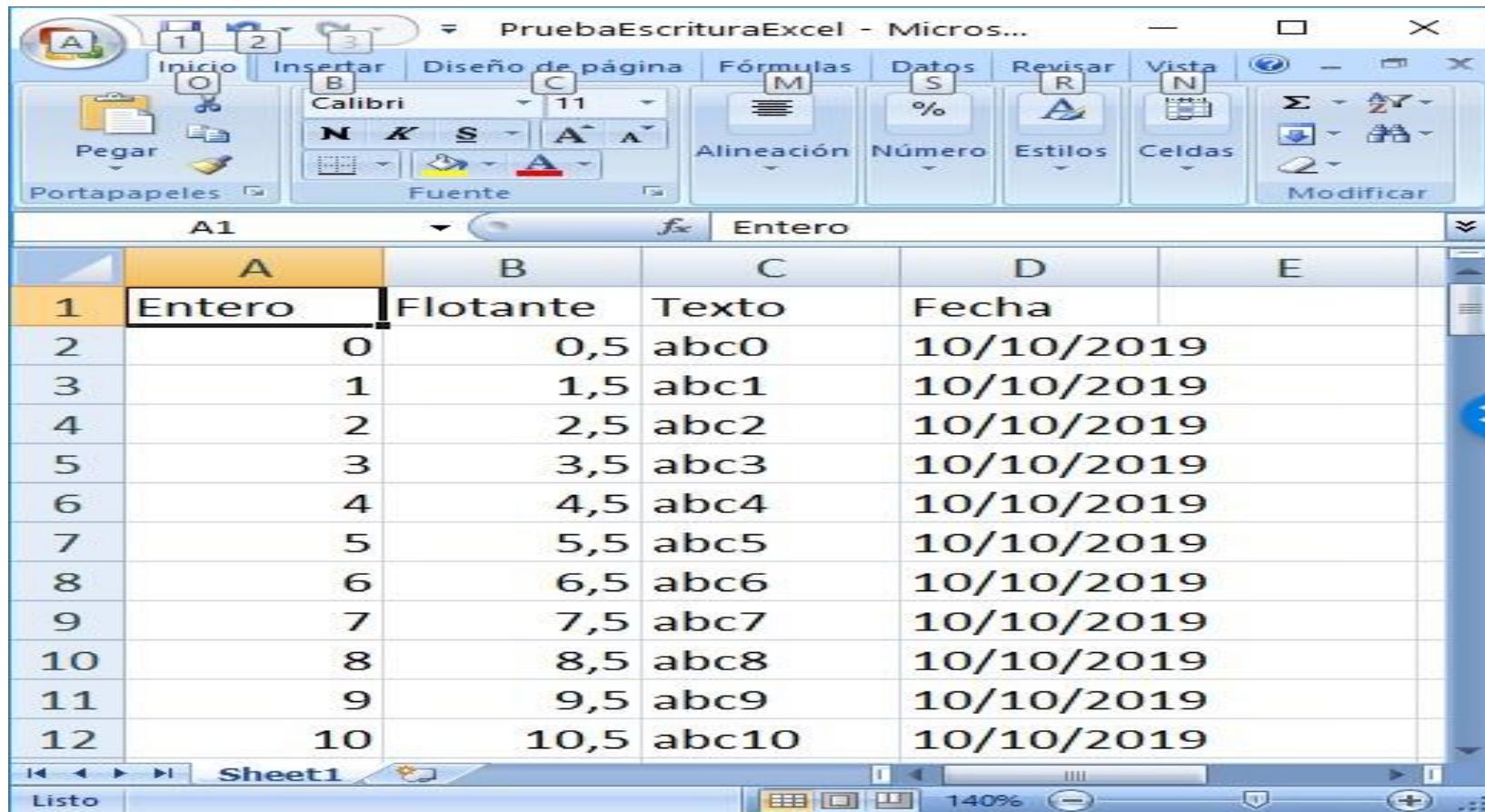


The screenshot shows a code editor window with the following details:

- Title Bar:** d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Escribe... (partially visible)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Tab Bar:** oop.py (active tab) and EscribeUnArchivoXLSX.Py
- Code Area:**

```
25 # escribimos el contenido de la lista de listas
26
27 row = 0
28 col = 0
29
30 for r in s:
31     for c in r:
32         hoja.write(row,col,c)
33         col = col+1
34
35     row=row+1
36     col=0
37
38 libro.close()
39
```
- Status Bar:** [Finished in 0.3s]
- Bottom Status Bar:** Line 4, Column 1, Tab Size: 4, Python

4. Verifico el resultado:



	A	B	C	D	E
1	Entero	Flotante	Texto	Fecha	
2	0	0,5	abc0	10/10/2019	
3	1	1,5	abc1	10/10/2019	
4	2	2,5	abc2	10/10/2019	
5	3	3,5	abc3	10/10/2019	
6	4	4,5	abc4	10/10/2019	
7	5	5,5	abc5	10/10/2019	
8	6	6,5	abc6	10/10/2019	
9	7	7,5	abc7	10/10/2019	
10	8	8,5	abc8	10/10/2019	
11	9	9,5	abc9	10/10/2019	
12	10	10,5	abc10	10/10/2019	

Sugerencias para el código Python:

*Relacionadas con la nomenclatura y
estructura*

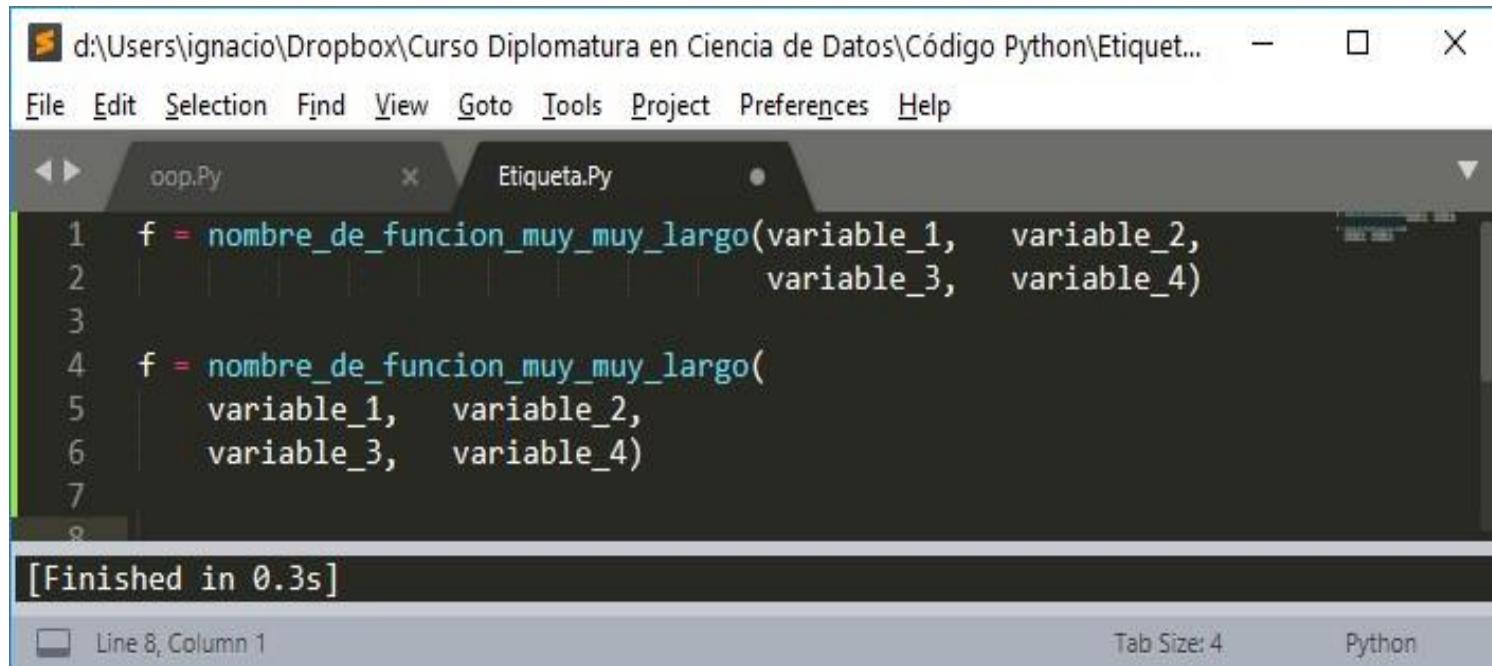
*Relacionadas con la programación
Sugerencias para el código Python:*

*Relacionadas con la nomenclatura y
estructura*

*Relacionadas con la programación
Sugerencias para nomenclatura:*

0. *La regla primordial es la consistencia dentro del equipo de desarrollo.*

1. *Indentar con cuatro espacios (no tabs)*
Sugerencias para nomenclatura:
2. *Continuación de líneas.*



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Etiqueta...

File Edit Selection Find View Goto Tools Project Preferences Help

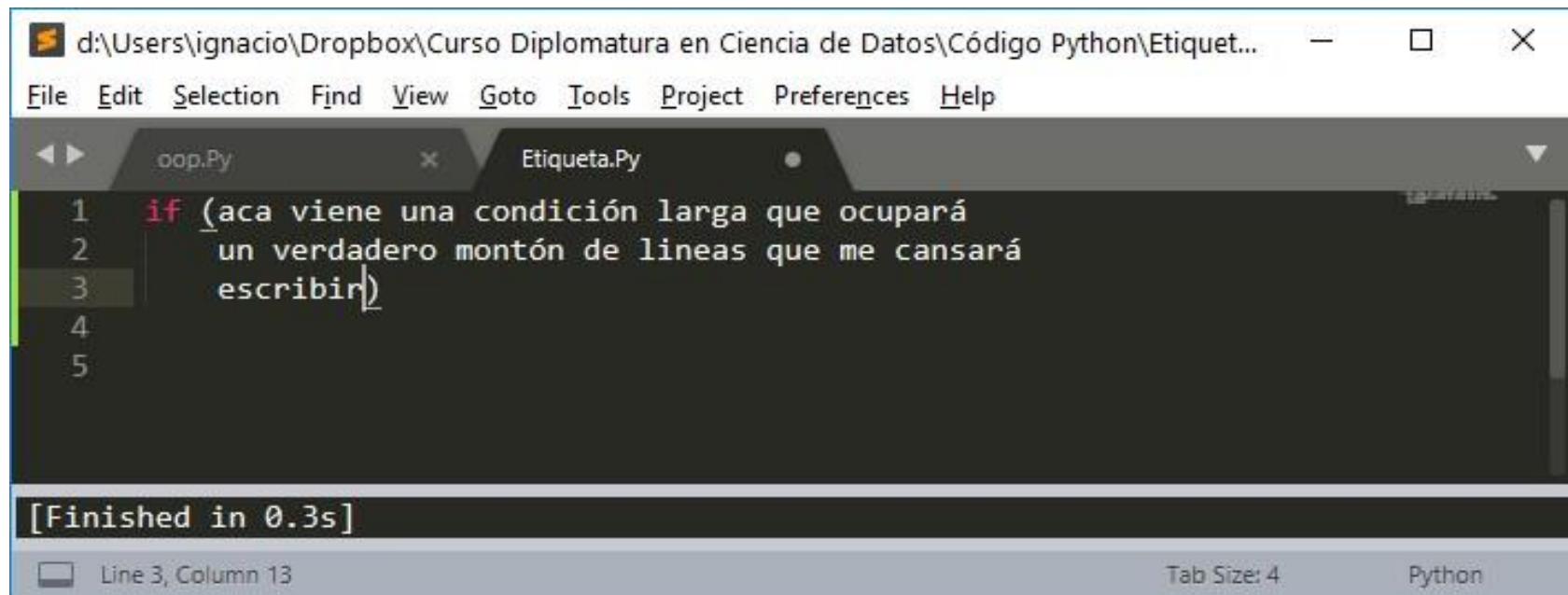
oop.py Etiqueta.py

```
1 f = nombre_de_funcion_muy_muy_largo(variable_1, variable_2,
2                                         variable_3, variable_4)
3
4 f = nombre_de_funcion_muy_muy_largo(
5     variable_1, variable_2,
6     variable_3, variable_4)
```

[Finished in 0.3s]

Line 8, Column 1 Tab Size: 4 Python

2. Continuación de líneas

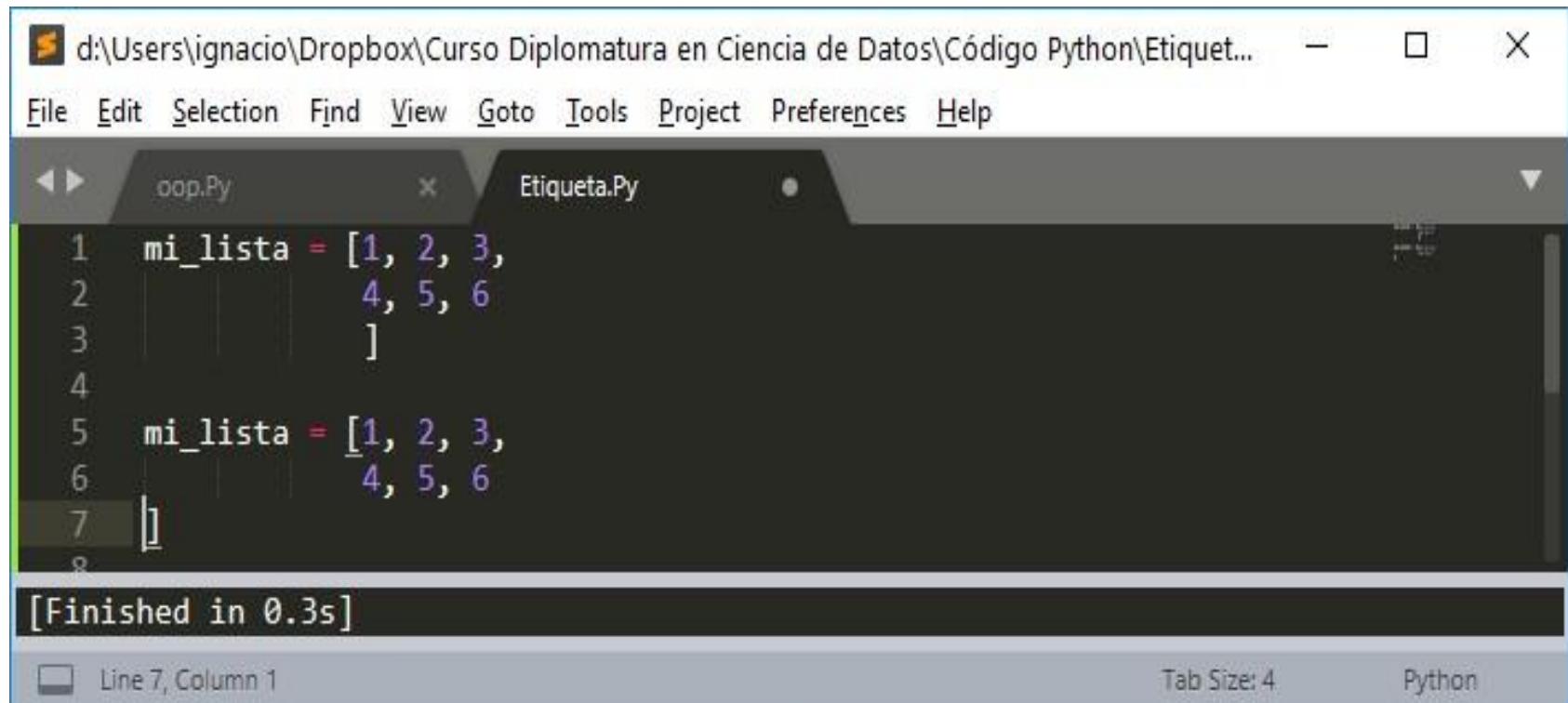


A screenshot of a Python code editor window. The title bar shows the path: d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Etiquet... . The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. There are two tabs open: oop.Py and Etiqueta.Py. The oop.Py tab contains the following code:

```
1 if (aca viene una condición larga que ocupará
2     un verdadero montón de lineas que me cansará
3     escribir)
4
5
```

The Etiqueta.Py tab is currently active. In the status bar at the bottom, it says [Finished in 0.3s], Line 3, Column 13, Tab Size: 4, and Python.

2. Continuación de líneas



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Etiquet...

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py Etiqueta.py

```
1 mi_lista = [1, 2, 3,
2                 4, 5, 6
3             ]
4
5 mi_lista = [1, 2, 3,
6                 4, 5, 6
7 ]
8
```

[Finished in 0.3s]

Line 7, Column 1 Tab Size: 4 Python

2. Continuación de líneas



The screenshot shows a code editor interface with two tabs: 'oop.py' and 'Etiqueta.py'. The 'oop.py' tab is active, displaying the following Python code:

```
1 mi_resultado = (a +  
2             b +  
3             c +  
4             d)  
5  
6 mi_resultado = (a  
7             +b  
8             +c  
9             -d)]
```

The 'Etiqueta.py' tab is visible in the background. The status bar at the bottom of the editor shows '[Finished in 0.1s]'.

3. Máximo largo de líneas



```
File Edit Selection Find View Goto Tools Project Preferences Help
```

```
oop.py x Etiqueta.py x
```

```
1 mi_lista = [1, 2, 3,
2                 4, 5, 6
3
4
5 mi_string = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\
6
7
8
```

[Finished in 0.1s]

Line 7, Column 1 Tab Size: 4 Python

72

4. Líneas en blanco y character set

2 líneas: Definiciones de alto nivel y clases

1 línea: Métodos

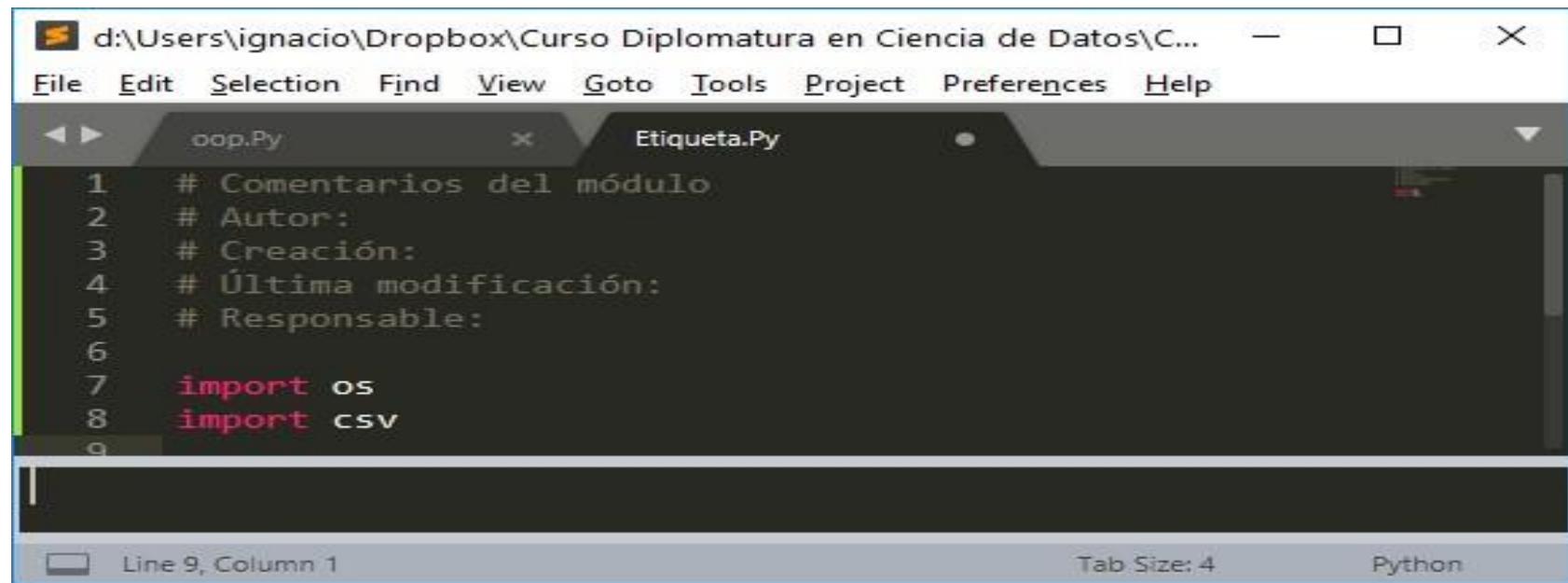
1 línea: para marcar divisiones en el código

Character set: UTF-8

5. Importación de paquetes

Instituto Data Science Argentina

A la cabeza del módulo, sólo después de los comentarios al módulo.



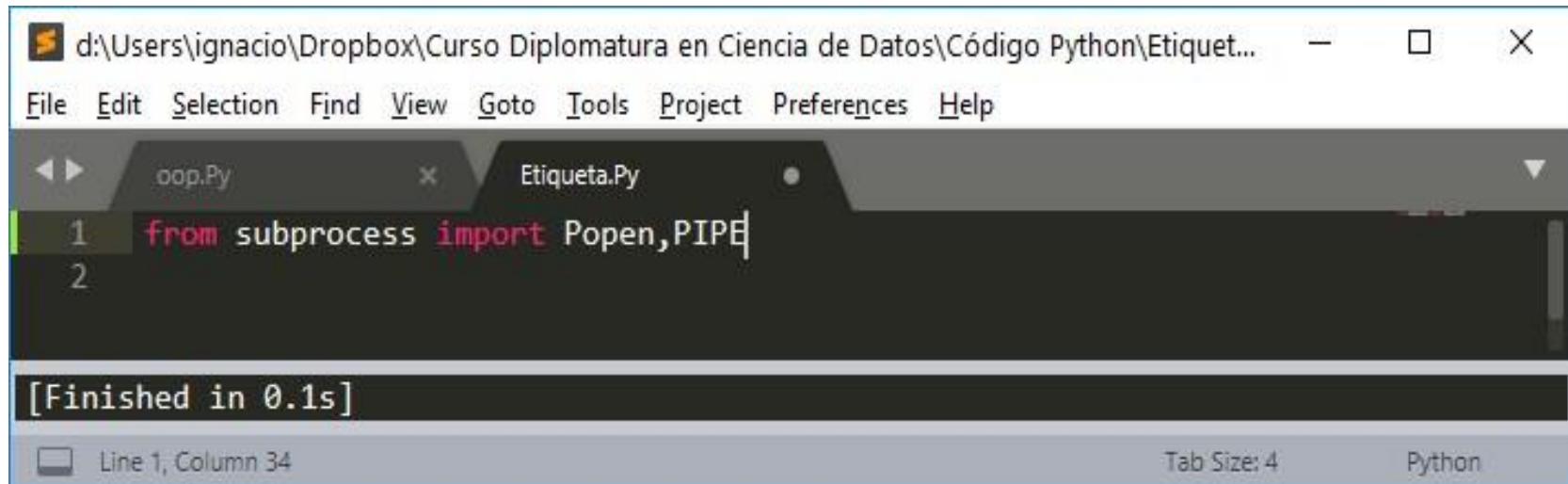
The screenshot shows a Python code editor window with two tabs: 'oop.py' and 'Etiqueta.py'. The 'oop.py' tab is active, displaying the following code:

```
1 # Comentarios del módulo
2 # Autor:
3 # Creación:
4 # Última modificación:
5 # Responsable:
6
7 import os
8 import csv
9
```

The status bar at the bottom indicates 'Line 9, Column 1', 'Tab Size: 4', and 'Python'.

5. Importación de paquetes

Si se usa from se considera aceptable:



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Etiqueta.py

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py Etiqueta.py

```
1 from subprocess import Popen,PIPE
2
```

[Finished in 0.1s]

Line 1, Column 34 Tab Size: 4 Python

5. Importación de paquetes

Se deben separar en tres grupos:

- *Librerías estándar:*
(viene con la instalación de Python)
- *Librerías de terceras partes*
(Hay que bajarlas aparte: XLSXwriter)
- *Librerías locales*
(Las desarrolladas por nuestro equipo)

6. Comillas

Las comillas dobles y simples son equivalentes.

Se recomienda usarlas de un solo tipo.

7. Espacios en blanco

Nunca entre la función y el paréntesis que se abre.

Nunca entre un parámetro y la coma siguiente.

Nunca entre tupla, lista o diccionario y los paréntesis, corchetes o llaves.

8. *Varias instrucciones en la misma línea*

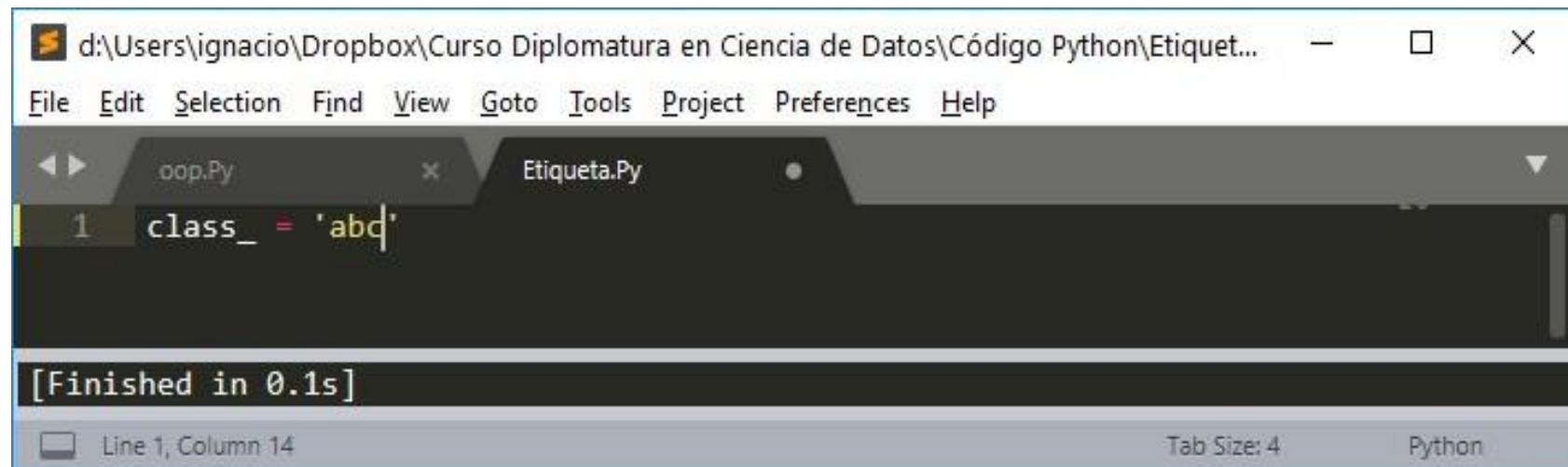
No!

9. Convenciones de nombres

- b (sólo minúsculas en la primer letra)
- B (siempre comenzando con mayúscula)
- Sólo minúsculas
- Sólo minúsculas con _
- Sólo mayúsculas
- Sólo mayúsculas con _
- Al iniciar una palabra utilizar una mayúscula.
- Lo mismo que la anterior pero comenzando en minúscula

10. Conflicto con palabras reservadas

Se salvan adicionando un _ al final de la palabra:



The screenshot shows a Python code editor window. The title bar indicates the path: d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Etiquet... The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. There are two tabs open: 'oop.Py' and 'Etiqueta.Py'. The 'oop.Py' tab contains the following code:

```
1 class_ = 'abc'
```

The word 'class' is highlighted in yellow, indicating it is a reserved keyword. A tooltip or status message at the bottom left says '[Finished in 0.1s]'. At the bottom, status bars show 'Line 1, Column 14', 'Tab Size: 4', and 'Python'.

11. Doble guión bajo:

Sólo para las palabras reservadas:

`__init__`

`__import__`

`__file__`

12. *Evitar problemas de lectura:*

Evitar:

- I: I minúscula que se puede confundir con 1 O:
O mayúscula que puede confundirse con 0
- I: I mayúscula que puede confunirse con 1

Si hay que usar I (ele) que sea L

13. Evitar caracteres especiales:

Si bien están dentro de ASCII

```
1 # Cosas que andan pero NO son una buena idea:
2 l = 1
3 o = 2
4 i = 3
5 print(l,o,i)
6 n = 1
7 a = 2
8 e = 3
9 i = 4
10 o = 5
11 u = 6
12 print(n,a,e,i,o,u)
```

14. Nombres de módulos y paquetes:

Cortos

Minúsculas

Se pueden usar _ en los módulos pero se desalienta en los paquetes

15. Nombres de funciones y variables

Separar las palabras con _

Minúsculas

Si colisiona con una palabra reservada agregar un _ al final

16. Nombres de funciones y variables

Separar las palabras con _

Minúsculas

Si colisiona con una palabra reservada agregar un _ al final

17. Nombres de métodos y propiedades

Separar las palabras con _

Minúsculas

Si colisiona con una palabra reservada agregar un _ al final

18. Diseñando para la herencia

No usar __ en los métodos y propiedades públicos
Minúsculas

Si colisiona con una palabra reservada agregar un _ al final

Para atributos simples propiedades

Ante la duda hacer los métodos y propiedades no públicos

MySQL:

- *pymysql*
 - *connect*
 - *query*
 - *commit*
 - *cursor*
 - *fetchone*

- *fetchmany*
- *fetchall*
- *close*

Agregamos datos

```
1 # Ejemplos de pymysql
2
3 import pymysql
4
5 conexion = pymysql.connect(user='root', password='Polemo$2017',
6                             host='127.0.0.1', port=3306,
7                             database='python')
8
9 k = 5
10
11 while True:
12     curso = input("Título del curso: ")
13     if curso == "":
14         break
15
16     q = "insert into cursos(IdCurso,Curso) values("+str(k)+", '"+curso+"')"
17     conexion.query(q)
18     conexion.commit()
19     k = k + 1
20
21 conexion.close()
```

Recuperar datos: `fetchall()`

```
1 # Ejemplos de pymysql
2
3 import pymysql
4
5 conexion = pymysql.connect(user='root', password='Polemo$2017',
6                             host='127.0.0.1', port=3306,
7                             database='python')
8
9 cur = conexion.cursor()
10
11 cur.execute("select * from cursos")
12
13 rows = cur.fetchall()
14
15 for row in rows:
16     print(row)
17
18 cur.close()
19 conexion.close()
20
(1, 'Diplomatura en Bases de Datos')
(2, 'Diplomatura en Ciencia de Datos')
(3, 'Diplomatura en Análisis de negocios')
(4, 'Diplomatura en BI')
(5, 'Científico de datos con R')
(6, 'Científico de datos con Python')
(7, 'Tópicos de Series Temporales')
(8, 'Especialista en Big Data con Apache Hadoop')
[Finished in 0.2s]
```

Recuperar datos: `fetchone()`

```
1 # Ejemplos de pymysql
2
3 import pymysql
4
5 conexion = pymysql.connect(user='root', password='Polemo$2017',
6                             host='127.0.0.1', port=3306,
7                             database='python')
8
9 mi_cursor = conexion.cursor()
10 mi_cursor.execute("select * from cursos")
11
12
13 while True:
14     row = mi_cursor.fetchone()
15     if row is None:
16         break
17     print(row)
18
19 mi_cursor.close()
20 conexion.close()
21
(1, 'Diplomatura en Bases de Datos')
(2, 'Diplomatura en Ciencia de Datos')
(3, 'Diplomatura en Análisis de negocios')
(4, 'Diplomatura en BI')
[Finished in 0.3s]
```

Recuperar datos: *fetchmany(1)*

```
1 # Ejemplos de pymysql
2
3 import pymysql
4
5 conexion = pymysql.connect(user='root', password='Polemo$2017',
6                             host='127.0.0.1', port=3306,
7                             database='python')
8
9 mi_cursor = conexion.cursor()
10 mi_cursor.execute("select * from cursos")
11
12
13 while True:
14     row = mi_cursor.fetchmany(1)
15     if row == []:
16         break
17     print(row)
18
19 mi_cursor.close()
20 conexion.close()
21
((1, 'Diplomatura en Bases de Datos'),)
((2, 'Diplomatura en Ciencia de Datos'),)
((3, 'Diplomatura en Análisis de negocios'),)
((4, 'Diplomatura en BI'),)
[Finished in 0.3s]
```

Recuperar datos: `fetchmany(n)`

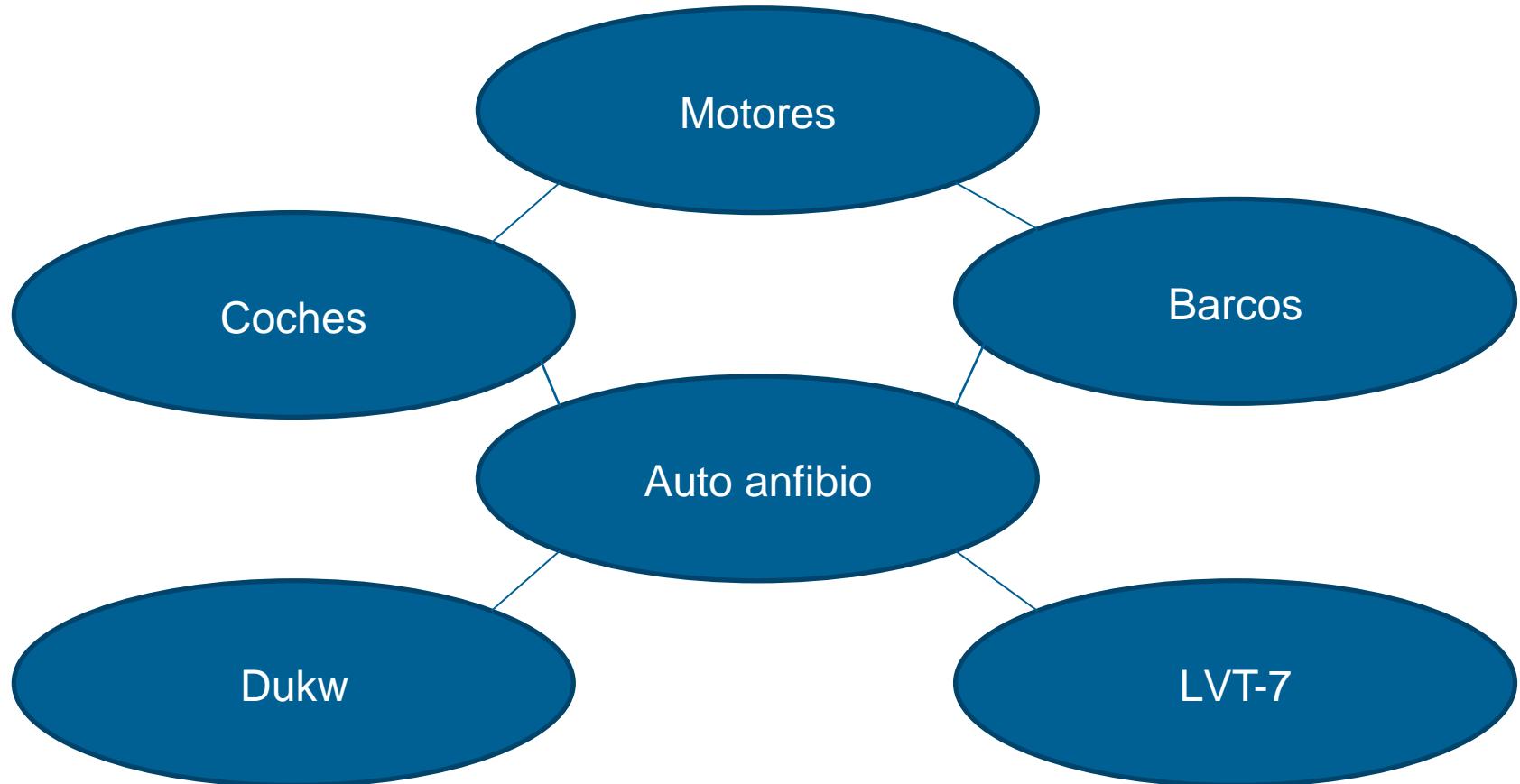
```
3  import pymysql
4
5  conexion = pymysql.connect(user='root', password='Polemo$2017',
6                             host='127.0.0.1', port=3306,
7                             database='python')
8
9  mi_cursor = conexion.cursor()
10 mi_cursor.execute("select * from cursos")
11
12
13 while True:
14     rows = mi_cursor.fetchmany(2)
15     if rows == () or rows is None:
16         break
17     print("Avanzo página")
18     for row in rows:
19         print(row)
20
21 mi_cursor.close()
22 conexion.close()
```

```
Avanzo página
(1, 'Diplomatura en Bases de Datos')
(2, 'Diplomatura en Ciencia de Datos')
Avanzo página
(3, 'Diplomatura en Análisis de negocios')
(4, 'Diplomatura en BI')
[Finished in 0.3s]
```

Adaptación de impedancias

- El objeto contiene toda la información de su estado.
- Cuando lo plasmamos en la base de datos esa información se reparte entre numerosas tablas.
- Al recuperarlo hay que juntar los pedacitos.

Visión de las clases



Atributos (super simple)

- Coches
 - Ruedas
 - Dukw
 - Carga
- Motores
 - Marca
 - Modelo
 - LVT-7
 - HMG
 - Barcos



– Calado

Instituto Data Science Argentina

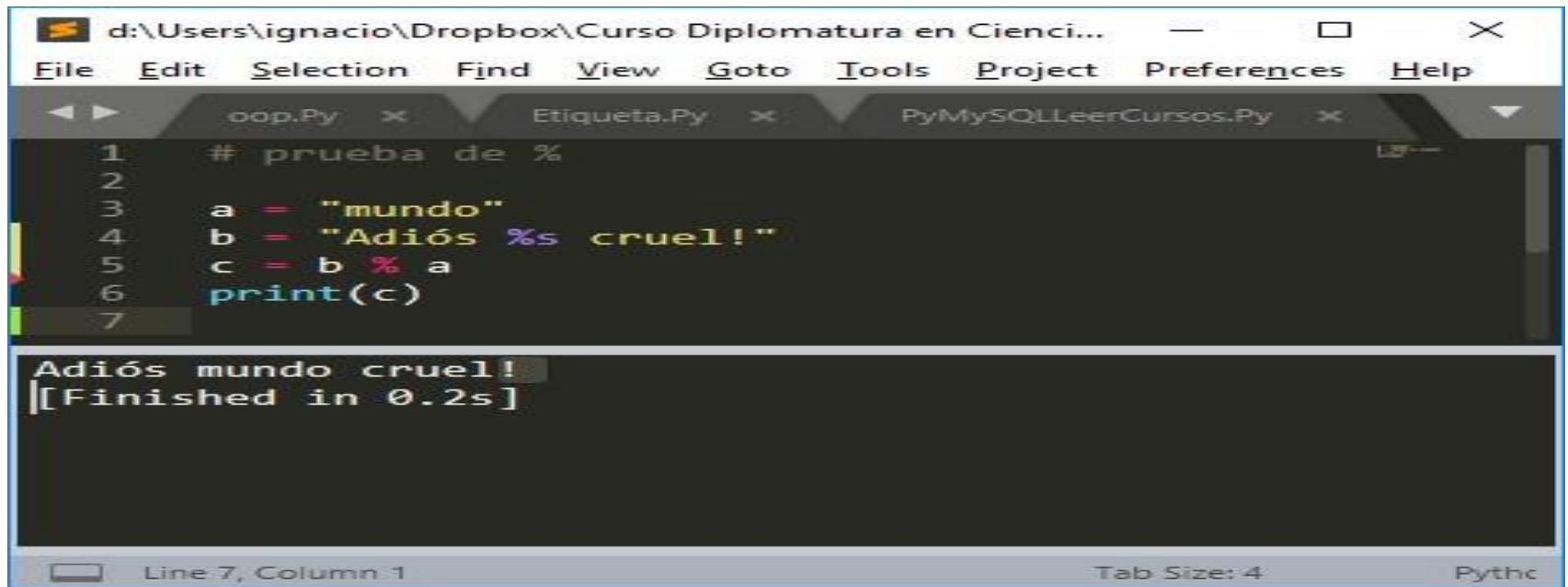
Strings y gráficos

- *Formateo de Strings*
 - %
 - *str.format*
 - *f-strings*
- *Funciones lambda*
- *Matplotlib*

Formateo básico:

Instituto Data Science Argentina

% sirve para insertar los datos de una variable string dentro de una cadena de caracteres.



A screenshot of a Python code editor showing a script named 'oop.py'. The code demonstrates string interpolation using the '%' operator:

```
# prueba de %
a = "mundo"
b = "Adiós %s cruel!"
c = b % a
print(c)
```

The output window shows the result of running the script:

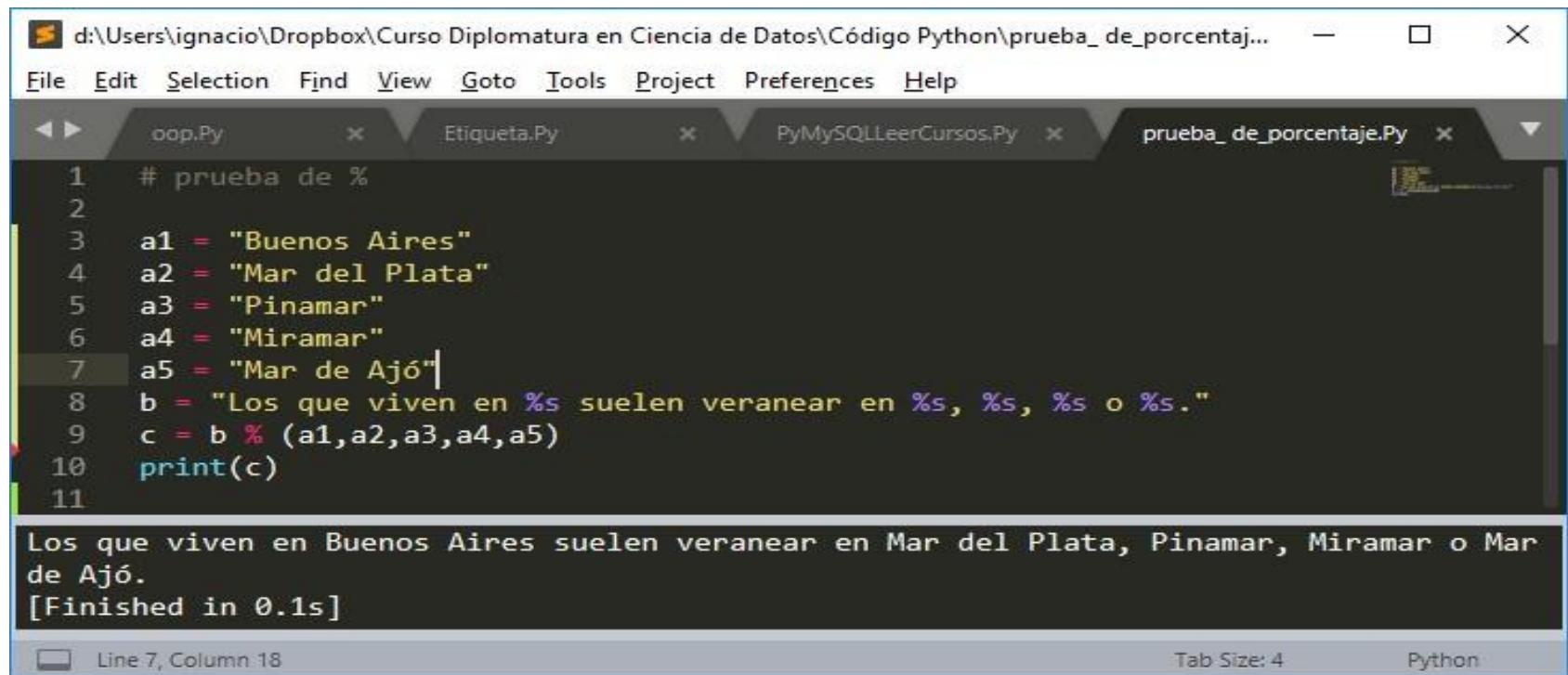
```
Adiós mundo cruel!
[Finished in 0.2s]
```

At the bottom of the editor, status bars indicate 'Line 7, Column 1' and 'Tab Size: 4'.

Formateo básico:

Instituto Data Science Argentina

Se pueden usar varias en secuencia:



A screenshot of a Python code editor showing a script named `prueba_de_porcentaje.py`. The code defines five variables `a1` through `a5` representing locations, and then uses `str.format()` to print a sentence including all of them. The output window shows the resulting string.

```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba_de_porcentaje.py
```

```
File Edit Selection Find View Goto Tools Project Preferences Help
```

```
oop.Py x Etiqueta.Py x PyMySQLLeerCursos.Py x prueba_de_porcentaje.Py x
```

```
1 # prueba de %
2
3 a1 = "Buenos Aires"
4 a2 = "Mar del Plata"
5 a3 = "Pinamar"
6 a4 = "Miramar"
7 a5 = "Mar de Ajó"
8 b = "Los que viven en %s suelen veranear en %s, %s, %s o %s."
9 c = b % (a1,a2,a3,a4,a5)
10 print(c)
11
```

```
Los que viven en Buenos Aires suelen veranear en Mar del Plata, Pinamar, Miramar o Mar de Ajó.
[Finished in 0.1s]
```

```
Line 7, Column 18 Tab Size: 4 Python
```

str.format()

Instituto Data Science Argentina



A screenshot of a Python code editor window. The title bar shows the path: "d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba_de_porcentaje.py". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The tabs at the top show "oop.Py", "Etiqueta.Py", "PyMySQLLeerCursos.Py", and "prueba_de_porcentaje.Py". The code editor displays the following Python script:

```
1 # prueba de str.format
2
3 a1 = "Buenos Aires"
4 a2 = "Mar del Plata"
5 a3 = "Pinamar"
6 a4 = "Miramar"
7 a5 = "Mar de Ajó"
8 b = "Los que viven en {} suelen veranear en {}, {}, {} o {}."
9 c = b.format(a1,a2,a3,a4,a5)
10 print(c)
11
```

The output window below shows the result of running the script:

```
Los que viven en Buenos Aires suelen veranear en Mar del Plata, Pinamar, Miramar o Mar de Ajó.
[Finished in 0.2s]
```

At the bottom, status bars indicate "Line 9, Column 1", "Tab Size: 4", and "Python".

Uso de diccionarios con str.format()



A screenshot of a Python code editor window. The title bar shows the path: "d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba_de_porcentaje.py". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The tab bar has four tabs: "oop.py", "Etiqueta.py", "PyMySQLLeerCursos.py", and "prueba_de_porcentaje.py". The code area contains the following Python code:

```
1 # prueba de str.format
2
3 d = {"c0": "Buenos Aires", "c1": "Mar del Plata"}
4 b = "Los que viven en {c0} suelen veranean en {c1} y los que viven en {c1} veranean en {c0}"
5 c = b.format(**d)
6 print(c)
```

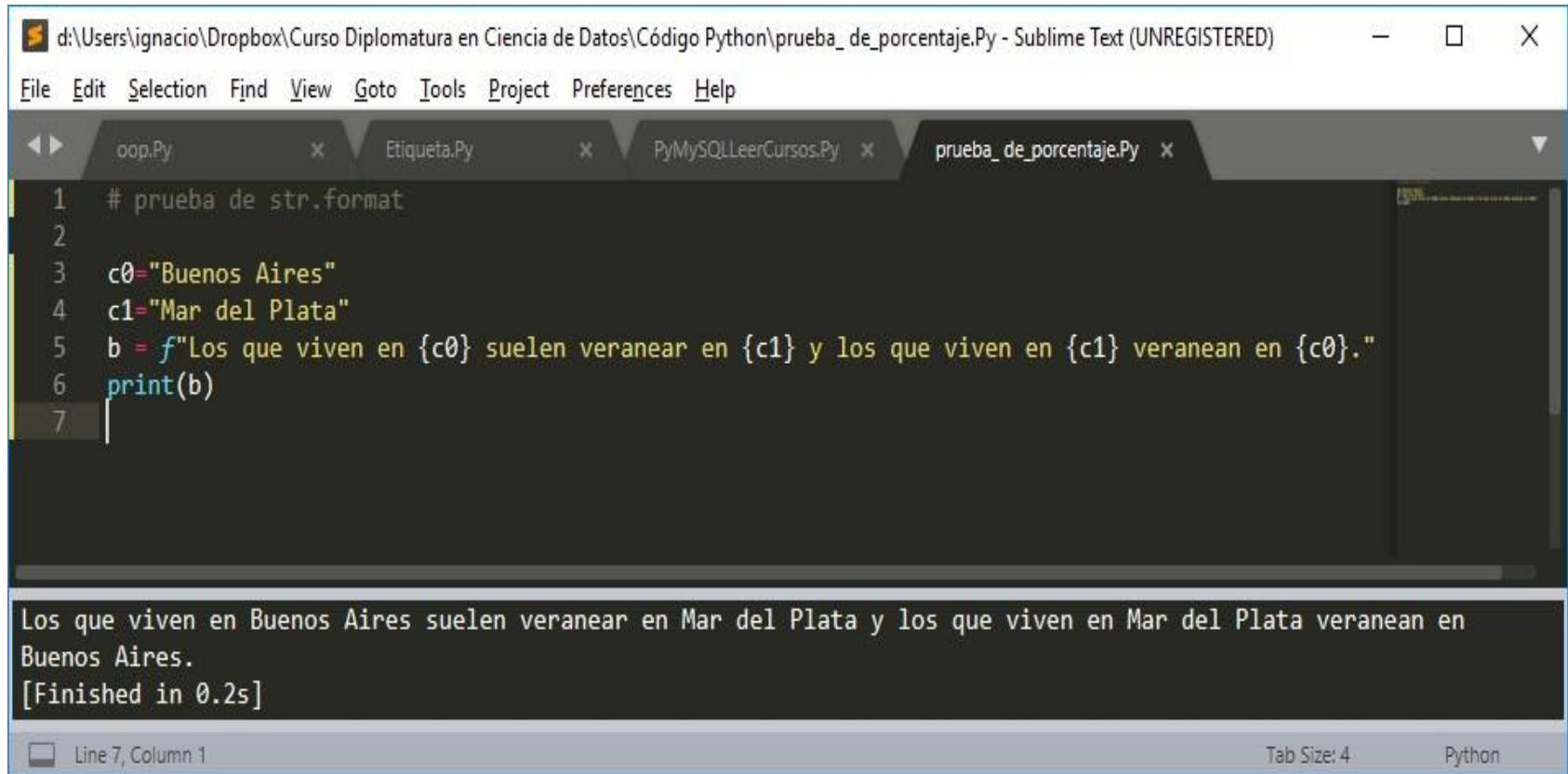
The output window below shows the result of running the code:

```
Los que viven en Buenos Aires suelen veranean en Mar del Plata y los que viven en Mar del Plata veranean en Buenos Aires.
[Finished in 0.1s]
```

At the bottom, status bars indicate "Line 6, Column 9", "Tab Size: 4", and "Python".

F-strings:

Instituto Data Science Argentina



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba_de_porcentaje.Py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py Etiqueta.Py PyMySQLLeerCursos.Py prueba_de_porcentaje.Py

```
1 # prueba de str.format
2
3 c0="Buenos Aires"
4 c1="Mar del Plata"
5 b = f"Los que viven en {c0} suelen veranear en {c1} y los que viven en {c1} veranean en {c0}."
6 print(b)
7
```

Los que viven en Buenos Aires suelen veranear en Mar del Plata y los que viven en Mar del Plata veranean en Buenos Aires.
[Finished in 0.2s]

Line 7, Column 1 Tab Size: 4 Python

F-strings llamando a funciones:



INSTITUTO

Data Science

d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba_de_porcentaje.Py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py Etiqueta.Py PyMySQLLeerCursos.Py prueba_de_porcentaje.Py

```
1 # prueba de f-strings
2
3 def suma(a,b):
4     return a+b
5
6 c0="Buenos Aires"
7 c1="Mar del Plata"
8 b = f"Los que viven en {c0} suelen veranear en {c1} cada 2 x {suma(3,1)}"
9 print(b)
10
```

Los que viven en Buenos Aires suelen veranear en Mar del Plata cada 2 x 4
[Finished in 0.1s]

Line 5, Column 1 Tab Size: 4 Python

Uso de diccionarios con f-strings



A screenshot of the Sublime Text editor. The title bar shows the path "d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba_de_porcentaje.Py - Sublime Text (UNREGISTERED)" and the window controls. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. Below the menu is a tab bar with four tabs: "oop.Py", "Etiqueta.Py", "PyMySQLLeerCursos.Py", and "prueba_de_porcentaje.Py". The main editor area contains the following Python code:

```
1 # prueba de f-strings
2
3 d = {'origen': 'Buenos Aires', 'destino': 'Mar del Plata'}
4
5 b = f'Los que viven en {d["origen"]} suelen veranear en {d["destino"]} cada 2 x 4'
6
7 print(b)
8
```

The output panel at the bottom shows the result of running the code:

```
Los que viven en Buenos Aires suelen veranear en Mar del Plata cada 2 x 4
[Finished in 0.1s]
```

At the bottom of the editor, status bars indicate "Line 6, Column 5", "Tab Size: 4", and "Python".

Funciones lambda



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\lambda.Py - Sublime Te...

File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py Etiqueta.Py PyMySQLLeerCursos.Py lambda.Py

```
1 # prueba de funciones lambda
2
3 x = Lambda a: a+10
4
5 print(x(5))
6
```

15
[Finished in 0.1s]

Line 4, Column 1: Build finished Tab Size: 4 Python

Funciones lambda – varios parámetros



The screenshot shows a Sublime Text editor window. The title bar indicates the file path: d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\lambda.py - Sublime Text. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The tab bar shows four files: oop.py, Etiqueta.py, PyMySQLLeerCursos.py, and lambda.py. The lambda.py file is the active tab, displaying the following Python code:

```
1 # prueba de funciones lambda
2
3 x = Lambda a,b,c: a+b+c
4
5 print(x(5,4,3))
6
```

In the status bar at the bottom, it says Line 5, Column 14, Tab Size: 4, and Python.

Output window content:

```
12
[Finished in 0.1s]
```

Funciones lambda en funciones:



The screenshot shows a Sublime Text editor window. The title bar indicates the file path: "d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\lambda.py - Sublime Text". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The tabs at the top show "oop.py", "Etiqueta.py", "PyMySQLLeerCursos.py", and "lambda.py". The main editor area contains the following Python code:

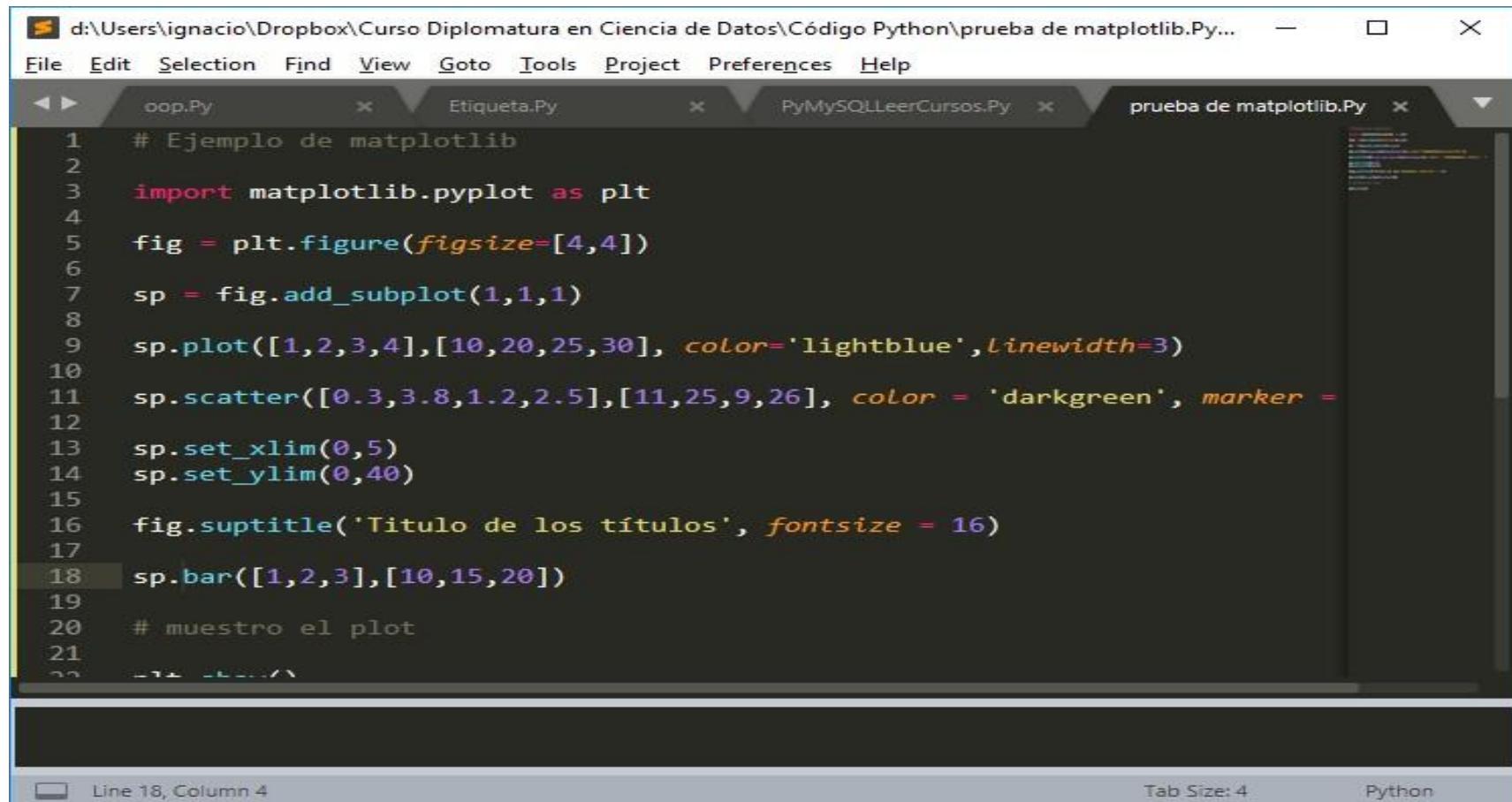
```
1 # prueba de funciones lambda
2
3 def mi_funcion(n):
4     return lambda a : a * n
5
6 doblador = mi_funcion(2)
7 triplicador = mi_funcion(3)
8
9 print(doblador(5), triplicador(6))
10
```

The output panel at the bottom displays the results of running the code: "10 18 [Finished in 0.1s]". The status bar at the bottom shows "Line 5, Column 1", "Tab Size: 4", and "Python".

Matplotlib

- Figure • Ax.fill • Ax.boxplot
- Subplot • Ax.fill_between • Ax.violinplot
- Suptitle • Ax.stackplot • Ax.pcolor
- Ax.bar • Ax.arrow • Ax.pcolormesh
- Ax.bars • Ax.quiver • Ax.contour
- Axhline • Ax.streamplot • Ax.contourf
- Axvline • Ax.hist • Axlabel

Figure, Subplot, Suptitle, Scatter, Bar



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba de matplotlib.Py...

```
File Edit Selection Find View Goto Tools Project Preferences Help
oop.Py x Etiqueta.Py x PyMySQLLeerCursos.Py x prueba de matplotlib.Py x

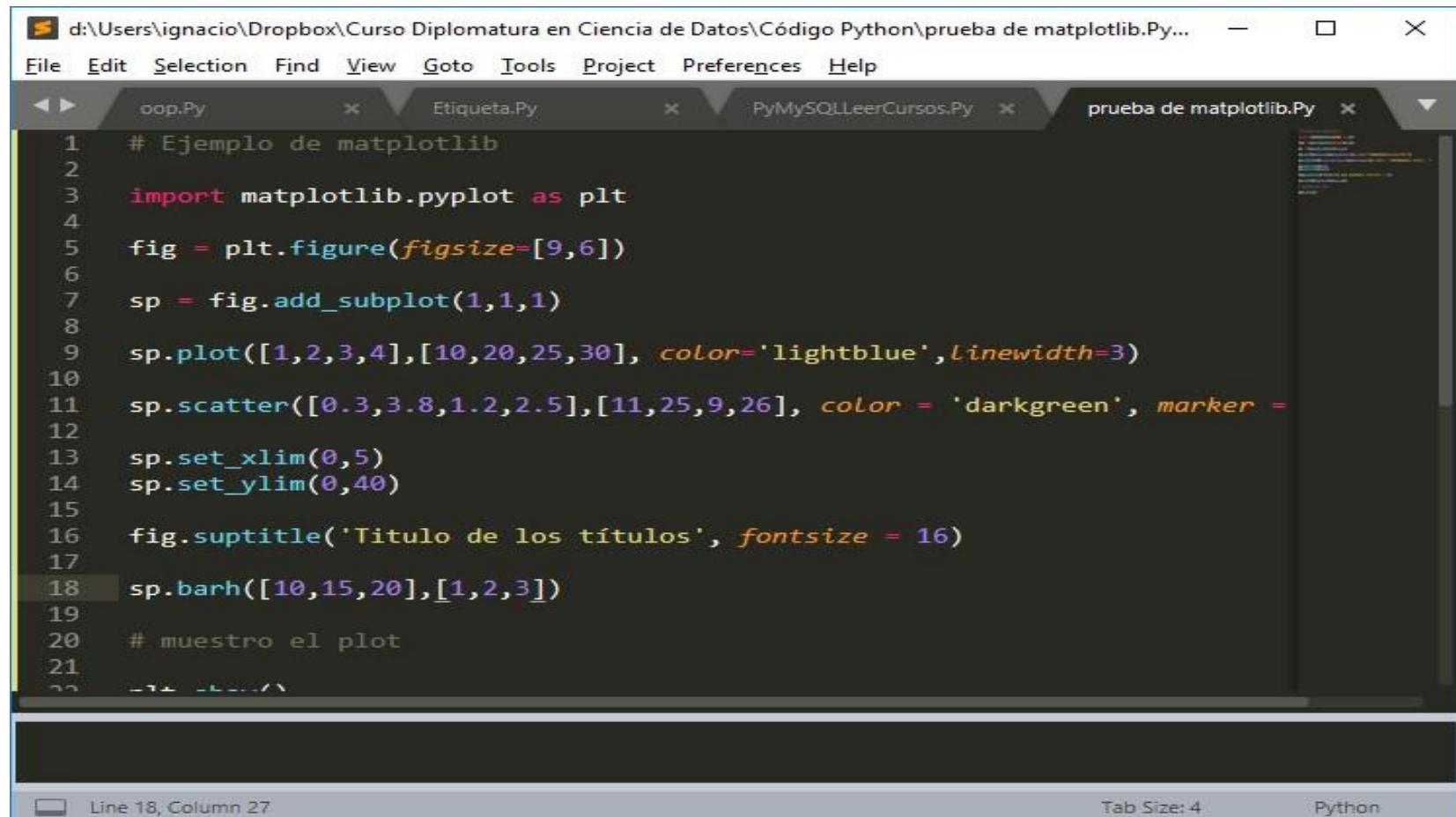
1 # Ejemplo de matplotlib
2
3 import matplotlib.pyplot as plt
4
5 fig = plt.figure(figsize=[4,4])
6
7 sp = fig.add_subplot(1,1,1)
8
9 sp.plot([1,2,3,4],[10,20,25,30], color='lightblue', linewidth=3)
10
11 sp.scatter([0.3,3.8,1.2,2.5],[11,25,9,26], color = 'darkgreen', marker =
12
13 sp.set_xlim(0,5)
14 sp.set_ylim(0,40)
15
16 fig.suptitle('Titulo de los títulos', fontsize = 16)
17
18 sp.bar([1,2,3],[10,15,20])
19
20 # muestro el plot
21
22 plt.show()
```

Line 18, Column 4 Tab Size: 4 Python

Figure, Subplot, Suptitle, Scatter, Bar



Figure, Subplot, Suptitle, Scatter & Barh

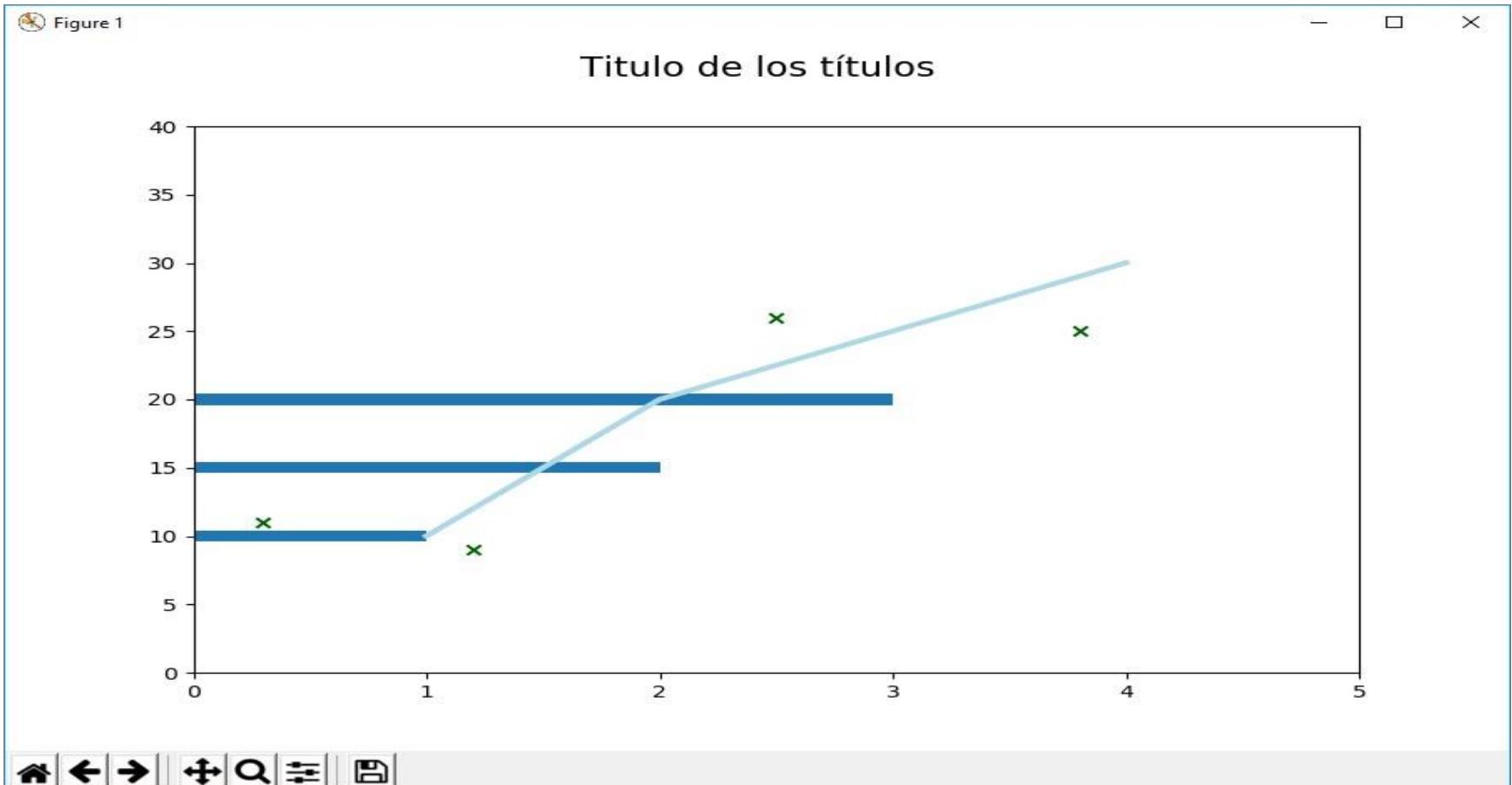


The screenshot shows a code editor window with several tabs at the top: 'oop.Py', 'Etiqueta.Py', 'PyMySQLLeerCursos.Py', and 'prueba de matplotlib.Py'. The 'prueba de matplotlib.Py' tab is active, displaying the following Python code:

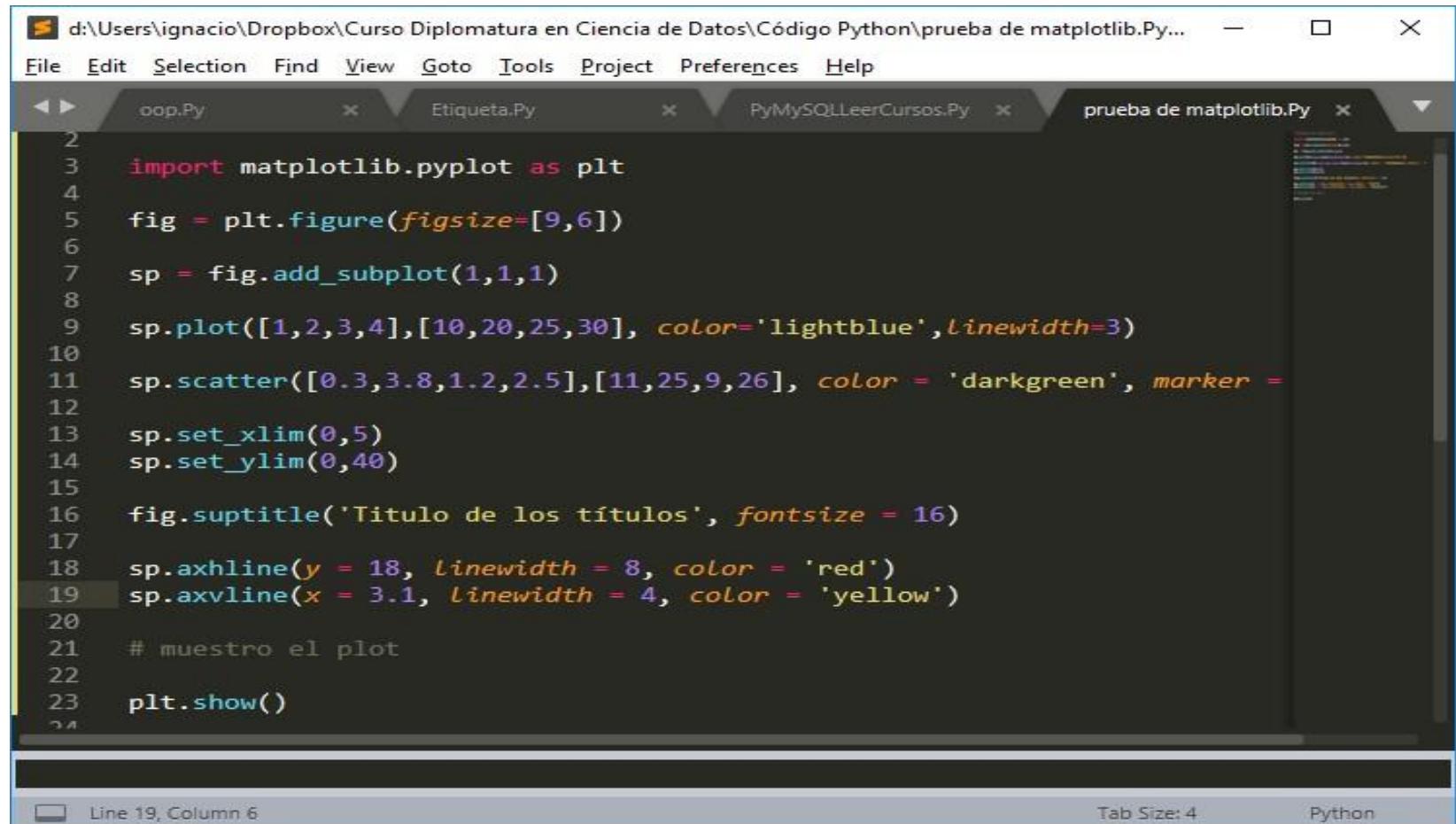
```
1 # Ejemplo de matplotlib
2
3 import matplotlib.pyplot as plt
4
5 fig = plt.figure(figsize=[9,6])
6
7 sp = fig.add_subplot(1,1,1)
8
9 sp.plot([1,2,3,4],[10,20,25,30], color='lightblue', linewidth=3)
10
11 sp.scatter([0.3,3.8,1.2,2.5],[11,25,9,26], color = 'darkgreen', marker =
12
13 sp.set_xlim(0,5)
14 sp.set_ylim(0,40)
15
16 fig.suptitle('Titulo de los títulos', fontsize = 16)
17
18 sp.bart([10,15,20],[1,2,3])
19
20 # muestro el plot
21
22
```

The status bar at the bottom indicates 'Line 18, Column 27', 'Tab Size: 4', and 'Python'.

Figure, Subplot, Suptitle, Scatter & Barh



Figure, Subplot, Suptitle, axhline, axvline

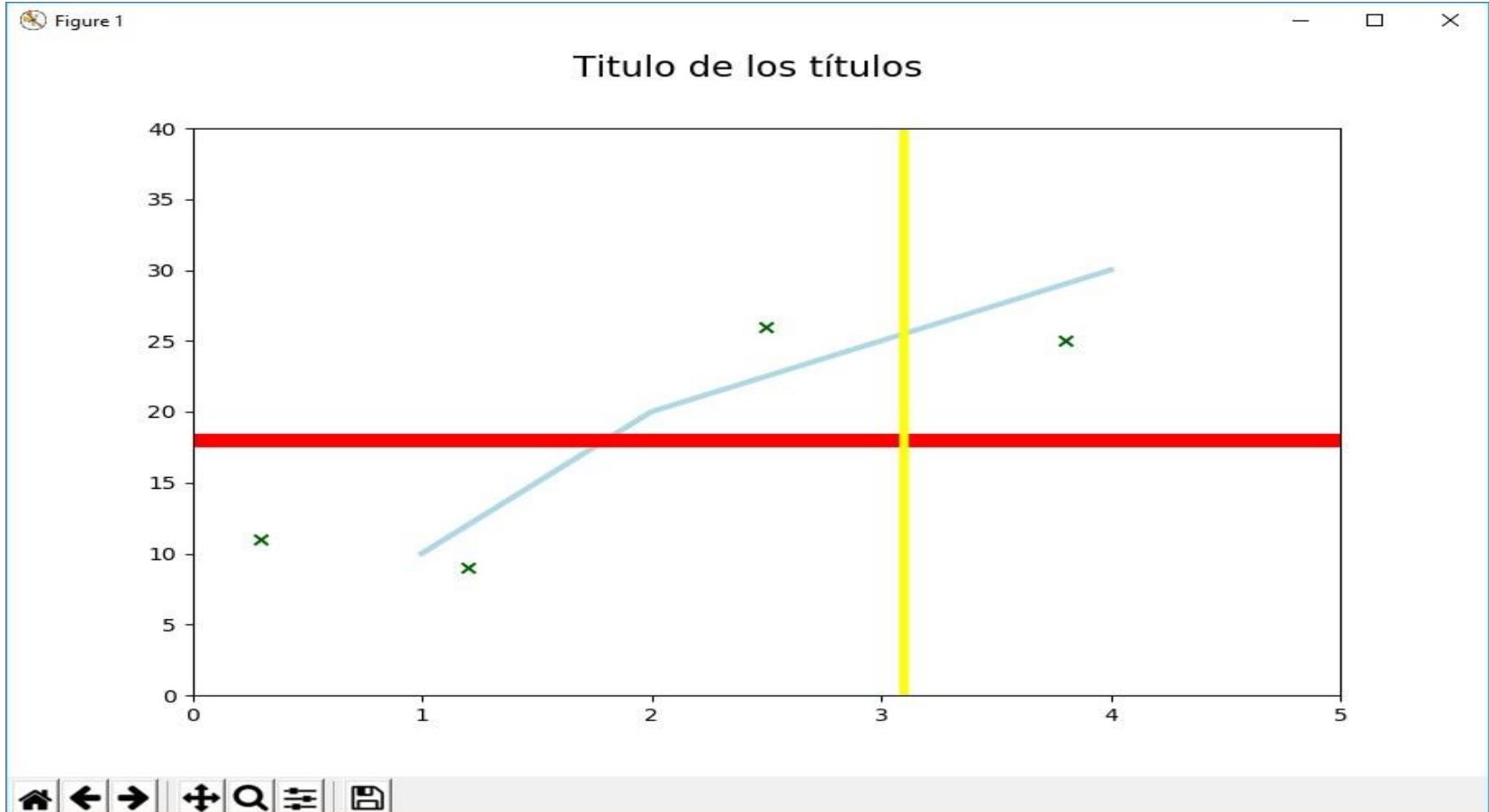


The screenshot shows a Python code editor with the following code:

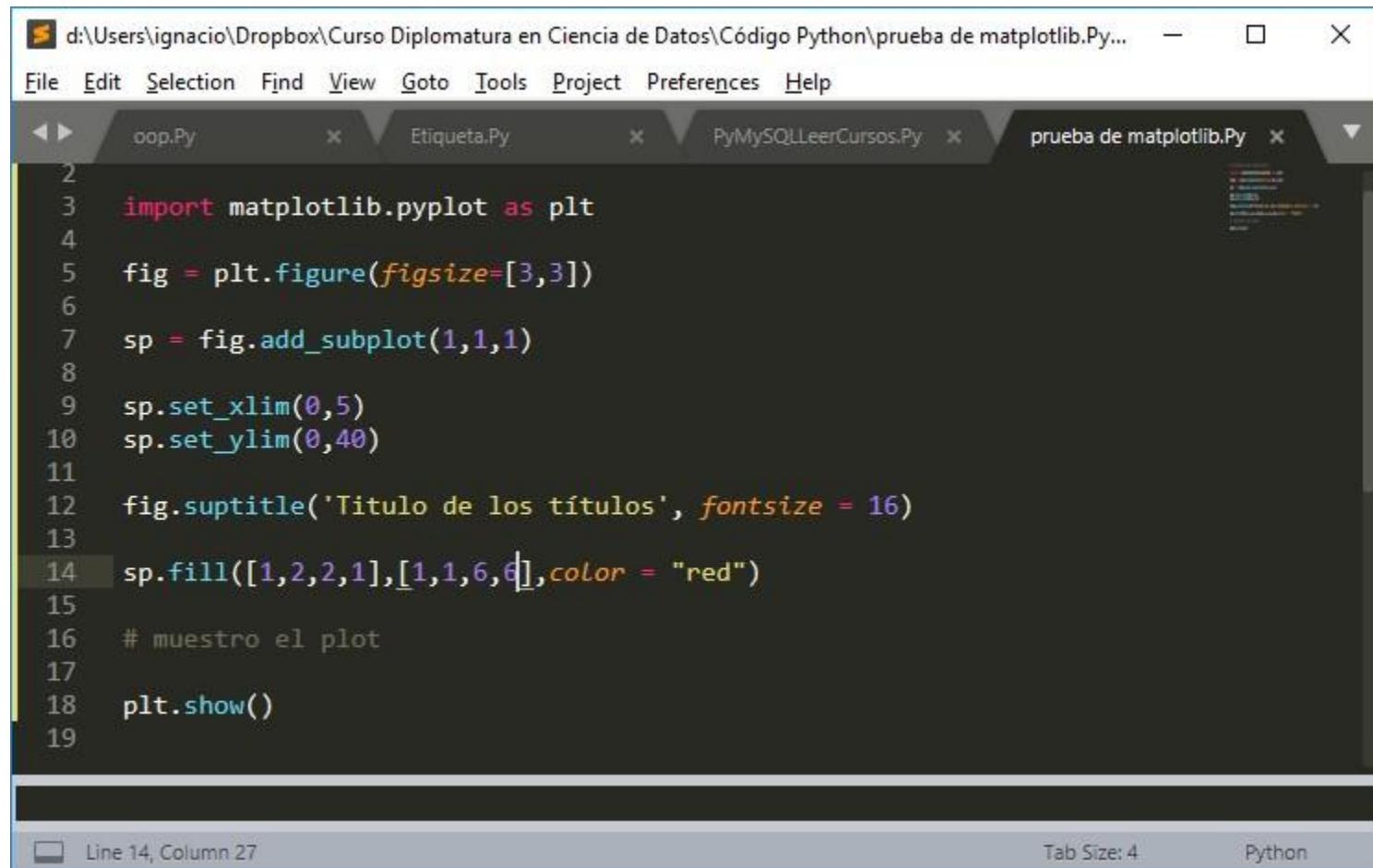
```
2
3     import matplotlib.pyplot as plt
4
5     fig = plt.figure(figsize=[9,6])
6
7     sp = fig.add_subplot(1,1,1)
8
9     sp.plot([1,2,3,4],[10,20,25,30], color='lightblue', linewidth=3)
10
11    sp.scatter([0.3,3.8,1.2,2.5],[11,25,9,26], color = 'darkgreen', marker =
12
13    sp.set_xlim(0,5)
14    sp.set_ylim(0,40)
15
16    fig.suptitle('Titulo de los titulos', fontsize = 16)
17
18    sp.axhline(y = 18, linewidth = 8, color = 'red')
19    sp.axvline(x = 3.1, linewidth = 4, color = 'yellow')
20
21    # muestro el plot
22
23    plt.show()
24
```

Line 19, Column 6 Tab Size: 4 Python

Figure, Subplot, Suptitle, axhline, axvline



Figure, Subplot, Suptitle, ax.fill



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba de matplotlib.Py... — X

File Edit Selection Find View Goto Tools Project Preferences Help

oop.py Etiqueta.py PyMySQLLeerCursos.py prueba de matplotlib.py

```
2
3 import matplotlib.pyplot as plt
4
5 fig = plt.figure(figsize=[3,3])
6
7 sp = fig.add_subplot(1,1,1)
8
9 sp.set_xlim(0,5)
10 sp.set_ylim(0,40)
11
12 fig.suptitle('Titulo de los títulos', fontsize = 16)
13
14 sp.fill([1,2,2,1],[1,1,6,6],color = "red")
15
16 # muestro el plot
17
18 plt.show()
19
```

Line 14, Column 27 Tab Size: 4 Python

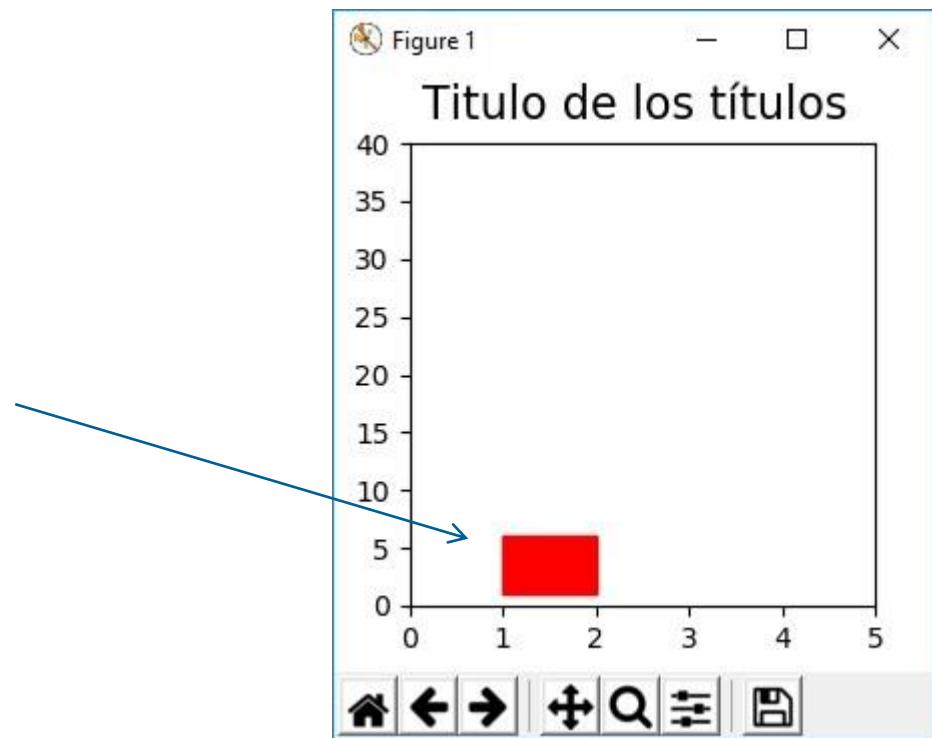
Figure, Subplot, Suptitle, ax.fill

Aparece un polígono delimitado por:

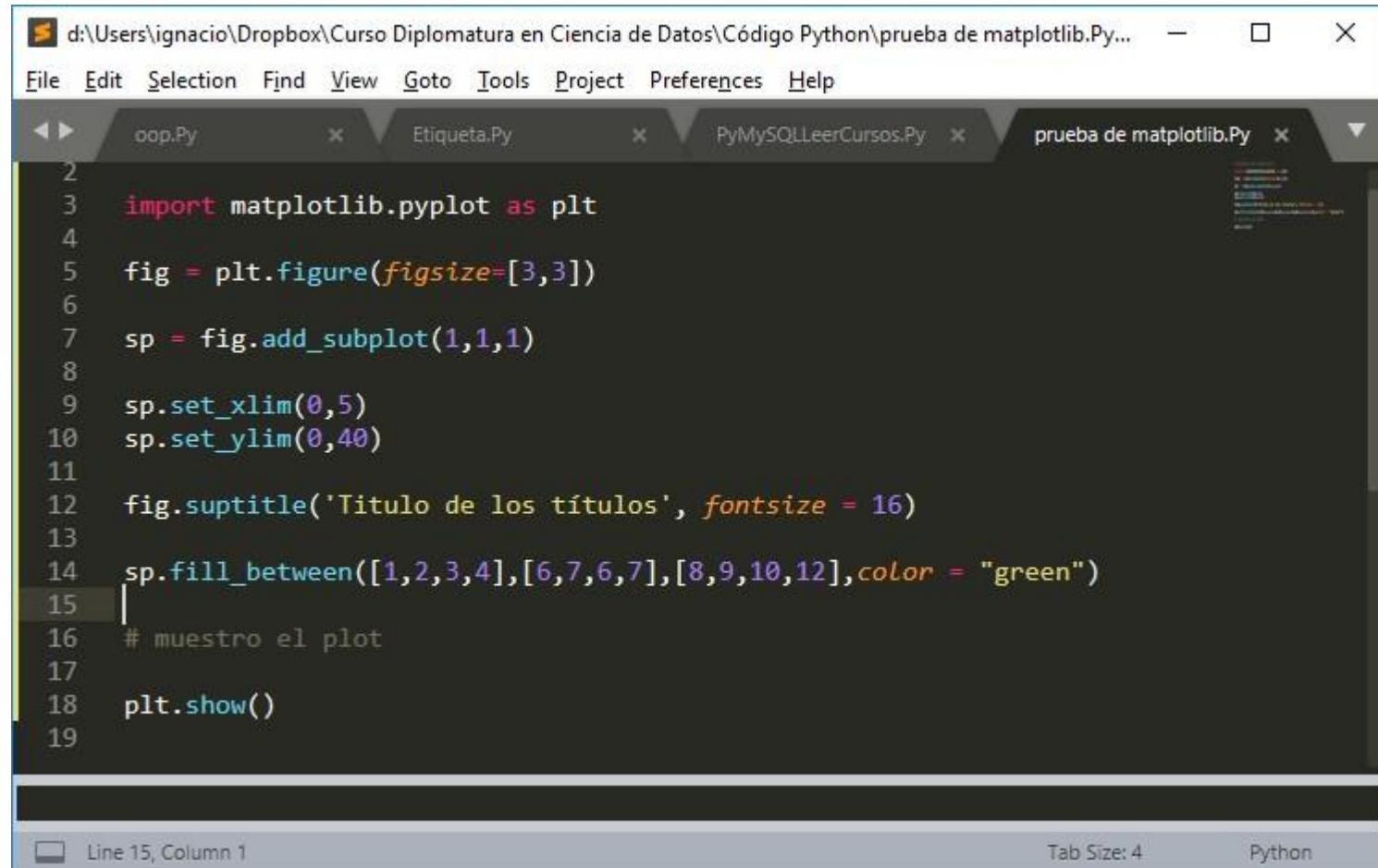
X: [1,2,2,1]

Y: [1,1,6,6]

Y relleno de rojo



Figure, Subplot, Suptitle, ax.fill_between



```
d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\prueba de matplotlib.Py... - □ X
File Edit Selection Find View Goto Tools Project Preferences Help
oop.py Etiqueta.py PyMySQLLeerCursos.py prueba de matplotlib.py
2
3     import matplotlib.pyplot as plt
4
5     fig = plt.figure(figsize=[3,3])
6
7     sp = fig.add_subplot(1,1,1)
8
9     sp.set_xlim(0,5)
10    sp.set_ylim(0,40)
11
12    fig.suptitle('Titulo de los títulos', fontsize = 16)
13
14    sp.fill_between([1,2,3,4],[6,7,6,7],[8,9,10,12],color = "green")
15
16    # muestro el plot
17
18    plt.show()
19
```

Line 15, Column 1 Tab Size: 4 Python

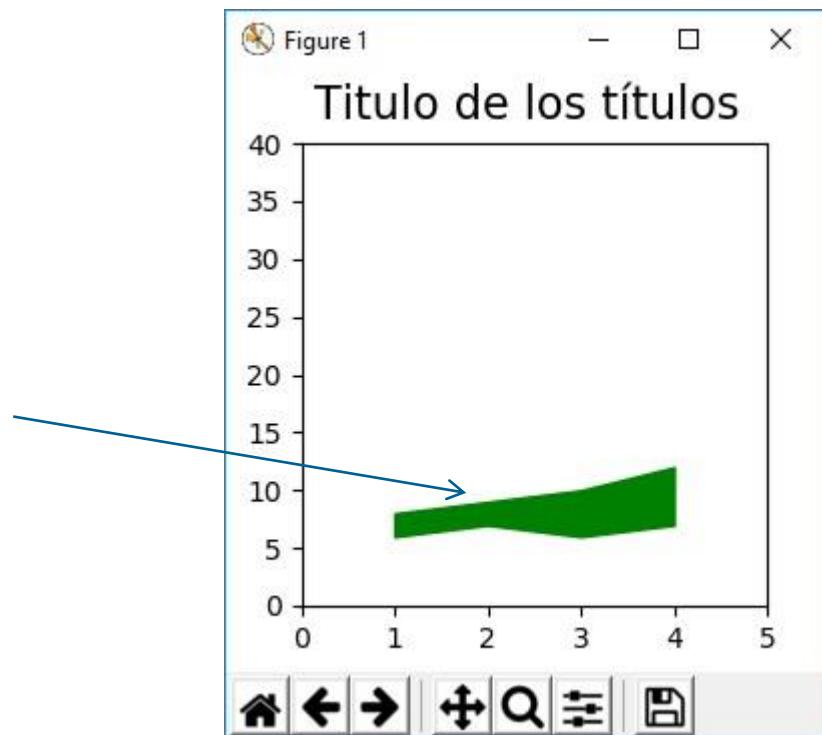
Figure, Subplot, Suptitle, ax.fill_between

Aparece un polígono delimitado por:

X: [1, 2, 3, 4]

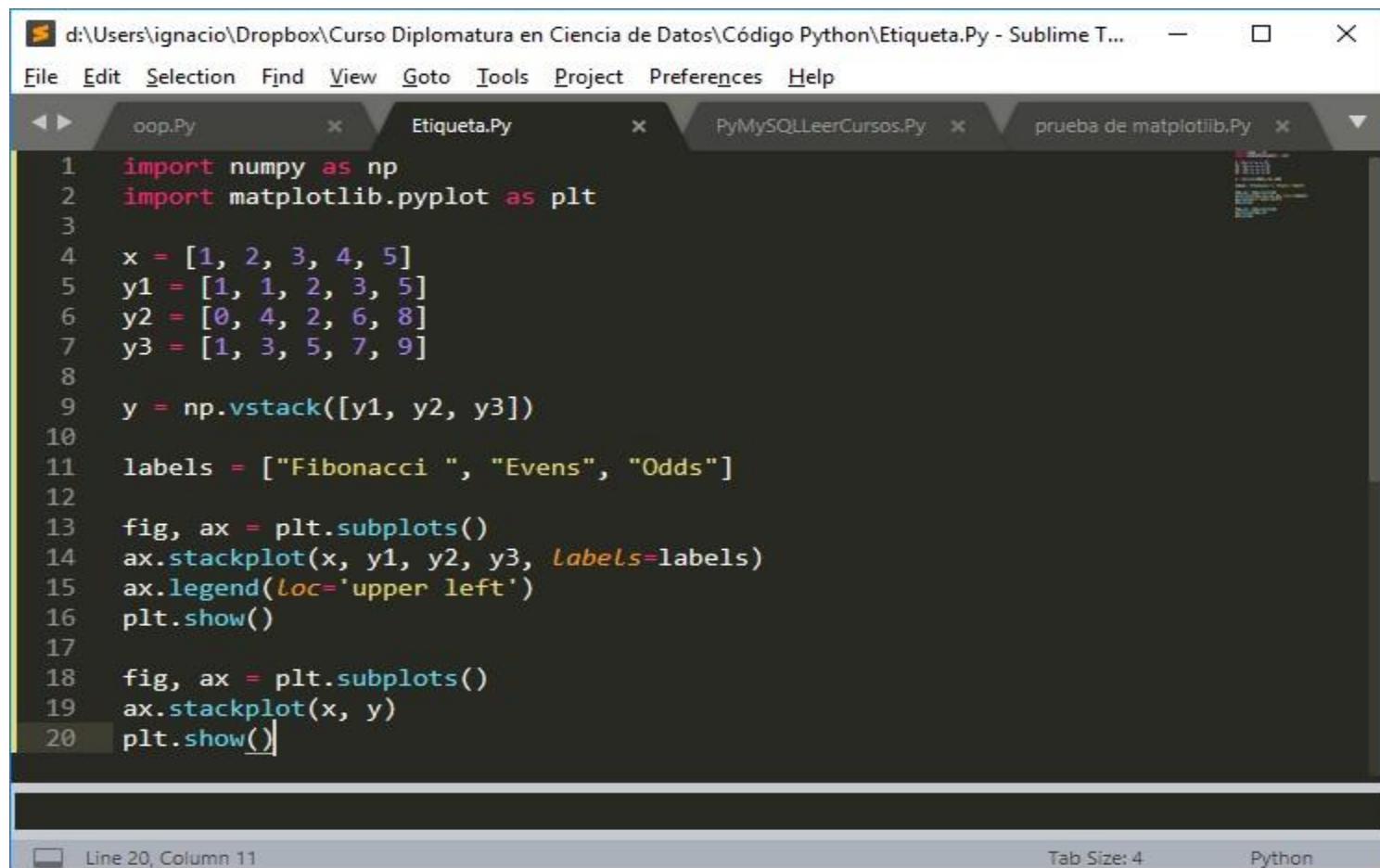
Y1: [6, 7, 6, 7]

Y2: [8, 9, 10, 12]



Y relleno de verde

Figure, Subplot, Suptitle, stackplot



The screenshot shows a Sublime Text window with four tabs open:

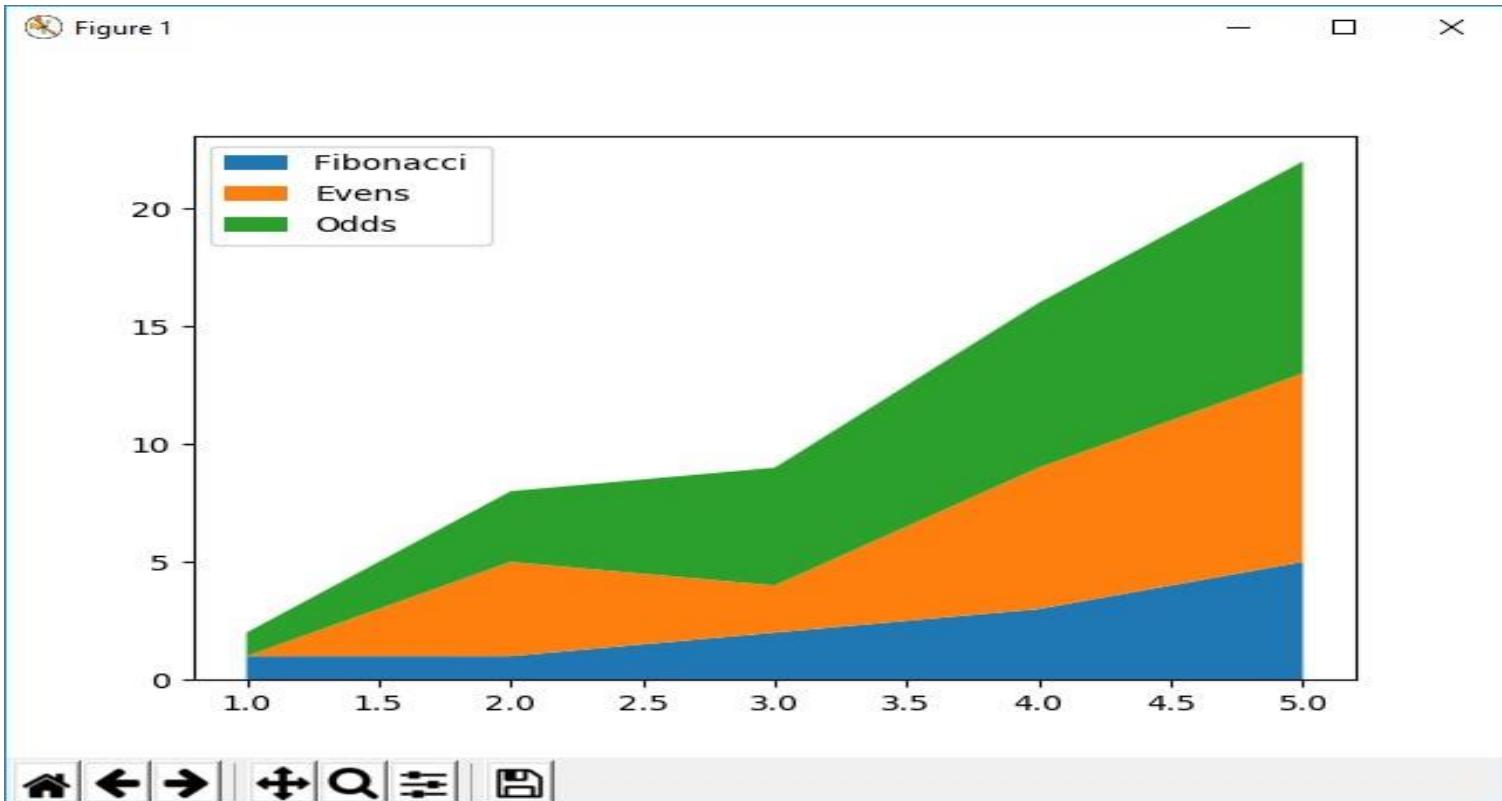
- oop.py
- Etiqueta.py (the active tab)
- PyMySQLLeerCursos.py
- prueba de matplotlib.py

The code in Etiqueta.py is as follows:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = [1, 2, 3, 4, 5]
5 y1 = [1, 1, 2, 3, 5]
6 y2 = [0, 4, 2, 6, 8]
7 y3 = [1, 3, 5, 7, 9]
8
9 y = np.vstack([y1, y2, y3])
10
11 labels = ["Fibonacci ", "Evens", "Odds"]
12
13 fig, ax = plt.subplots()
14 ax.stackplot(x, y1, y2, y3, labels=labels)
15 ax.legend(loc='upper left')
16 plt.show()
17
18 fig, ax = plt.subplots()
19 ax.stackplot(x, y)
20 plt.show()
```

At the bottom of the window, it says "Line 20, Column 11".

Figure, Subplot, Suptitle, ax.stackplot()



d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Etiqueta.Py - Sublime Text (U... - X

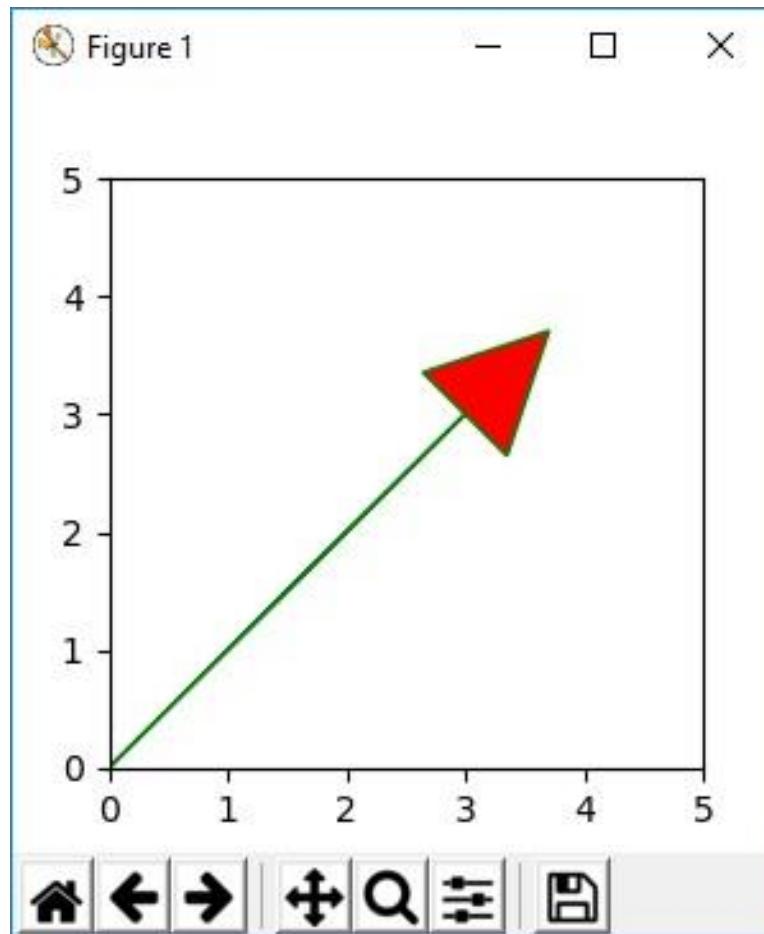
File Edit Selection Find View Goto Tools Project Preferences Help

oop.Py Etiqueta.Py PyMySQLLeerCursos.Py prueba de matplotlib.Py

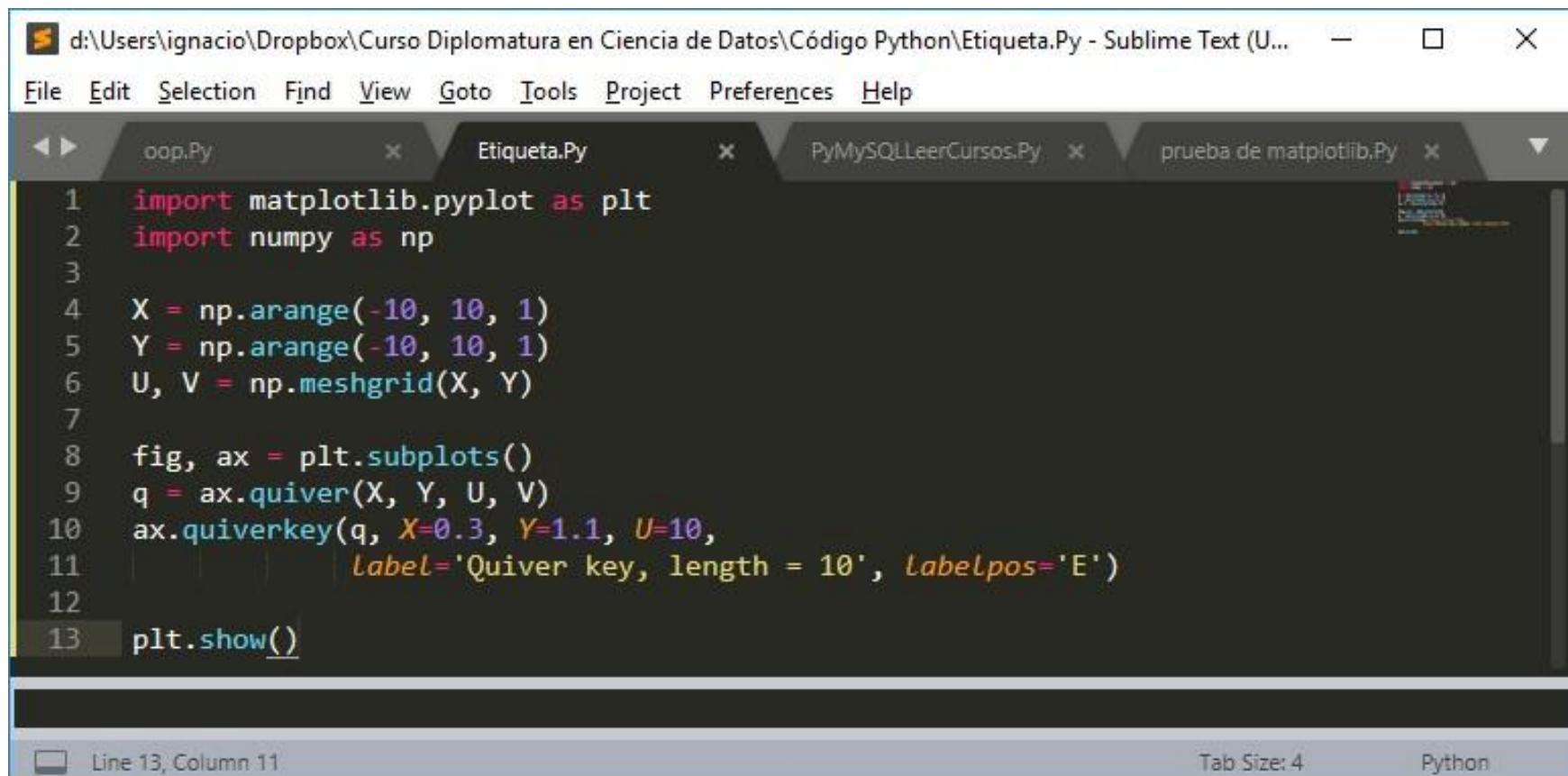
```
1 import matplotlib.pyplot as plt
2
3 fig = plt.figure(figsize=[3,3])
4 sp = fig.add_subplot(1,1,1)
5 sp.set_xlim(0,5)
6 sp.set_ylim(0,5)
7 sp.arrow(0,0,3,3,head_width = 1, head_length = 1, fc = 'red', ec = 'green')
8 plt.show()
9
```

Line 5, Column 17 Tab Size: 4 Python

Figure, Subplot, Suptitle, ax.arrow



Figure, Subplot, Suptitle, ax.quiver()

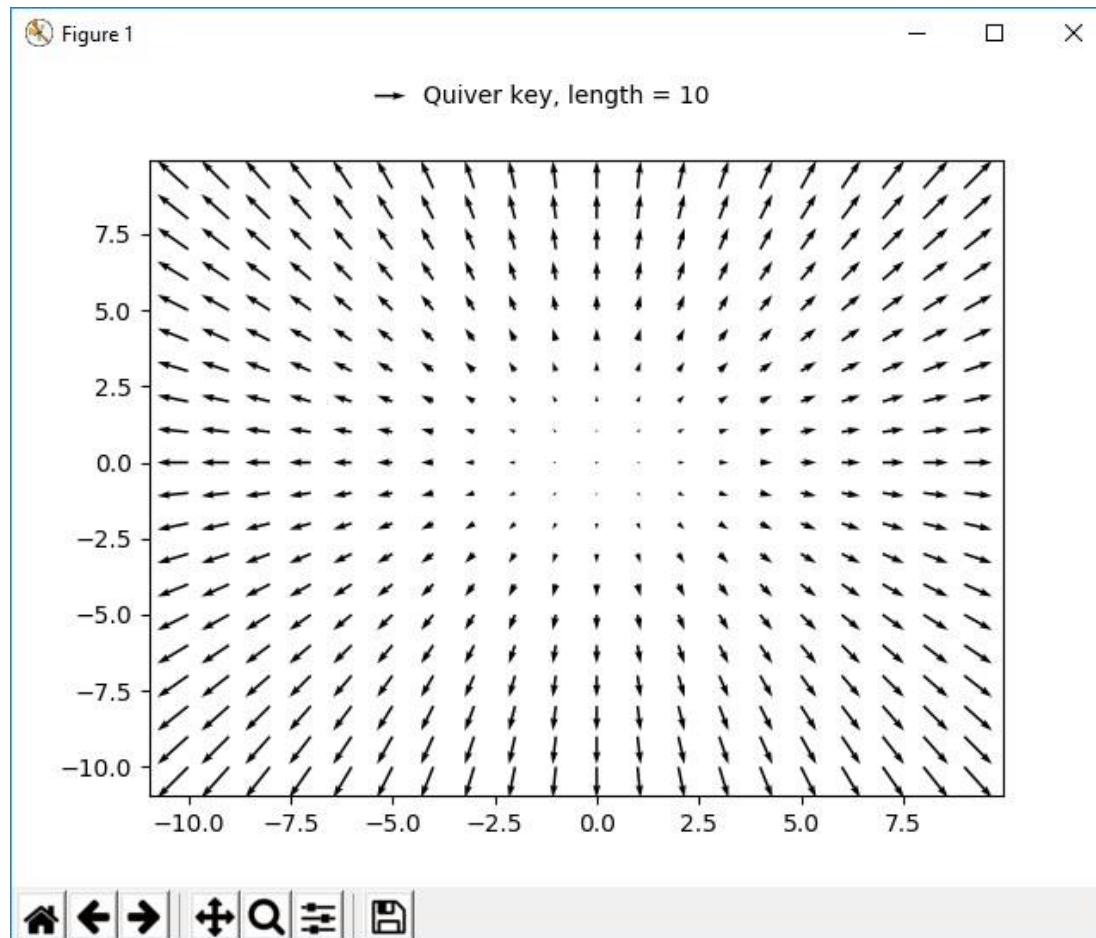


The screenshot shows a Sublime Text window with four tabs: oop.py, Etiqueta.py (which is the active tab), PyMySQLLeerCursos.py, and prueba de matplotlib.py. The code in Etiqueta.py is as follows:

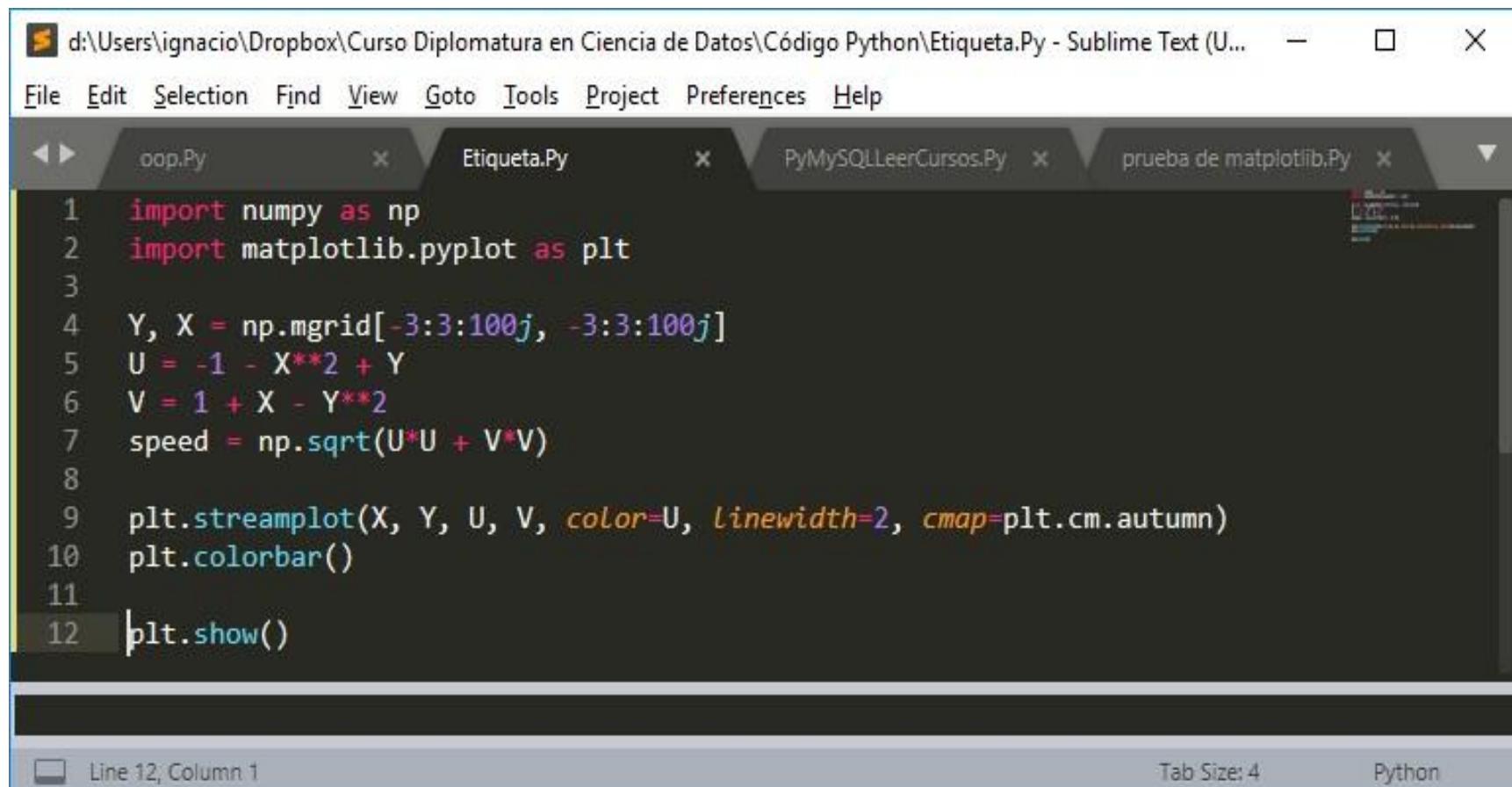
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 X = np.arange(-10, 10, 1)
5 Y = np.arange(-10, 10, 1)
6 U, V = np.meshgrid(X, Y)
7
8 fig, ax = plt.subplots()
9 q = ax.quiver(X, Y, U, V)
10 ax.quiverkey(q, X=0.3, Y=1.1, U=10,
11                 label='Quiver key, length = 10', labelpos='E')
12
13 plt.show()
```

The status bar at the bottom indicates "Line 13, Column 11".

Figure, Subplot, Suptitle, ax.quiver()



Figure, Subplot, Suptitle, ax.streamplot()

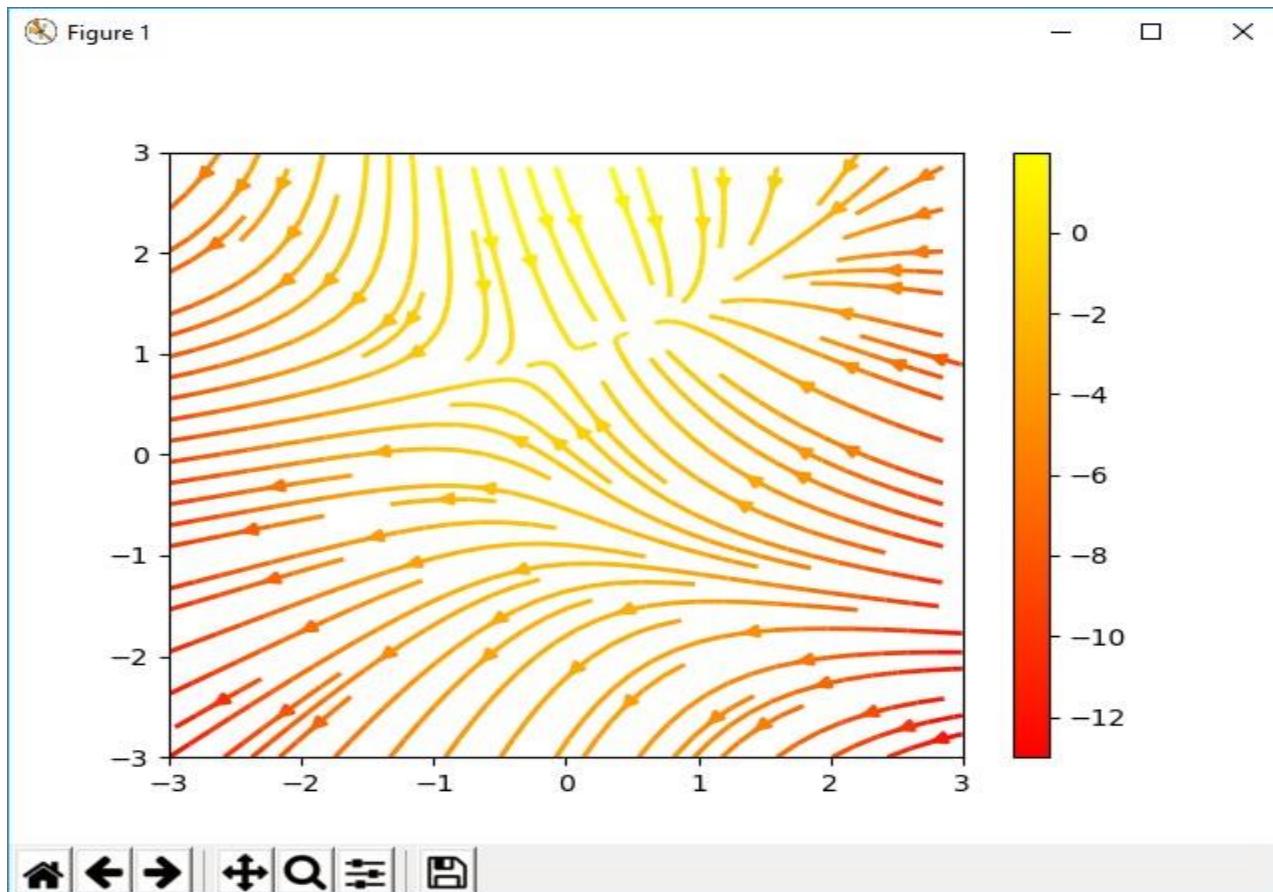


The screenshot shows a Sublime Text window with several tabs open. The active tab contains the following Python code:

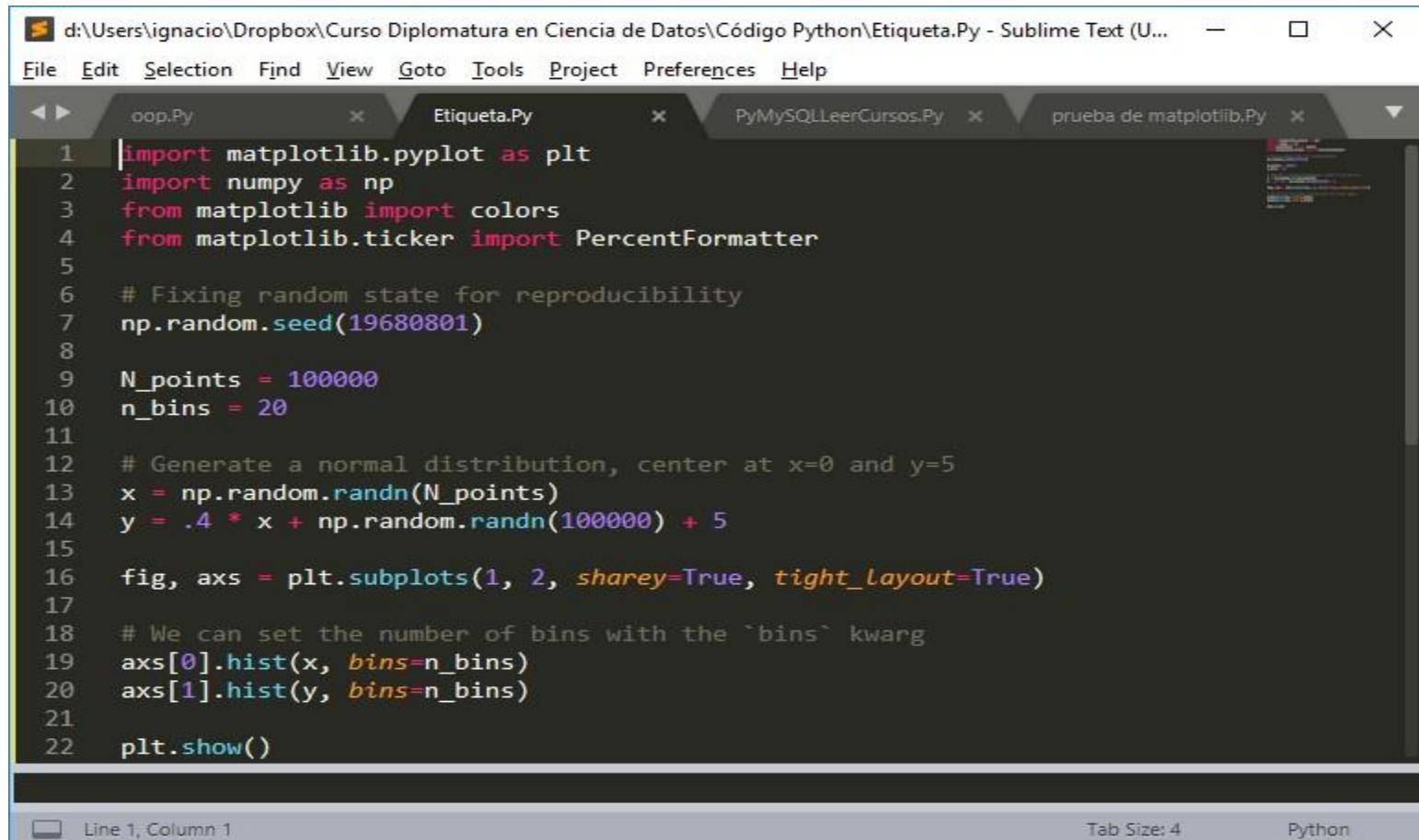
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 Y, X = np.mgrid[-3:3:100j, -3:3:100j]
5 U = -1 - X**2 + Y
6 V = 1 + X - Y**2
7 speed = np.sqrt(U*U + V*V)
8
9 plt.streamplot(X, Y, U, V, color=U, linewidth=2, cmap=plt.cm.autumn)
10 plt.colorbar()
11
12 plt.show()
```

The status bar at the bottom indicates "Line 12, Column 1", "Tab Size: 4", and "Python".

Figure, Subplot, Suptitle, ax.streamplot()



Figure, Subplot, Suptitle, ax.hist()

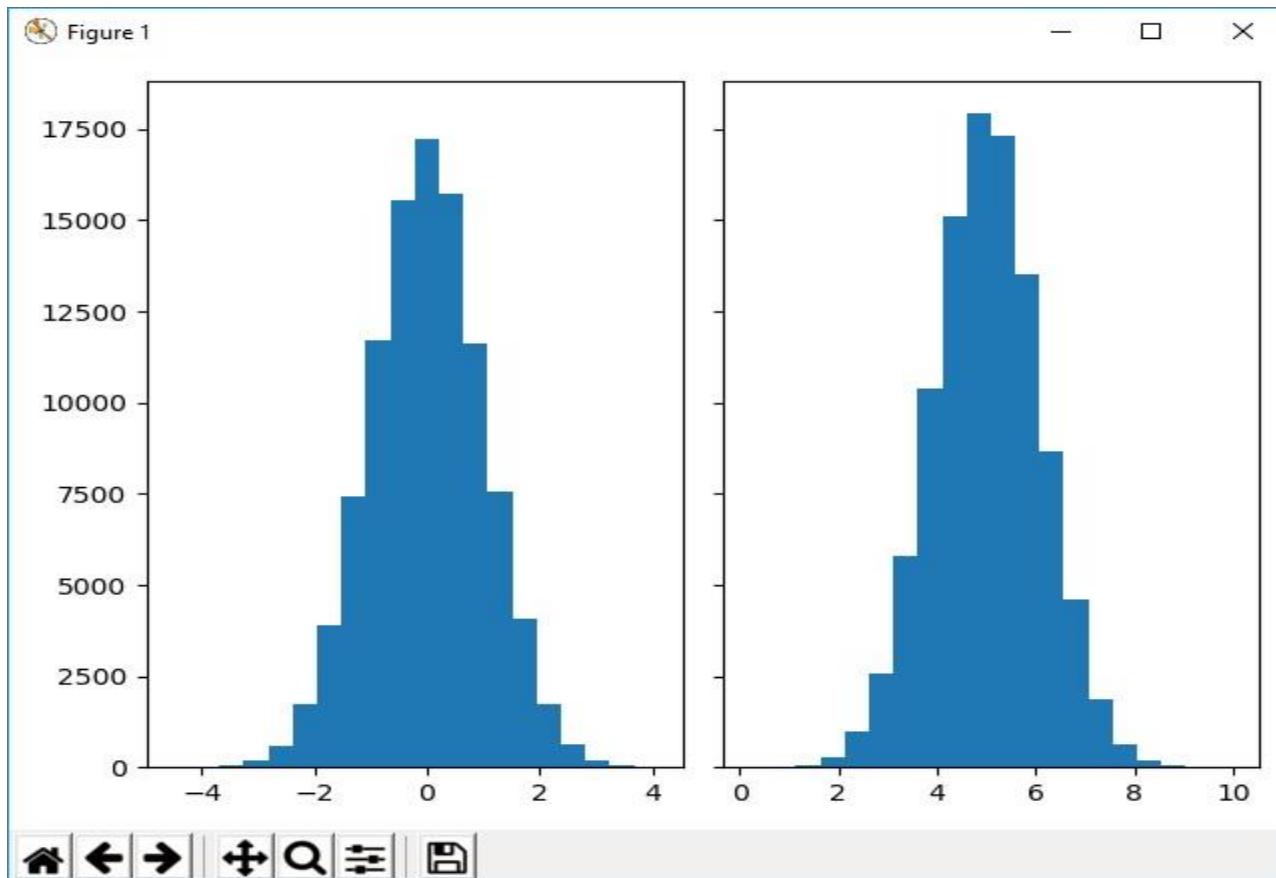


The screenshot shows a Sublime Text window with four tabs: 'oop.py', 'Etiqueta.py' (which is the active tab), 'PyMySQLLeerCursos.py', and 'prueba de matplotlib.py'. The code in 'Etiqueta.py' is as follows:

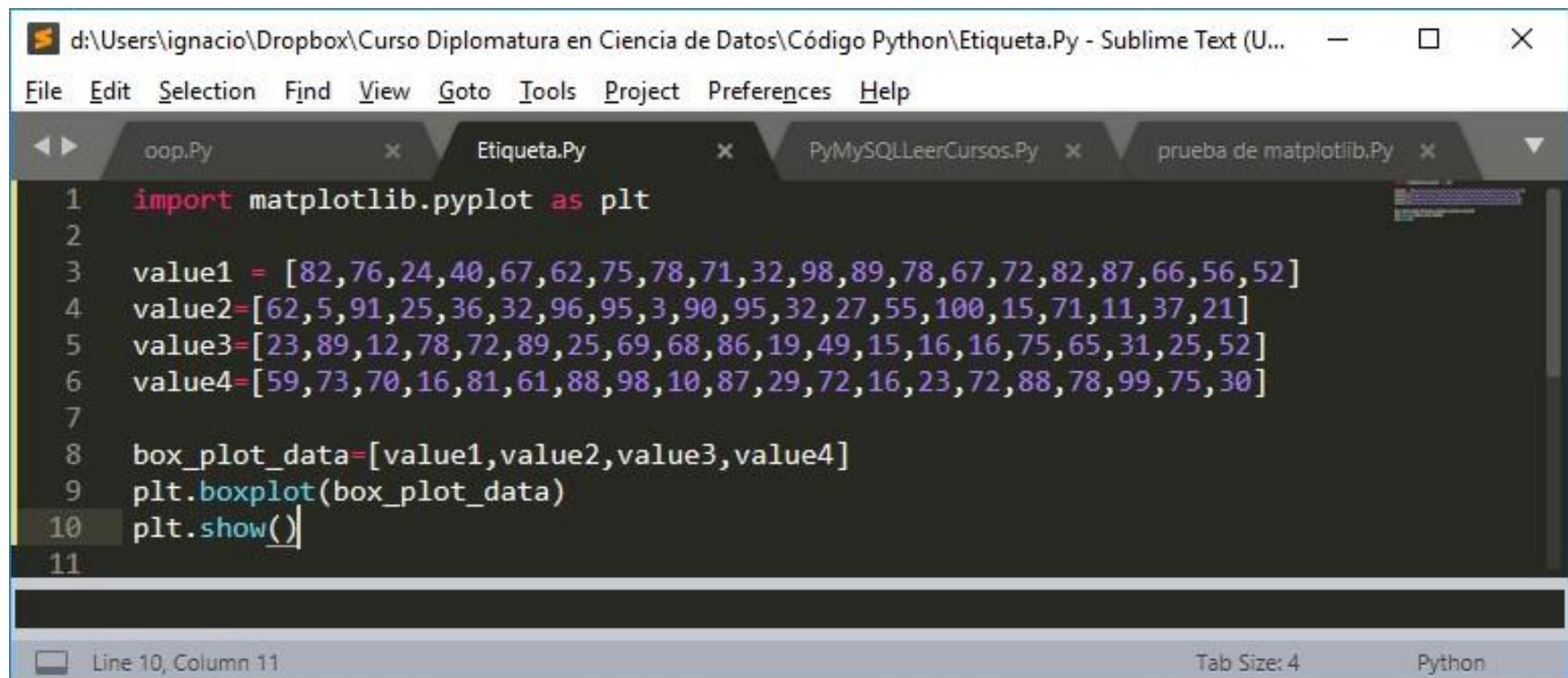
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from matplotlib import colors
4 from matplotlib.ticker import PercentFormatter
5
6 # Fixing random state for reproducibility
7 np.random.seed(19680801)
8
9 N_points = 100000
10 n_bins = 20
11
12 # Generate a normal distribution, center at x=0 and y=5
13 x = np.random.randn(N_points)
14 y = .4 * x + np.random.randn(100000) + 5
15
16 fig, axs = plt.subplots(1, 2, sharey=True, tight_layout=True)
17
18 # We can set the number of bins with the `bins` kwarg
19 axs[0].hist(x, bins=n_bins)
20 axs[1].hist(y, bins=n_bins)
21
22 plt.show()
```

At the bottom of the window, it says 'Line 1, Column 1' and 'Tab Size: 4'.

Figure, Subplot, Suptitle, ax.hist()



Figure, Subplot, Suptitle, ax.boxplot()

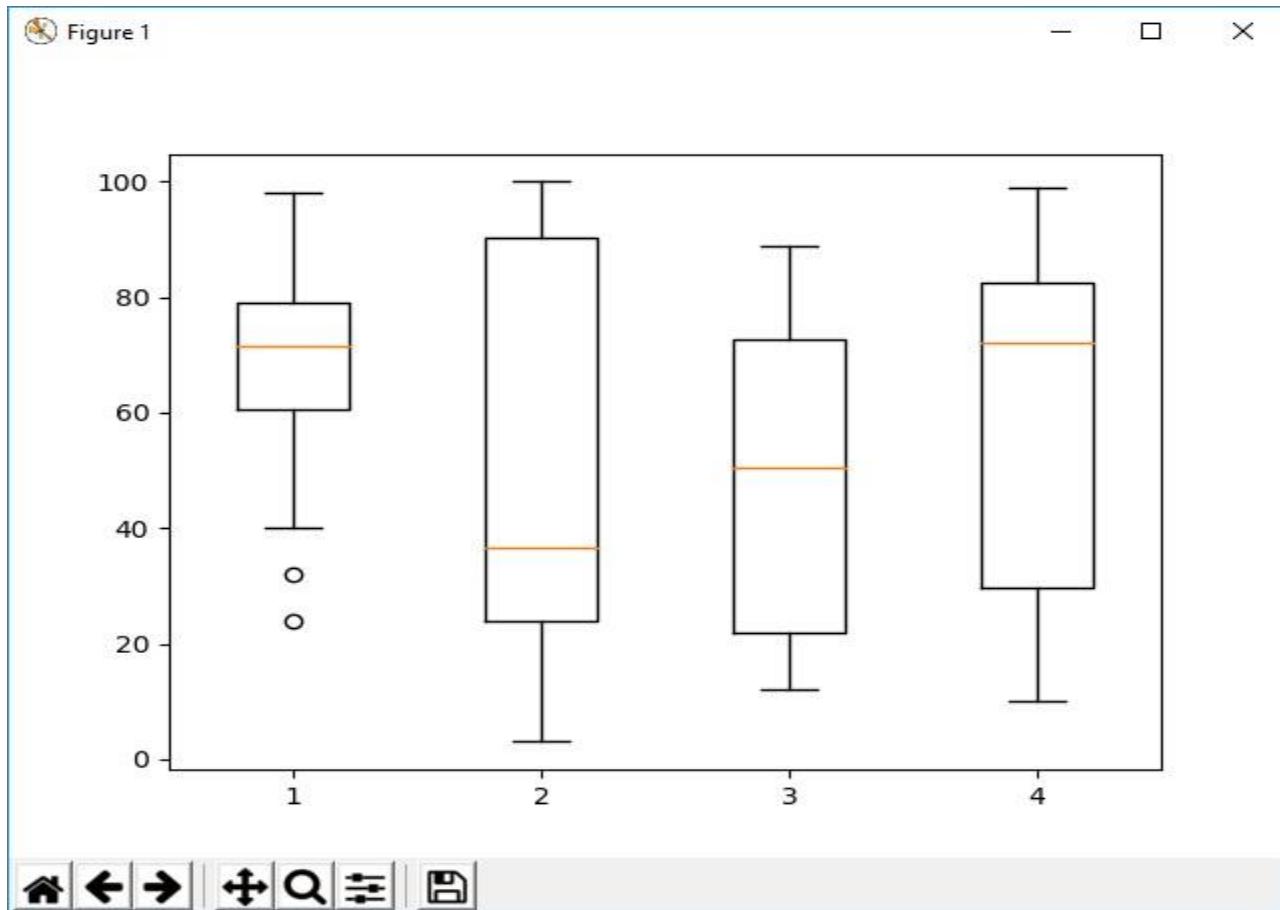


The screenshot shows a Sublime Text window with multiple tabs open. The active tab contains Python code for generating a box plot. The code imports matplotlib.pyplot, defines four lists of numerical values (value1, value2, value3, value4), and then uses plt.boxplot() to create a box plot of these values. The code is as follows:

```
import matplotlib.pyplot as plt
value1 = [82, 76, 24, 40, 67, 62, 75, 78, 71, 32, 98, 89, 78, 67, 72, 82, 87, 66, 56, 52]
value2=[62, 5, 91, 25, 36, 32, 96, 95, 3, 90, 95, 32, 27, 55, 100, 15, 71, 11, 37, 21]
value3=[23, 89, 12, 78, 72, 89, 25, 69, 68, 86, 19, 49, 15, 16, 16, 75, 65, 31, 25, 52]
value4=[59, 73, 70, 16, 81, 61, 88, 98, 10, 87, 29, 72, 16, 23, 72, 88, 78, 99, 75, 30]
box_plot_data=[value1,value2,value3,value4]
plt.boxplot(box_plot_data)
plt.show()
```

The status bar at the bottom indicates "Line 10, Column 11".

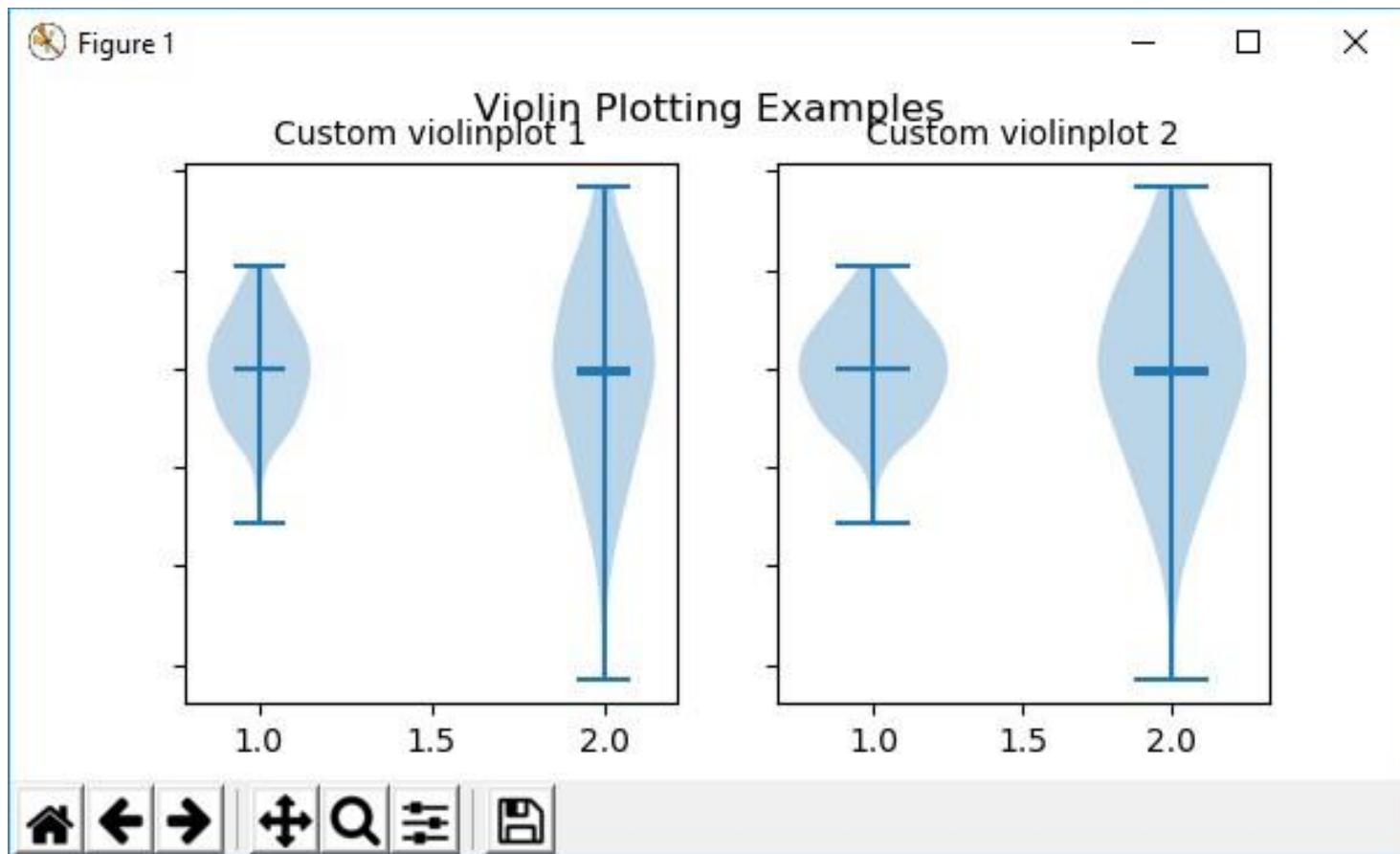
Figure, Subplot, Suptitle, ax.fill_boxplot()



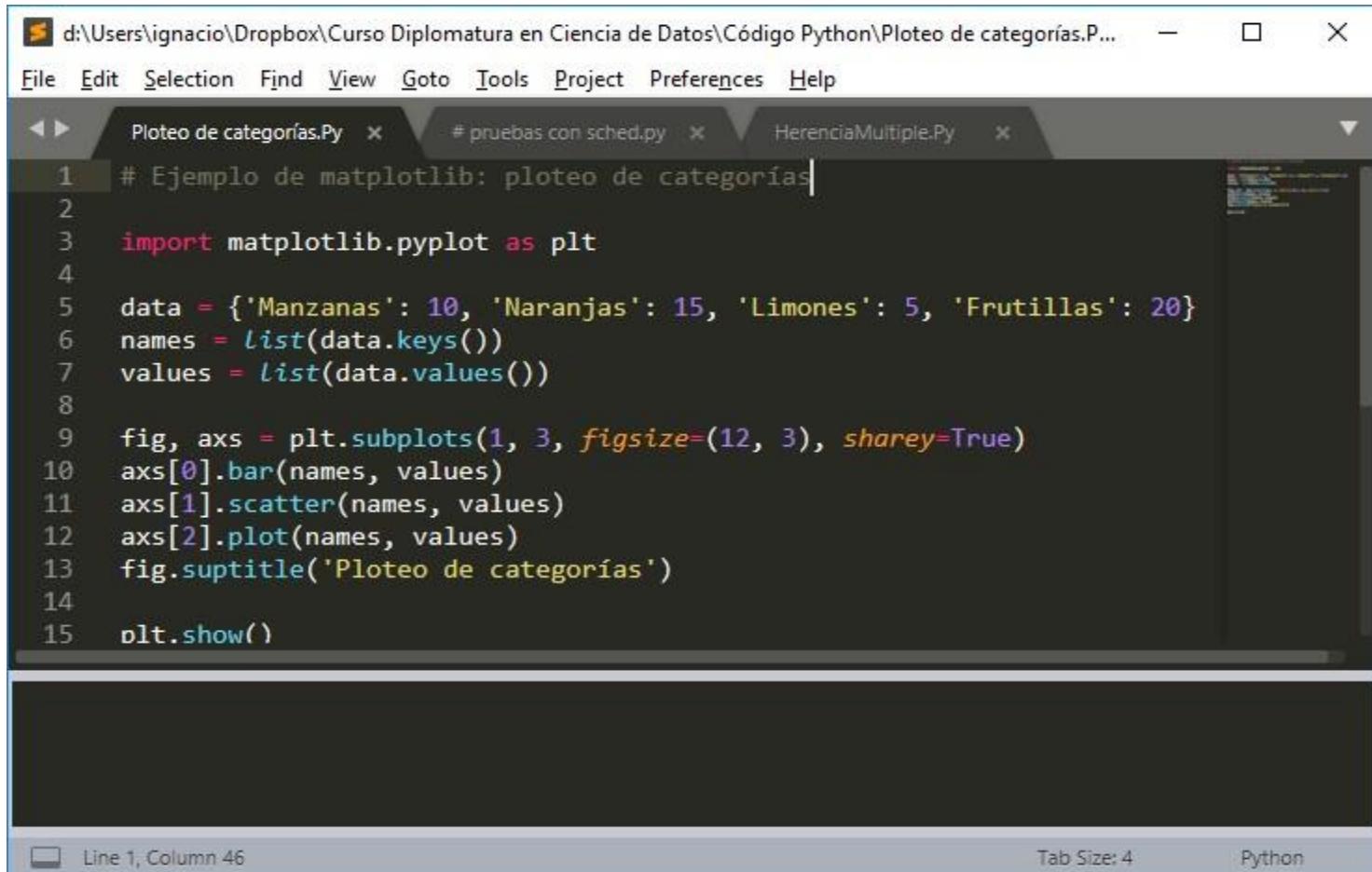
Figure, Subplot, Suptitle, ax.violinplot()

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Fixing random state for reproducibility
5 np.random.seed(19680801)
6
7 # fake data
8 fs = 10 # fontsize
9 pos = [1, 2]
10 data = [np.random.normal(0, std, size=100) for std in pos]
11 fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(6, 3))
12
13 axes[0].violinplot(data, pos, points=20, widths=0.3,
14                      showmeans=True, showextrema=True, showmedians=True)
15 axes[0].set_title('Custom violinplot 1', fontsize=fs)
16
17 axes[1].violinplot(data, pos, points=40, widths=0.5,
18                      showmeans=True, showextrema=True, showmedians=True,
19                      bw_method='silverman')
20 axes[1].set_title('Custom violinplot 2', fontsize=fs)
21
22 for ax in axes.flat:
23     ax.set_yticklabels([])
24
25 fig.suptitle("Violin Plotting Examples")
26 fig.subplots_adjust(hspace=0.4)
27 plt.show()
```

Figure, Subplot, Suptitle, ax.violinplot()



Ploteo de categorías



The screenshot shows a code editor window with the following details:

- Title Bar:** d:\Users\ignacio\Dropbox\Curso Diplomatura en Ciencia de Datos\Código Python\Ploteo de categorías.Py
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Tab Bar:** Ploteo de categorías.Py (selected), # pruebas con sched.py, HerenciaMultiple.Py
- Code Area:**

```
1 # Ejemplo de matplotlib: ploteo de categorías
2
3 import matplotlib.pyplot as plt
4
5 data = {'Manzanas': 10, 'Naranjas': 15, 'Limones': 5, 'Frutillas': 20}
6 names = list(data.keys())
7 values = list(data.values())
8
9 fig, axs = plt.subplots(1, 3, figsize=(12, 3), sharey=True)
10 axs[0].bar(names, values)
11 axs[1].scatter(names, values)
12 axs[2].plot(names, values)
13 fig.suptitle('Ploteo de categorías')
14
15 plt.show()
```
- Status Bar:** Line 1, Column 46, Tab Size: 4, Python

Ploteo de categorías

Figure 1

- □ X

Ploteo de categorías

