



FUNDAMENTO DE PROGRAMACIÓN

José Antonio Gonzales Jurado

Desarrollo Seguro | eJPTv1.0 | eWPTv2.0 | eWPTXv2.0 | Offensive Security |
SFPC | CyberArk Certified Trustee | ISO27001F

"Si yo tuviera ocho horas para cortar un árbol, me gustaría
pasar los primeros seis de ellos afilando mi hacha"



CRONOGRAMA:

Día 1:

- Fundamento de Programación.
 - Conceptos Básicos.
 - Números.
 - Cadena de Caracteres (Textos).
 - Lista.
 - Lectura por teclado.
 - Operadores y Expresiones.
 - Controladores de Flujo.
-

EDITORES DE CODIGO



Sublime Text



Atom



Visual Studio Code





Sublime Text 3

- Ligero
- Multiplataforma
- Adaptable
- Simpleza



Atom

- Funcionalidades extra
- Integración con Git
- Personalizable



Visual Studio Code

- Autocompletado de código
 - Debugging
 - Snippets

INTRODUCCIÓN

Python nace hacia finales de los años 80, cuando en navidad de 1989, **Guido Van Rossum**, un científico nacido en países bajos decide empezar un proyecto para darle continuidad al lenguaje de programación ABC.



CARACTERÍSTICAS DE PYTHON

- Es un lenguaje simple, por lo que el pseudo-código natural de Python es una de sus mayores fortalezas.
- Es un lenguaje de alto nivel.
- Multiplataforma.
- Extensas Librerías.
- Potente y Flexible.
- Fácil de aprender.





Scripting

¿QUE PUEDO HACER CON PYTHON



Web



Aplicaciones Desktop



Aplicaciones Móviles



Machine Learning

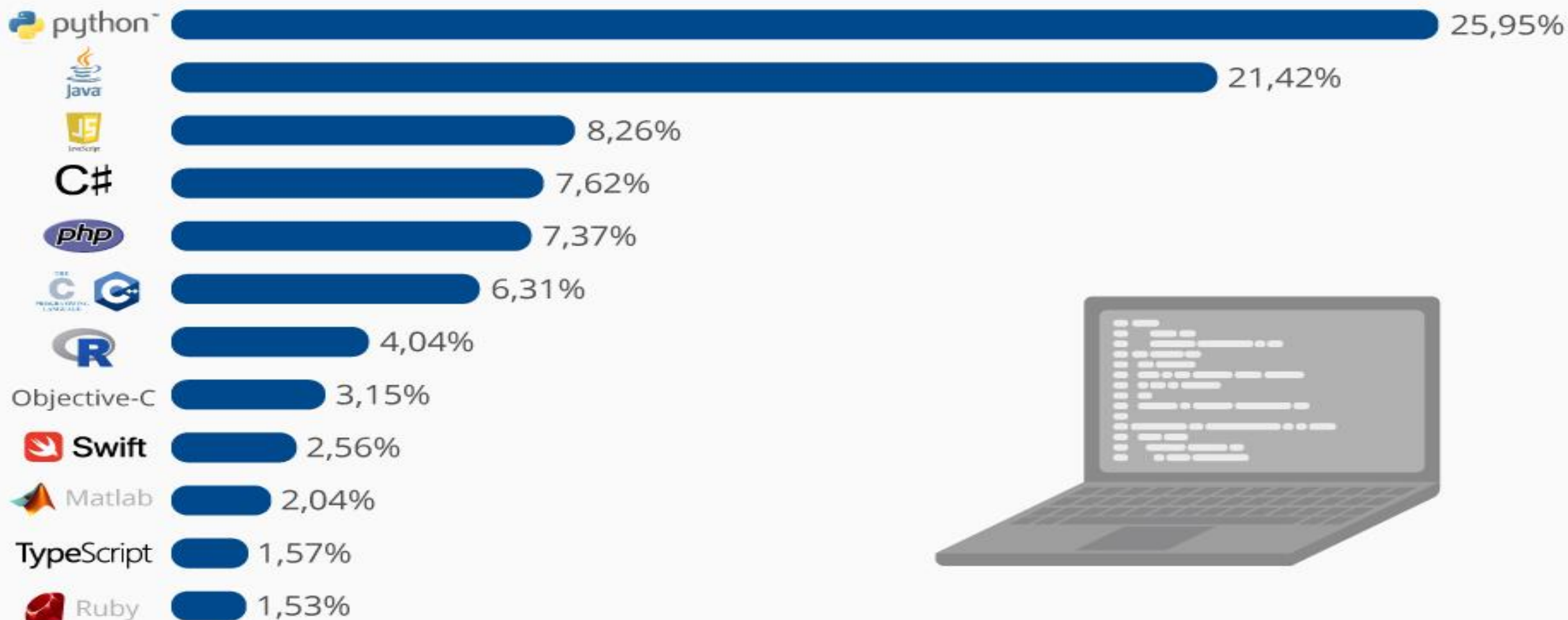


SITIOS QUE USAN PYTHON



Los lenguajes de programación más usados

Porcentaje de uso de los lenguajes de programación más populares del mundo*



* Datos procedentes del Índice PYPL. Este se basa en las búsquedas en Google de tutoriales de lenguajes de programación.

Datos de enero de 2019

Fuente: PYPL

- Proceso de Instalación de Python en Linux y Sublime Text 3:
 - Podemos instalar paquetes en python usando pip, que es una herramienta de gestión de paquetes para python. Aquí te mostraremos cómo instalar Python en Ubuntu 18.04.
 - A continuación, actualizaremos la lista de paquetes usando el siguiente comando. Puedes ejecutar este comando como administrador o como usuario normal con privilegios sudo.

```
root@localhost: ~  
root@localhost:~# sudo apt update  
Hit:1 http://pe.archive.ubuntu.com/ubuntu bionic InRelease  
Get:2 http://pe.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]  
Get:3 http://pe.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]  
Get:4 http://pe.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]  
Fetched 252 kB in 2s (165 kB/s)  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
39 packages can be upgraded. Run 'apt list --upgradable' to see them.  
root@localhost:~#
```

- Python 3 se instala por defecto en la distribución Linux Ubuntu 18.04. Entonces necesitaremos instalar el paquete python3-pip usando el siguiente comando. También instalará las dependencias requeridas, agilizando así el proceso.

```
root@localhost: ~  
root@localhost:~# apt install python3-pip  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done
```

- Para instalar una versión anterior de Python, puedes usar el siguiente comando:

```
root@localhost: ~  
root@localhost:~# apt install python  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python-minimal python2.7 python2.7-minimal  
Suggested packages:  
  python-doc python-tk python2.7-doc binfmt-support  
The following NEW packages will be installed:  
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-minimal python2.7 python2.7-minimal  
0 upgraded, 7 newly installed, 0 to remove and 38 not upgraded.  
Need to get 3965 kB of archives.  
After this operation, 16.8 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```


- Al igual que instalamos el pip en Python 3, también podemos instalar pip en una versión anterior, usando el siguiente comando:

```
root@localhost: ~  
root@localhost:~# apt install python-pip  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libpython-all-dev libpython-dev libpython2.7 libpython2.7-dev python-all python-all-dev python-asn1crypto python-cffi-backend python-crypto python-cryptography python-dbus  
  python-dev python-enum34 python-gi python-idna python-ipaddress python-keyring python-keyrings.alt python-pkg-resources python-secretstorage python-setuptools python-six  
  python-wheel python-xdg python2.7-dev  
Suggested packages:  
  python-crypto-doc python-cryptography-doc python-cryptography-vectors python-dbus-dbg python-dbus-doc python-enum34-doc python-gi-cairo gnome-keyring libkf5wallet-bin  
  gir1.2-gnomekeyring-1.0 python-fs python-gdata python-keyczar python-secretstorage-doc python-setuptools-doc  
The following NEW packages will be installed:  
  libpython-all-dev libpython-dev libpython2.7 libpython2.7-dev python-all python-all-dev python-asn1crypto python-cffi-backend python-crypto python-cryptography python-dbus  
  python-dev python-enum34 python-gi python-idna python-ipaddress python-keyring python-keyrings.alt python-pip python-pkg-resources python-secretstorage python-setuptools  
  python-six python-wheel python-xdg python2.7-dev  
0 upgraded, 26 newly installed, 0 to remove and 38 not upgraded.  
Need to get 31.4 MB of archives.  
After this operation, 54.3 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

- Puedes verificar las versiones de cada uno utilizando los siguientes comandos pip:

```
root@localhost: ~  
root@localhost:~# pip --version  
pip 9.0.1 from /usr/lib/python2.7/dist-packages (python 2.7)  
root@localhost:~# pip3 --version  
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)  
root@localhost:~#
```

- Instalación de Sublime Text 3 en Linux:

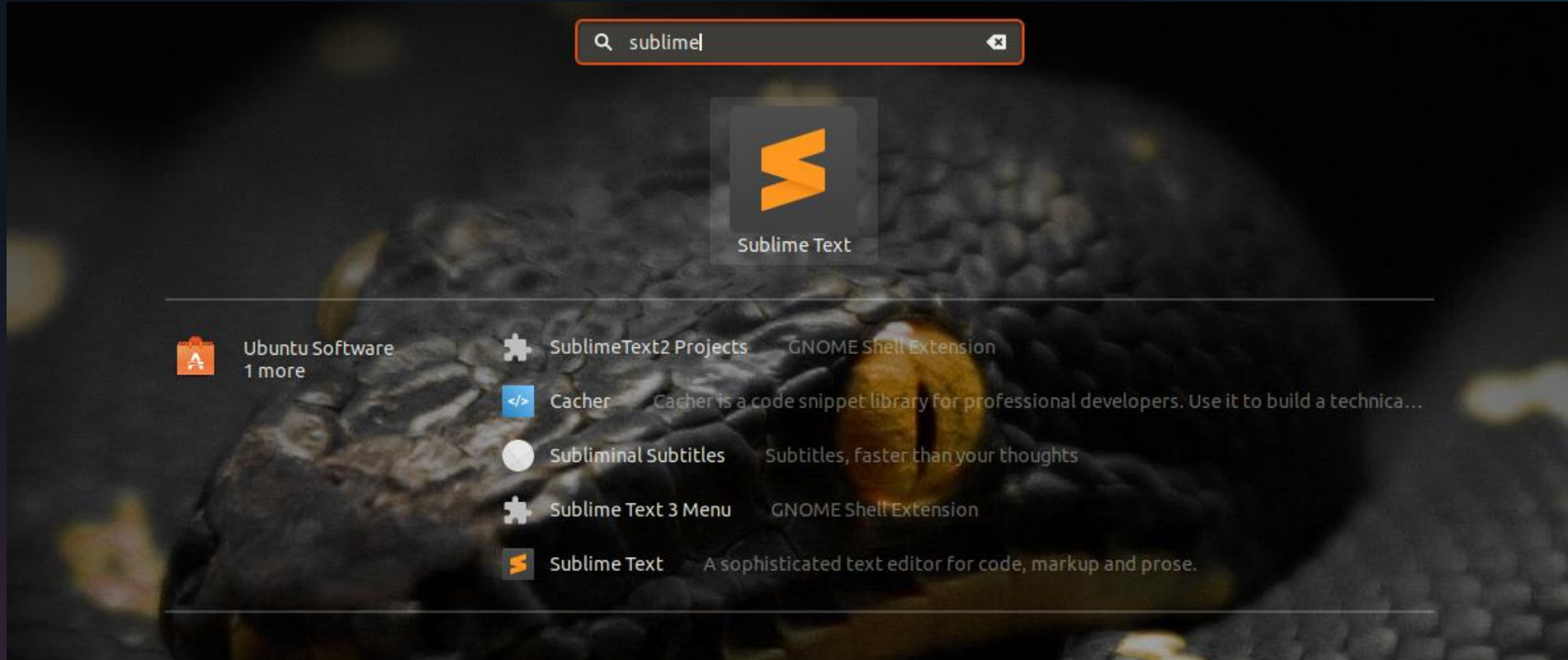
- Para poder descargar e instalar el producto debemos habilitar los repositorios del fabricante incluyendo su llave de firma. Para ello nos iremos a la consola de comandos o shell:
- `$ wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -`
- `$ sudo apt-add-repository "deb https://download.sublimetext.com/ apt/stable/"`

```
root@linux: ~  
File Edit View Search Terminal Help  
root@linux:~# wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -  
OK  
root@linux:~# sudo apt-add-repository "deb https://download.sublimetext.com/ apt/stable/"  
Hit:1 http://pe.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease  
Hit:3 http://pe.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:4 http://pe.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Get:5 https://download.sublimetext.com apt/stable/ InRelease [2.562 B]  
Get:6 https://download.sublimetext.com apt/stable/ Packages [1.026 B]  
Fetched 3.588 B in 1s (3.217 B/s)  
Reading package lists... Done  
root@linux:~# |
```

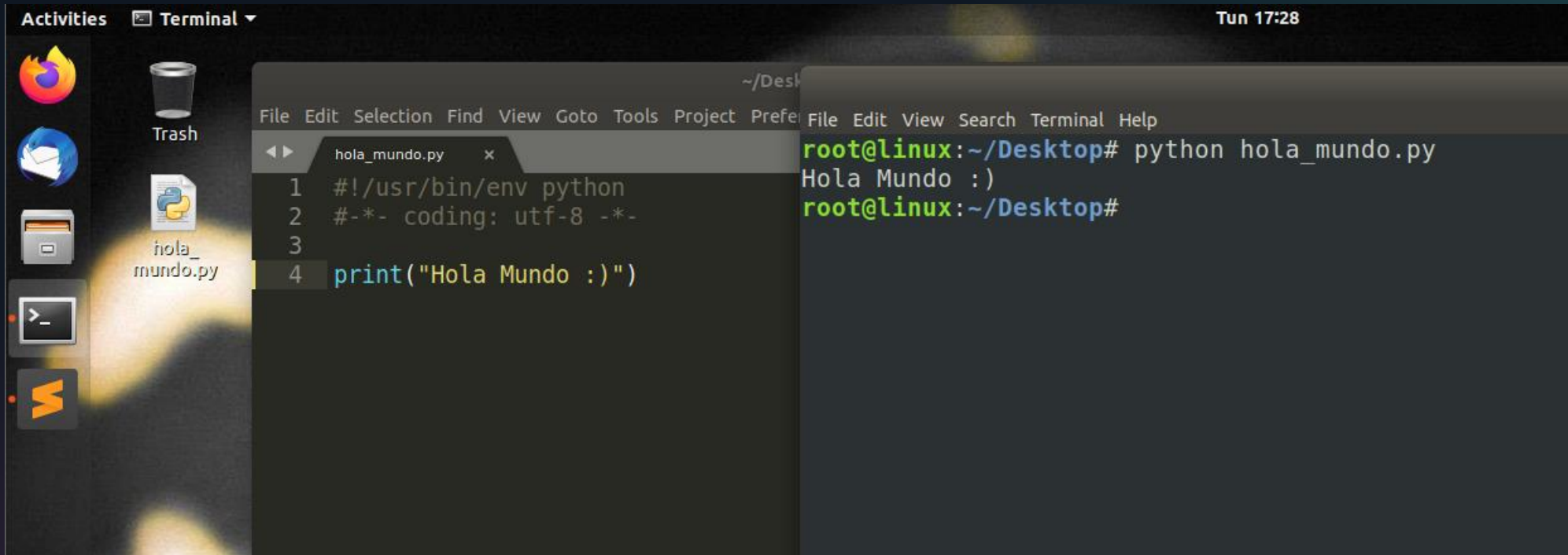
- Una vez añadido, actualizamos repositorios e instalamos:
- \$ apt update
- \$ apt install sublime-text

```
root@linux: ~  
File Edit View Search Terminal Help  
root@linux:~# apt update  
Hit:1 http://pe.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease  
Hit:3 https://download.sublimetext.com apt/stable/ InRelease  
Hit:4 http://pe.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:5 http://pe.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
16 packages can be upgraded. Run 'apt list --upgradable' to see them.  
root@linux:~# apt install sublime-text  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  sublime-text  
0 upgraded, 1 newly installed, 0 to remove and 16 not upgraded.  
Need to get 9.835 kB of archives.  
After this operation, 34,8 MB of additional disk space will be used.  
Get:1 https://download.sublimetext.com apt/stable/ sublime-text 3211 [9.835 kB]  
Fetched 9.835 kB in 3s (2.901 kB/s)  
Selecting previously unselected package sublime-text.  
(Reading database ... 173409 files and directories currently installed.)  
Preparing to unpack .../sublime-text_3211_amd64.deb ...  
Unpacking sublime-text (3211) ...  
Setting up sublime-text (3211) ...  
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...  
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...  
Processing triggers for hicolor-icon-theme (0.17-2) ...  
Processing triggers for mime-support (3.60ubuntu1) ...  
root@linux:~#
```

- Tras instalarlo nos iremos al escritorio y desde el buscador escribiremos el nombre del programa, tal y como se muestra en la imagen:



- Listo, ya tenemos instalado correctamente Sublime Text 3 en Linux.



The screenshot shows a Linux desktop environment. On the left is a dock with icons for Firefox, Trash, a file manager, a document, and a terminal. The main area contains two windows. The first is a Sublime Text 3 editor window titled 'hola_mundo.py' with the following code:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 print("Hola Mundo :)")
```

The second window is a terminal titled 'Terminal' showing the execution of the script:

```
root@linux:~/Desktop# python hola_mundo.py
Hola Mundo :)
root@linux:~/Desktop#
```

The terminal output confirms that the Python script was executed successfully and printed 'Hola Mundo :)'. The system clock in the top right corner indicates 'Tun 17:28'.

- Conceptos Básicos:

- Interprete Interactivo:
- Python ofrece la posibilidad de crear código en un entorno llamado **Interprete Interactivo**. El intérprete funciona de manera similar a una shell de Unix: basta que ejecutemos python.

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Bienvenidos al curso de Python :)"
Bienvenidos al curso de Python :)
>>>
```

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ^Z
```

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>
```

- Usando Python3:

- Para usar el interprete en Python3 solo ejecutamos en el terminal "py".
- Una de la diferencia de utilizar la función print en python3, es que deberemos ingresar (""") para que nuestra función pueda imprimir el mensaje, a diferencia de python2 que solo usamos la función print mas "".

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>py
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

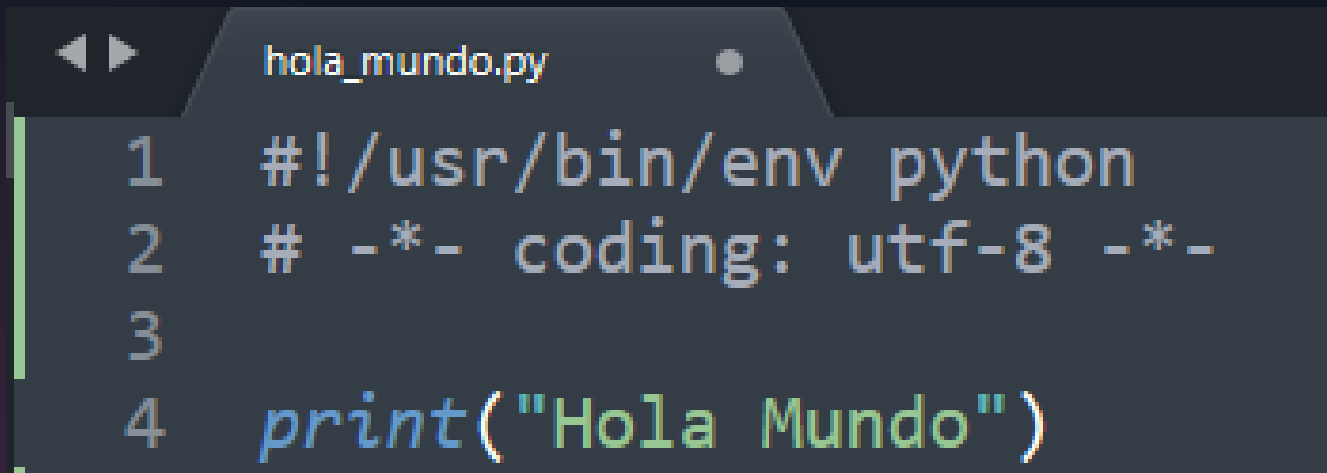
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hola Mundo"
      File "<stdin>", line 1
        print "Hola Mundo"
              ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Hola Mundo")?
>>> print ("Hola Mundo")
Hola Mundo
>>> print (1)
1
>>> print (1+1)
2
>>>
```

- Deficiencia de utilizar el interprete al usar Python3:
 - Al utilizar el interprete lo que no podemos hacer es imprimir el mensaje dentro de doble comillas " ", ya que eso es una función.
 - No podemos compartir nuestro código a un amigo, por lo que el interprete se ejecuta en tiempo real.
 - Si cerramos el terminal por accidente perdemos todo el contenido.
 - A continuación en la siguiente imagen veremos como podemos corregir esta problemática.

- Codificación de Código Fuente:

- De forma predeterminada, los archivos fuente de Python se tratan como codificados en UTF-8. Para mostrar todos estos caracteres correctamente, tu editor debe reconocer que el archivo es UTF-8, y debe usar una fuente que admita todos los caracteres del archivo.
- En la siguiente imagen estamos agregando una almhuadilla (#) y un símbolo de exclamación (!), para nuestra codificación de nuestro código fuente.



```
hola_mundo.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  print("Hola Mundo")
```

- Tipos de Datos:

- Los tipos de Datos son mutables (modificables) o inmutables (no modificables).
- Los tipos inmutables se manipulan copiando valores (numéricos, cadenas de caracteres y tuplas), los tipos mutables se manipulan por punteros (listas, diccionarios).
- Los tipos de datos en Python son secuencias (cadena de caracteres, tuplas y listas) o colecciones (diccionarios).

- **Strings:**

- Estos tipos de datos se componen mayormente de letras y símbolos y van entre comillas, ejemplo "Hola mundo!"

- **Enteros:**

- Son aquellos números enteros que no contienen decimales, ejemplo: 4, 6, 9

- **Flotantes:**

- Son aquellos números reales que si contienen decimales, como por ejemplo: 1.2, 4.2, 3.5

- **Booleanos:**

- Estos son valores del tipo lógico y se basan en valores True(cierto) o False(falso).

- Los Números:

- El intérprete puede utilizarse como una simple calculadora; puedes introducir una expresión y este escribirá los valores. La sintaxis es sencilla: los operadores `+`, `-`, `*`, y `/` funcionan como en la mayoría de los lenguajes (por ejemplo, Pascal o C); los paréntesis (`()`) pueden ser usados para agrupar. Por ejemplo:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5
>>> 8 / 5 # la división siempre devuelve un número de punto flotante
1.6
>>>
```

- Los números son tipos clásicos para aquellos acostumbrados a los lenguajes de programación. Encontramos enteros, longs, reales y complejos.
- El tipado en Python es dinámico, no se requiere definir el tipo de variable, se asigna solo.
- Podemos comprobar esto mediante la función `type()`.
- Para probar esto podemos entrar en el interprete interactivo.

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> nb_entero = 1
>>> type(nb_entero)
<type 'int'>
>>>
>>> nb_long = 3000000000000000
>>> type(nb_long)
<type 'long'>
>>>
>>> nb_real = 3.14
>>> type(nb_real)
<type 'float'>
>>>
>>> nb_complejo = 3+2j
>>> type(nb_complejo)
<type 'complex'>
>>>
```

- Números Complejos:

- Python puede hacer cálculos con número complejos. La parte imaginaria se acompaña de la letra "j".

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1j + 2 + 3j
(3+4j)
>>> (1+1j) + 1j
(1+2j)
```

- La letra "j" debe ir acompañada siempre de un número y unida a él.

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + j
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'j' is not defined
>>> 1 + 1 j
  File "<stdin>", line 1
    1 + 1 j
        ^
SyntaxError: invalid syntax
>>> 1 + 1j
(1+1j)
>>>
```

- Las cuatro operación básicas:

- Las cuatro operaciones aritméticas básicas son la suma (+), la resta (-), la multiplicación (*) y la división (/).
- Al hacer operaciones en las que intervienen números enteros y decimales, el resultado es siempre decimal. En el caso de que el resultado no tenga parte decimal, Python escribe 0 como parte decimal para indicar que el resultado es un número decimal:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 4.5 * 3
13.5
>>>
>>> 4.5 * 2
9.0
>>>
```


- Al sumar, restar o multiplicar números enteros, el resultado es entero.

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 2
3
>>>
>>> 3 - 4
-1
>>>
>>> 5 * 6
30
>>>
```

- Al dividir números enteros, el resultado es siempre decimal, aunque sea un número entero. Cuando Python escribe un número decimal, lo escribe siempre con parte decimal, aunque sea nula.

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 9 / 2.0
4.5
>>>
>>> 9 / 3.0
3.0
>>>
```

- Cadenas de Textos:

- Además de números, Python puede manipular cadenas de texto, las cuales pueden ser expresadas de distintas formas. Pueden estar encerradas en comillas simples ('...') o dobles ("...").
- Una cadena de caracteres es una lista ordenada de caracteres. Como hemos dicho antes, es inmutable y de tipo secuencia.
- A continuación en la siguiente ejemplo nos muestra el tipo de dato al que pertenece nuestra cadena de texto:

```
strings.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  saludo = "Hola como estas....!!!" #Creamos nuestra variable llamado saludo
5  print(type(saludo)) #Utilizamos la funcion type, para que nos indique si nuestra cadena es tipo String
6
```

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python strings.py
<type 'str'>
```

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>
```

- Ahora si queremos utilizar el interprete interactivo de python para identificar el tipo de dato al que pertenece nuestra cadena de texto, ejecutaremos lo siguiente:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> saludo = "Hola como estas....!!!"
>>> print(type(saludo))
<type 'str'>
>>>
```

- Las triples comillas simples (' ' '...' ' ') y las triples comillas dobles ("""...""") se emplean para cadenas multilineas:

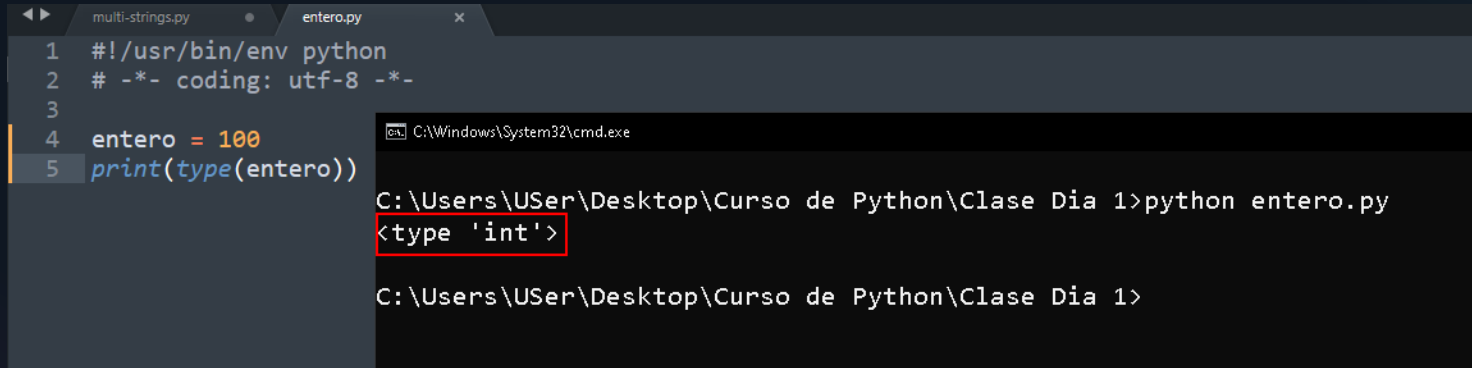
```
multi-strings.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  saludo = '''Hola como estas
5             cuando nos podemos reunir?
6             necesitamos cerrar el proyecto'''
7
8  print(type(saludo))
9
```

- Enteros:

- Los números enteros son aquellos que no tienen decimales, tanto positivos como negativos (además del cero). En Python se pueden representar mediante el tipo int (de integer, entero) o el tipo long (largo). La única diferencia es que el tipo long permite almacenar números más grandes. Es aconsejable no utilizar el tipo long a menos que sea necesario, para no malgastar memoria.
- A continuación, en la siguiente imagen vamos a representar su uso mediante el interprete o mediante el editor de código Sublime Text.

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> entero = 100
>>> print(type(entero))
<type 'int'>
>>>
```

- Utilizando el editor de código Sublime Text, podemos validar el tipo de dato al que pertenece nuestra variable entero.



```
multi-strings.py  entero.py x
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 entero = 100
5 print(type(entero))

C:\Windows\System32\cmd.exe
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python entero.py
<type 'int'>
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>
```

- Entero Long:

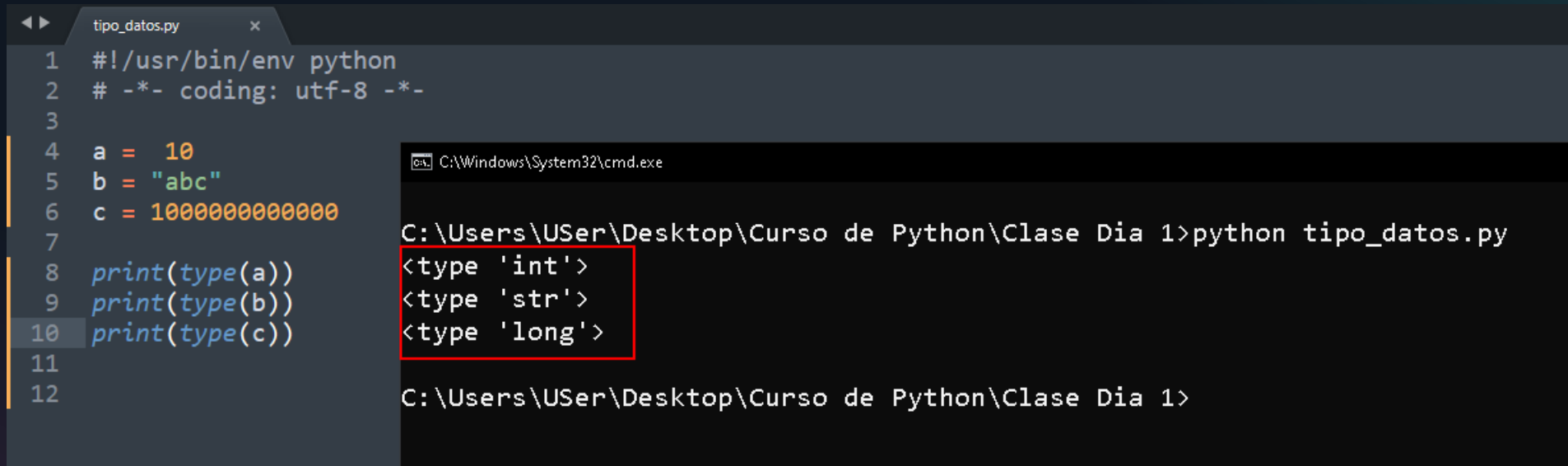
- El tipo long de Python permite almacenar números de cualquier precisión, limitado por la memoria disponible en la máquina.
- Al asignar un número a una variable esta pasará a tener tipo int, a menos que el número sea tan grande como para requerir el uso del tipo long.

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> entero = 10
>>> type(entero)
<type 'int'>
>>>
```


- También puede indicar a Python que un número se almacene usando long añadiendo una L al final:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> entero = 23L
>>> type(entero)
<type 'long'>
>>>
>>> entero = 10000000000000000000L
>>> type(entero)
<type 'long'>
>>>
```

- Ahora si queremos verificar los tipos de datos de los int, str y long en un solo archivo, procederemos ejecutar el siguiente código en nuestro editor de código Sublime Text.



The image shows a Sublime Text editor window with a file named `tipo_datos.py`. The code in the editor is as follows:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 a = 10
5 b = "abc"
6 c = 1000000000000000
7
8 print(type(a))
9 print(type(b))
10 print(type(c))
11
12
```

To the right of the editor is a terminal window titled `C:\Windows\System32\cmd.exe`. It shows the command `python tipo_datos.py` being executed. The output of the script is displayed in the terminal:

```
C:\Users\USER\Desktop\Curso de Python\Clase Dia 1>python tipo_datos.py
<type 'int'>
<type 'str'>
<type 'long'>
```

The output lines are highlighted with a red box in the terminal window. Below the output, the terminal shows the prompt `C:\Users\USER\Desktop\Curso de Python\Clase Dia 1>`.

- Números Flotantes:

- Los números reales son los que tienen decimales. En Python se expresan mediante el tipo float. En otros lenguajes de programación, como C, tiene también el tipo double, similar a float pero de mayor precisión (double = doble precisión).

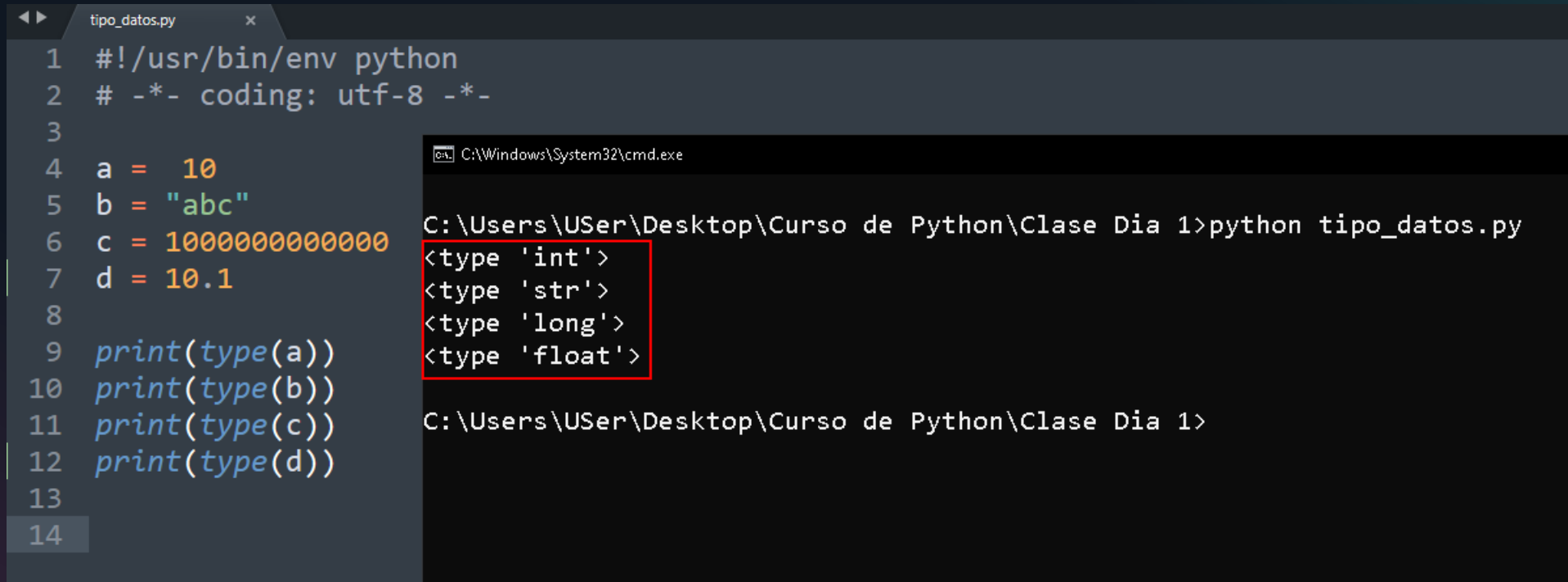
```
flotante.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  flotante = 10.5
5  print(type(flotante))

C:\Windows\System32\cmd.exe

C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python flotante.py
<type 'float'>

C:\Users\User\Desktop\Curso de Python\Clase Dia 1>
```

- Ahora si queremos verificar los tipos de datos (int, str, long y flout) en un solo archivo, procederemos ejecutar el siguiente código en nuestro editor de código Sublime Text.



The image shows a Sublime Text editor window with a file named `tipo_datos.py`. The code in the editor is as follows:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 a = 10
5 b = "abc"
6 c = 1000000000000000
7 d = 10.1
8
9 print(type(a))
10 print(type(b))
11 print(type(c))
12 print(type(d))
13
14
```

Below the editor, a terminal window titled `C:\Windows\System32\cmd.exe` shows the execution of the script:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python tipo_datos.py
<type 'int'>
<type 'str'>
<type 'long'>
<type 'float'>

C:\Users\User\Desktop\Curso de Python\Clase Dia 1>
```

The output of the script is displayed in the terminal, with the first four lines (`<type 'int'>`, `<type 'str'>`, `<type 'long'>`, and `<type 'float'>`) highlighted by a red rectangle.

- Las Listas:

- Una lista es una lista ordenada de elementos, de tipo secuencia y es mutable.
- La lista vacía se define con corchetes: `mi_lista = [...]`.
- Las listas pueden contener ítems de diferentes tipos, pero usualmente los ítems son del mismo tipo.

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> lista = [1,4,9,16,25]
>>> lista
[1, 4, 9, 16, 25]
>>>
```


- Una lista también puede llevar entre [...] datos tipos string.

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> lista_string = ["Hola como estas....:")
>>> print(type(lista_string))
<type 'list'>
>>>
```

- Una lista también puede contener cualquier tipo de elementos (str, int, float y etc).

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> mi_lista = [1,'dos',3,'cuatro',5,6,7]
>>> mi_lista2 = [8.0,'nueve',10,11,12]
>>> print mi_lista, type(mi_lista)
[1, 'dos', 3, 'cuatro', 5, 6, 7] <type 'list'>
>>>
```

- Una lista es un objeto que dispone de muchos métodos. La diferencia con la tupla es que podemos insertar elementos en la lista así como eliminarlos.
- Contamos con varias soluciones para añadir un elemento. Todo depende del sitio donde lo queremos añadir.
- Para añadirlo al final de la lista, utilizaremos el método `append()`.

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> mi_lista = [1,'dos',3,4,5]
>>> mi_lista.append('seis')
>>> print mi_lista
[1, 'dos', 3, 4, 5, 'seis']
>>>
```

- Si deseamos insertar un elemento en un lugar de la lista, utilizaremos el método insert().

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> mi_lista = [1,'dos',3,4,5,'seis']
>>> mi_lista.insert(3,'7.1')
>>> print mi_lista
[1, 'dos', 3, '7.1', 4, 5, 'seis']
>>>
```

- La modificación de un elemento se realiza mediante la asignación.

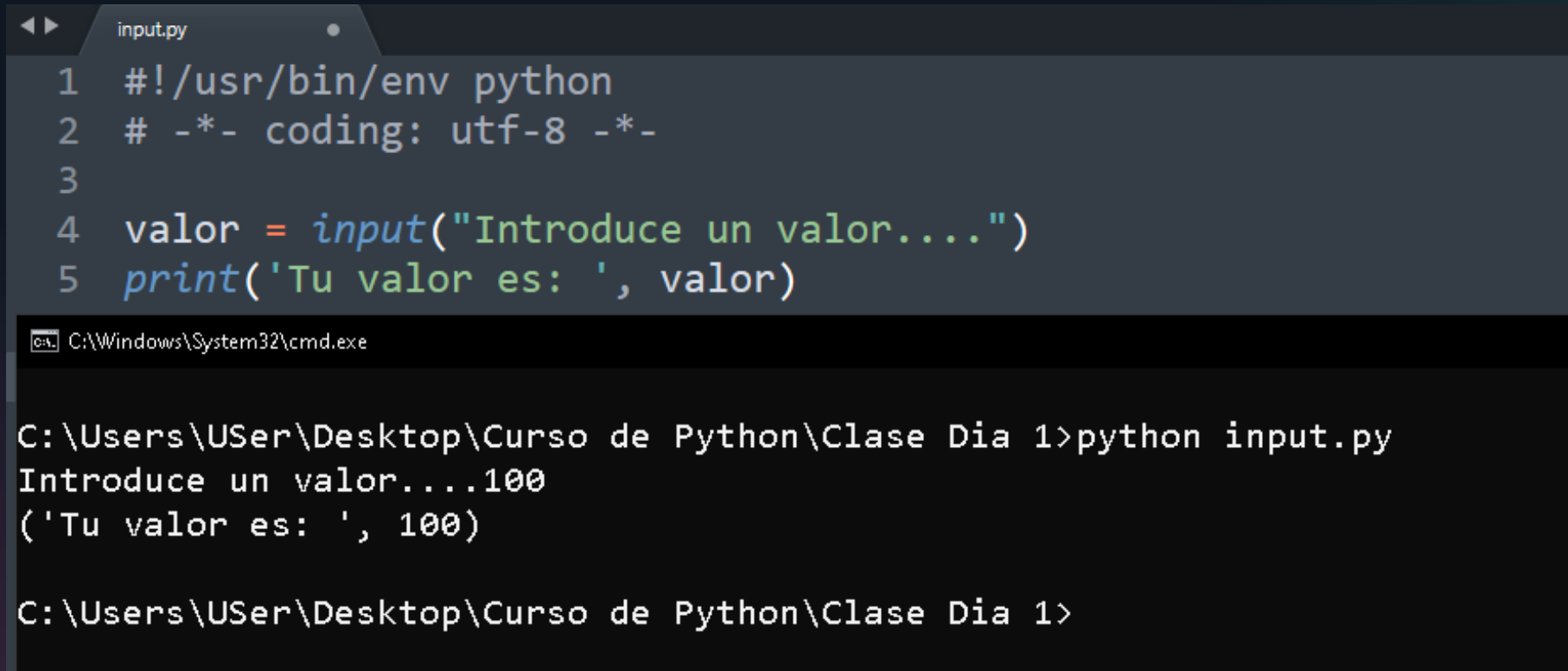
```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> mi_lista = [1,'dos',3,4,5,'seis']
>>> mi_lista[4]='cuatro y medio'
>>> print mi_lista
[1, 'dos', 3, 4, 'cuatro y medio', 'seis']
>>>
```

- Ahora si deseamos eliminar un elemento de la lista podemos eliminarlo mediante el método del().

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> mi_lista = [1,'dos',3,4,5,'seis']
>>> mi_lista[4]='cuatro y medio'
>>> print mi_lista
[1, 'dos', 3, 4, 'cuatro y medio', 'seis']
>>>
>>> del(mi_lista[4])
>>> print mi_lista
[1, 'dos', 3, 4, 'seis']
>>>
```

- Lectura por Teclado:

- Una forma de trabajar con datos dinámicos es a través del teclado con la instrucción `input()` que lee y devuelve una cadena:



```
input.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  valor = input("Introduce un valor....")
5  print('Tu valor es: ', valor)
```

```
C:\Windows\System32\cmd.exe

C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python input.py
Introduce un valor....100
('Tu valor es: ', 100)

C:\Users\User\Desktop\Curso de Python\Clase Dia 1>
```

- Ahora si queremos interpretar la instrucción `input ()`, mediante el interprete de python, podemos ejecutar lo siguiente:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python input.py
Introduce un valor....100
('Tu valor es: ', 100)

C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> valor = input("Introduce un valor...")
Introduce un valor...100
>>> print(valor)
100
>>> valor
100
>>>
```


- Una cadena y un número no se pueden operar, dará error:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> valor = input("Introduce un número entero: ")
Introduce un número entero: valor + 50
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'valor' is not defined
>>>
```

- Tenemos que utilizar la función `int()` para transformar una variable cadena a entero:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> valor = int(input("Introduce un número entero: "))
Introduce un número entero: 100
>>> valor * 2
200
>>>
```

- También tenemos la función `float()` que hace lo propio pero transformando la cadena a flotante:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> valor = float(input("Introduce un número entero o flotante: "))
Introduce un número entero o flotante: 120
>>> valor * 2
240.0
>>>
```

- Ejercicios > Lectura por Teclado:

- Ejercicio 01:

- Escribí un programa que solicite al usuario que ingrese su nombre. El nombre se debe almacenar en una variable llamada nombre. A continuación se debe mostrar en pantalla el texto "Ahora estás en la matrix, [usuario]", donde "[usuario]" se reemplazará por el nombre que el usuario haya ingresado.

- Ejercicio 02:

- Escribí un programa que solicite al usuario ingresar un número con decimales y almacenarlo en una variable. A continuación, el programa debe solicitar al usuario que ingrese un número entero y guardarlo en otra variable. En una tercera variable se deberá guardar el resultado de la suma de los dos números ingresados por el usuario. Por último, se debe mostrar en pantalla el texto "El resultado de la suma es [suma]", donde "[suma]" se reemplazará por el resultado de la operación.

- **Ejercicio 03:**

- Escribí un programa que solicite al usuario dos números y los almacene en dos variables. En otra variable, almacena el resultado de la suma de esos dos números y luego muestra ese resultado en pantalla.
- A continuación, el programa debe solicitar al usuario que ingrese un tercer número, el cual se debe almacenar en una nueva variable. Por último, muestra en pantalla el resultado de la multiplicación de este nuevo número por el resultado de la suma anterior.

- **Ejercicio 04:**

- Escribí un programa que solicite al usuario ingresar la cantidad de kilómetros recorridos por una motocicleta y la cantidad de litros de combustible que consumió durante ese recorrido. Mostrar el consumo de combustible por kilómetro.

- **Ejercicio 05:**

- Escribí un programa que solicite al usuario ingresar tres números para luego mostrarle el promedio de los tres.

- **Ejercicio 06:**

- Escribí un programa que solicite al usuario el ingreso de dos palabras, las cuales se guardarán en dos variables distintas. A continuación, almacena en una variable la concatenación de la primera palabra, más un espacio, más la segunda palabra. Muestra este resultado en pantalla.

- Solución > Ejercicio 01:

```
ejercicio01.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #Solucion
5
6  nombre = input("Tu nombre: ")
7  print("Ahora estás en la matrix,", nombre)
8
```

- Resultado:

```
Ejercicios Practicos\Ejercicios>py ejercicio01.py
Tu nombre: pedro
Ahora estás en la matrix, pedro

C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\
Ejercicios Practicos\Ejercicios>
```


- Solución > Ejercicio 02:

```
ejercicio02.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #Solucion
5
6  a=float(input("Primer numero: "))
7  b=int(input("Segundo numero: "))
8  c=a+b
9  print("El resultado de la suma es", c)
10
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Ejercicios Practicos\Ejercicios>python ejercicio02.py
Primer numero: 10.2
Segundo numero: 4
('El resultado de la suma es', 14.2)
```

- Solución > Ejercicio 03:

```
ejercicio03.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #Solucion
5
6  n1=int(input("Ingresa un numero: "))
7  n2=int(input("Ingresa otro numero: "))
8  suma=n1+n2
9  print("Suman: ", suma)
10 n3=int(input("Ingresa un nuevo numero: "))
11 print("Multiplicacion de la suma por el ultimo numero: ",suma*n3)
12
```

- Resultado:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1_Curso Python\Ejercicios Practicos\Ejercicios>python ejercicio03.py
Ingresa un numero: 10
Ingresa otro numero: 20
('Suman: ', 30)
Ingresa un nuevo numero: 100
('Multiplicacion de la suma por el ultimo numero: ', 3000)
```

- Solución > Ejercicio 04:

```
ejercicio04.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #Solucion
5
6  kilometros=float(input("Kilometros recorridos: "))
7  litros=float(input("Litros de combustible gastados: "))
8  print("El consumo por kilometro es de", kilometros/litros)
9
```

- Resultado:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1_Curso Python\Ejercicios Practicos\Ejercicios>python ejercicio04.py
Kilometros recorridos: 100
Litros de combustible gastados: 20
('El consumo por kilometro es de', 5.0)
```

- Solución > Ejercicio 05:

```
ejercicio05.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #Solucion
5
6  n1=float(input("Primer numero: "))
7  n2=float(input("Segundo numero: "))
8  n3=float(input("Tercer numero: "))
9  print("El promedio de los tres es", (n1+n2+n3)/3)
10
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Ejercicios Practicos\Ejercicios>python ejercicio05.py
Primer numero: 20
Segundo numero: 200
Tercer numero: 10
('El promedio de los tres es', 76.66666666666667)
```

- Solución > Ejercicio 06:

```
ejercicio06.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #Solucion
5
6  palabra1=input("Primera palabra: ")
7  palabra2=input("Segunda palabra: ")
8  frase=palabra1+" "+palabra2
9  print(frase)
10
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Ejercicios Practicos\Ejercicios>py ejercicio06.py
Primera palabra: hola
Segunda palabra: como estas
hola como estas
```

- Operadores y Expresiones:

- Sabemos que la programación gira en torno a la información, en torno a los datos, ¿pero con qué propósito?
- Así que en este tema trataremos las operaciones de los tipos básicos, pero antes introduciremos un nuevo tipo de dato muy especial.

Operadores Relacionales			
Operador	Operación	Ejemplo	Resultado
=	Igual que	'hola' = 'lola'	FALSO
<>	Diferente a	'a' <> 'b'	VERDADERO
<	Menor que	7 < 15	VERDADERO
>	Mayor que	22 > 11	VERDADERO
<=	Menor o igual que	15 <= 22	VERDADERO
>=	Mayor o igual que	35 > = 20	VERDADERO

Relacionales

- == ¿Son iguales?
- != ¿Son distintos?
- < "Menos qué"
- > "Mayor qué"
- <= "Menor o igual a"
- >= "Mayor o igual a"

Lógicos

- and
- or
- not

Estos devuelven un valor lógico

Aritméticos

- + Suma

- - Resta

- * Multiplicación

- ** Exponente

- / División

- // División Entera

- % Modulo o residuo

- Tipos de datos adicionales:

- Existen otros tipos de datos como los booleanos y el tipo set.
- El tipo boolean acepta los valores True (verdaderos) y False (falso). Ya hemos visto este tipo de datos al utilizar el operador in.
- El tipo set es una colección no ordenada de datos únicos, set soporta la unión, la intersección, la diferencia y la simetría.
- A continuación en la siguiente imagen tenemos un ejemplo de operadores relacionales.
- Igual que:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> 3 == 2
False
>>>
```

- Operadores Relacionales:

- Distinto de:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> 3 != 2
True
>>>
```

- Mayor que:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> 3 > 2
True
>>>
```

- Operadores Relacionales:

- Mayor o igual que:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> 3 >= 4
False
>>>
```

- Menor o igual que:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> 3 <= 4
True
>>>
```

- También podemos comparar variables:

```
C:\Users\USeer\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> a = 10
>>> b = 5
>>> a > b
True
>>>
>>> b != a
True
>>>
>>> a == b*2
True
>>>
>>>
```

- Operadores Lógicos (and, or y not):

- Encontramos 3 operadores especiales para realizar operaciones lógicas. Normalmente se utilizan para agrupar, excluir y negar expresiones. Puede ayudar echar un vistazo a [esta explicación](#) sobre las tablas de la verdad:
- Not (Negación lógica).
- Niega un valor o expresión lógica:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> not True
False
>>>
>>> not False
True
>>>
>>> not True == False
True
>>>
>>> True == True
True
>>>
```


- And (Conjunción lógica):

- Devuelve verdadero sólo si se cumplen todas las condiciones:
- El operador and evalúa si el valor a la izquierda y el de la derecha son True, y en el caso de ser cierto, devuelve True.

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print(True and True)
True
>>>
>>> print(True and False)
False
>>>
>>> print(False and True)
False
>>>
>>> print(False and False)
False
>>>
```

- Operador or:

- El operador or devuelve True cuando al menos uno de los elementos es igual a True. Es decir, evalúa si el valor a la izquierda o el de la derecha son True.

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Ejercicios Practicos>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print(True or True)
True
>>>
>>> print(True or False)
True
>>>
>>> print(False or True)
True
>>>
>>> print(False or False)
False
>>>
```

- Operadores Aritméticos:

- Los operadores aritméticos o arithmetic operators son los más comunes que nos podemos encontrar, y nos permiten realizar operaciones aritméticas sencillas, como pueden ser la suma, resta o exponente.

Operador	Nombre	Ejemplo
+	Suma	$x + y = 13$
-	Resta	$x - y = 7$
*	Multiplicación	$x * y = 30$
/	División	$x/y = 3.333$
%	Módulo	$x \% y = 1$
**	Exponente	$x ** y = 1000$
//	Cociente	3

- A continuación, mostraremos en la siguiente tabla todos ellos con un ejemplo, donde $x=10$ y $y=3$.

<pre>operador_aritmetico.py x 1 #!/usr/bin/env python 2 # -*- coding: utf-8 -*- 3 4 x = 10; y = 3 5 print("Operadores aritmeticos") 6 print("x+y =", x+y) 7 print("x-y =", x-y) 8 print("x*y =", x*y) 9 print("x/y =", x/y) 10 print("x%y =", x%y) 11 print("x**y =", x**y) 12 print("x//y =", x//y)</pre>	<pre>C:\Windows\System32\cmd.exe C:\Users\User\Desktop\Curso de Python\Clase Dia 1>python operador_aritmetico.py Operadores aritmeticos ('x+y =', 13) ('x-y =', 7) ('x*y =', 30) ('x/y =', 3) ('x%y =', 1) ('x**y =', 1000) ('x//y =', 3) C:\Users\User\Desktop\Curso de Python\Clase Dia 1></pre>
--	--

- Ejercicios > Operadores y Expresiones:

- Ejercicio 1:
 - Realiza un programa que lea 2 números por teclado y determine los siguientes aspectos (es suficiente con mostrar True o False):
 - Si los dos números son iguales.
 - Si los dos números son diferentes.
 - Si el primero es mayor que el segundo.
 - Si el segundo es mayor o igual que el primero.
- Ejercicio 2:
 - Utilizando operadores lógicos, determina si una cadena de texto introducida por el usuario tiene una longitud mayor o igual que 3 y a su vez es menor que 10 (es suficiente con mostrar True o False).

- Ejercicio 3:
- Realiza un programa que cumpla el siguiente algoritmo utilizando siempre que sea posible operadores en asignación:
- Guarda en una variable `numero_magico` el valor 12345679 (sin el 8).
- Lee por pantalla otro `numero_usuario`, especifica que sea entre 1 y 9.
- Multiplica el `numero_usuario` por 9 en sí mismo.
- Multiplica el `numero_magico` por el `numero_usuario` en sí mismo.
- Finalmente muestra el valor final del `numero_magico` por pantalla.

- Solución > Ejercicio 1:

```
solucion1.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  n1 = float(input("Introduce el primer número: "))
5  n2 = float(input("Introduce el segundo número: "))
6
7  print("Son iguales: ", n1 == n2)
8  print("Son diferentes: ", n1 != n2)
9  print("El primero es mayor que el segundo: ", n1 > n2)
10 print("El segundo es mayor o igual que el primero: ", n1 <= n2)
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Operadores y Expresiones>python solucion1.py
Introduce el primer número: 10
Introduce el segundo número: 10
('Son iguales: ', True)
('Son diferentes: ', False)
('El primero es mayor que el segundo: ', False)
('El segundo es mayor o igual que el primero: ', True)

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Operadores y Expresiones>
```


- Solución > Ejercicio 2:

```
solucion2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  cadena = input("Escribe una cadena: ")
5  print("La longitud de la cadena es mayor o igual que 3 y menor que 10: ",
6        len(cadena) >= 3 and len(cadena) < 10 )
7
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Operadores y Expresiones>py solucion2.py
Escribe una cadena: hola
La longitud de la cadena es mayor o igual que 3 y menor que 10:  True

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Operadores y Expresiones>
```

- Solución > Ejercicio 3:

```
solucion3.py x
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 numero_magico = 12345679
5 numero_usuario = int(input("Introduce un número del 1 al 9: "))
6 numero_usuario *= 9
7 numero_magico *= numero_usuario
8 print("El número mágico es: ", numero_magico)
9
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Operadores y Expresiones>python solucion3.py
Introduce un número del 1 al 9: 5
('El número mágico es: ', 555555555)

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Operadores y Expresiones>
```

- Controladores de Flujo:

- Si un programa sólo ejecutara instrucciones planas unas tras otras no servirían de mucho. Por suerte ahí es donde aparecen las sentencias de control.
- Python, como el resto de los lenguajes de programación, tenemos varios modos de controlar el flujo del programa.
- En Python tenemos tres modos elementos para controlar el flujo:
- La declaración **if**, que ejecuta un bloque particular de comandos en función del resultado de un test.
- La declaración **while**, que ejecuta un bloque de comandos mientras que se cumpla un test determinado.
- La declaración **for loop**, que ejecuta un bloque de comandos un cierto número de veces.
- En Python es "obligatorio" definir una estructura de control del siguiente modo:

Sentencias

Las **sentencias** son los elementos básicos en los que se divide el código en un lenguaje de **programación**.

- Sentencias "if", "elif" y "else":
 - Estas estructuras de control nos sirven para ejecutar una porción de código cuando se cumpla un o varias determinadas condiciones

"if" (si):

Nos ayuda a iniciar el proceso de condición y es el condicional básico

"elif" (sino si):

Podemos usarlo cuantas veces queramos y nos sirve para validar ciertas condiciones, va de la mano con "if"

"else" (sino):

Con él, ejecutamos por último un trozo de código si ninguna condición fue capaz de cumplirse

- La instrucción if ... elif ... else nos permitirá verificar condiciones y, en funciones del resultado, efectuar las acciones asociadas a este.

```
if <condicion>  
    #ejecucion si la condicion es verdadera.  
elif <otra condicion>  
    #ejecucion si otra condicion es verdadera.  
else;  
    #ejecucion si la condicion es falsa.
```

- If puede utilizarse solo, con else o con elif y else.

Bucles o ciclos

- Un bucle o ciclo, en programación, es una sentencia que ejecuta repetidas veces un trozo de código, hasta que la condición asignada a dicho bucle deja de cumplirse.
 - while
 - for

- El bucle while permite repetir una operación mientras que la condición sea verdadera.

```
while <condicion>
    #ejecutar si la condicion es verdadera.
```

- Para salir de un bucle while, podemos utilizar la instrucción break. Si queremos saltar una iteración, emplearemos la instrucción continue.
- A continuación, se presenta un ejemplo del uso del bucle while controlado por conteo:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 21:01:17) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> suma, numero = 0, 1
>>> while numero <= 10:
...     suma = numero + suma
...     numero = numero + 1
...
>>> print "La suma es: " + str(suma)
La suma es: 55
>>>
```

- A continuación, se presenta un ejemplo del uso del bucle while NO controlado por conteo:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # while = bucle
5
6  numero = int(raw_input("Numero: "))
7  mensaje = raw_input("Mensaje: ")
8
9  i = 0 #Como nuestro iterador es igual 0
10
11 while i <= numero:
12     print(mensaje)
```

C:\Windows\System32\cmd.exe

[illegible]

- Para corregir este error vamos a decirle a $i=i+1$, para que de esta manera siempre que while se ejecute al iterador i que es igual 0 se le va sumar 1

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # while = bucle
5
6  numero = int(raw_input("Numero: "))
7  mensaje = raw_input("Mensaje: ")
8
9  i = 0
10
11 while i <= numero:
12     print(mensaje)
13     i=i+1
```

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python while_no_conteo.py  
Numero: 10  
Mensaje: hola  
hola  
hola  
hola  
hola  
hola  
hola  
hola  
hola  
hola  
hola  
hola  
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>
```

- Las Sentencias if, elif y else:

- Tal vez el tipo más conocido de sentencia sea el if. Por ejemplo:

```
if.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  numero = int(raw_input("Ingresa un número entero, por favor: "))
5
6  if numero < 0:
7      numero = 0
8      print 'El número ingresado es negativo cambiado a cero.'
9  elif numero == 0:
10     print 'El número ingresado es 0.'
11 elif numero == 1:
12     print 'El número ingresado es 1.'
13 else:
14     print 'El número ingresado es mayor que uno.'
15
```

- En la siguiente imagen tenemos el resultado de nuestro programa:

```
C:\Windows\System32\cmd.exe

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python if.py
Ingresa un número entero, por favor: 10
El número ingresado es mayor que uno.

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python if.py
Ingresa un número entero, por favor: 1
El número ingresado es 1.

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python if.py
Ingresa un número entero, por favor: -10
El número ingresado es negativo cambiado a cero.

C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>
```

- En el ejemplo anterior usa dos funciones integradas en el interprete Python:
- La función `int()` que convierte el valor ingresado a número entero.
- La función `raw_input()` que lee el valor ingresado por la entrada estándar.

- La Sentencia else:

- Se encadena a un If para comprobar el caso contrario (en el que no se cumple la condición):

```
else.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  n = int(raw_input("Ingresa un número par, por favor: "))
5
6  if n % 2 == 0:
7      print("es un número par")
8  else:
9      print("es un número impar")
10
```

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python else.py
Ingresa un número par, por favor: 10
es un número par
```

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>python else.py
Ingresa un número par, por favor: 7
es un número impar
```

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo>
```

- Ejercicios > Controles de Flujo:

- Ejercicio 1:
 - Realiza un programa que lea dos números por teclado y permita elegir entre 3 opciones en un menú:
 - Mostrar una suma de los dos números.
 - Mostrar una resta de los dos números (el primero menos el segundo).
 - Mostrar una multiplicación de los dos números.
- Ejercicio 2:
 - Realiza un programa que lea un número impar por teclado. Si el usuario no introduce un número impar, debe repetirse el proceso hasta que lo introduzca correctamente.

- Ejercicio 3:
 - Realiza un programa que sume todos los números enteros pares desde el 0 hasta el 100.
- Ejercicio 4:
 - Realiza un programa que pida al usuario cuantos números quiere introducir. Luego lee todos los números y realiza una media aritmética.

- Solución > Ejercicio 1:

```
solucion1.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  n1 = float(input("Introduce un numero: "))
5  n2 = float(input("Introduce otro numero: "))
6  opcion = 0
7
8  print("\nQue quieres hacer\n \n1) Sumar los dos numeros \n2) Restar los dos numeros \n3) Multiplicar los dos numeros")
9  opcion = int(input("Introduce un numero: "))
10
11 if opcion == 1:
12     print("La suma de",n1,"+",n2,"es",n1+n2)
13 elif opcion == 2:
14     print("La resta de",n1,"-",n2,"es",n1-n2)
15 elif opcion == 3:
16     print("El producto de",n1,"*",n2,"es",n1*n2)
17 else:
18     print("Opcion incorrecta")
19
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Controladores de flujo\Ejercicios y Soluciones>python solucion1.py
Introduce un numero: 10
Introduce otro numero: 15

Que quieres hacer

1) Sumar los dos numeros
2) Restar los dos numeros
3) Multiplicar los dos numeros
Introduce un numero: 3
('El producto de', 10.0, '*', 15.0, 'es', 150.0)
```

- Solución > Ejercicio 2:

```
solucion2.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  numero = 0
5  while numero % 2 == 0: # Mientras sea par repetimos el proceso
6      numero = int(input("Introduce un numero impar: "))
7
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Controla
dores de flujo\Ejercicios y Soluciones>python solucion2.py
Introduce un numero impar: 4
Introduce un numero impar: 2
Introduce un numero impar: 3

C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Controla
dores de flujo\Ejercicios y Soluciones>python solucion2.py
Introduce un numero impar: 5

C:\Users\User\Desktop\Curso de Python\Clase Dia 1_Curso Python\Controla
dores de flujo\Ejercicios y Soluciones>python solucion2.py
Introduce un numero impar: 2
Introduce un numero impar: 7
```

- Solución > Ejercicio 3:

```
solucion3.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # Primera forma con función sum()
5  suma = sum( range(0, 101, 2))
6  print(suma)
7
8  # Segunda forma con bucles
9  num = 0
10 suma = 0
11
12 while num <= 100:
13     if num % 2 == 0:
14         suma += num
15     num += 1
16
17 print(suma)
18
```

- Resultado:

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo\Ejercicios y Soluciones>python solucion3.py
2550
2550
```

```
C:\Users\User\Desktop\Curso de Python\Clase Dia 1\Controladores de flujo\Ejercicios y Soluciones>
```

- Solución > Ejercicio 4:

```
solucion4.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 suma = 0
5 numeros = int(input("Cuantos numeros quieres introducir: "))
6 for x in range(numeros):
7     suma += float(input("Introduce un numero: "))
8 print("Se han introducido", numeros, "numeros que en total han sumado",
9       suma, "y la media es", suma/numeros)
10
```

- Resultado:

```
C:\Users\USer\Desktop\Curso de Python\Clase Dia 1_Curso Python\Controladores de flujo\Ejercicios y Soluciones>python solucion4.py
Cuantos numeros quieres introducir: 4
Introduce un numero: 2
Introduce un numero: 5
Introduce un numero: 3
Introduce un numero: 1
('Se han introducido', 4, 'numeros que en total han sumado', 11.0, 'y la media es', 2.75)
```



¡GRACIAS!

José Antonio Gonzales Jurado

Teléfono:

(+51) 977177907

Correo electrónico:

anton_gonzaj@hotmail.com

Linkedin:

<https://www.linkedin.com/in/jose-antonio-gonzales-jurado//>

