

Programación I

Maestría en Ciencia de Datos

Clase 2

IT PhD Rocío del Carmen Chávez Álvarez

Modificaciones para
que el ejecutable
corra desde
cualquier path

pyinstaller --onefile
"nombre_del_archivo.py"

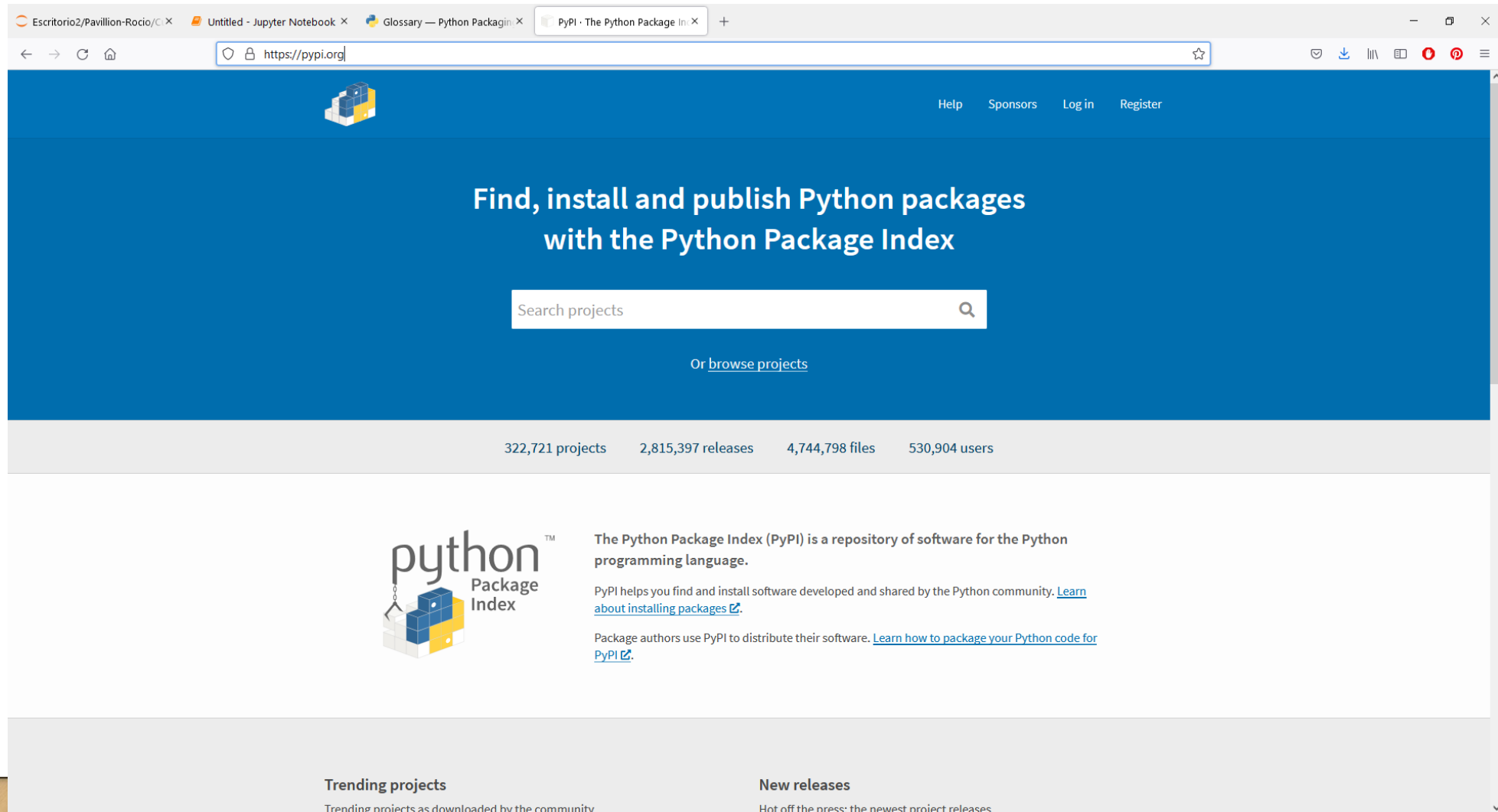
Probar el .exe desde
cualquier ruta

```
C:\Users\rocio\Escritorio2\Pavillion-Rocio\Cursos impartidos Presenciales\Programacion I MCD\Scripts\Histogramas en Python2.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

< > untitled Histogramas en Python2.py x matplotlibrc.template

1 #Indicamos el directorio en el que vamos a trabajar
2 import os
3 import sys
4
5 #Detectar la ruta del ejecutable
6 ruta_aplicacion= os.path.dirname(sys.executable)
7 #Importamos los paquetes que vamos a necesitar
8 import pandas as pd #Contiene funciones que nos ayudan en el análisis de datos
9 import matplotlib.pyplot as plt #Nota: Fue necesario instalar matplotlib 2 con conda inst
10 #y comentar la linea 4 del archivo _classic_test_patch.py
11
12
13 #Leemos el archivo a analizar
14 atletas = pd.read_csv(f'{ruta_aplicacion}/categorias de corredores.csv', index_col=0) #Co
15 atletas.info()
16
17 #Vemos las primeras filas del archivo
18 atletas.head()
19
20 #Creamos el histograma de la variable Tiempo
21 plt.figure(1)
22 plt.hist(atletas['Tiempo'], 15, color="yellow", ec="black")
23 plt.title("Histograma Tiempo")
24
25 plt.savefig(f'{ruta_aplicacion}/Histograma.jpg')
```

Python Package Index



Conociendo Jupyter Notebook



Home

Environments

Learning

Community

Documentation

Applications on

base (root)

Channels

Refresh



JupyterLab

1.1.4

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



Notebook

6.0.1

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



Orange 3

3.21.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Launch



Qt Console

4.5.5

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



C:/users/Rocio/

Home Page - Select or creat. X

localhost:8888/tree

QuitLogout

FilesRunningClusters

Select items to perform actions on them.

UploadNew↺

0

/

Name

Last Modified

File size

<input type="checkbox"/>	3D Objects	hace 3 meses
<input type="checkbox"/>	Anaconda3	hace 4 días
<input type="checkbox"/>	Apple	hace un año
<input type="checkbox"/>	Application Data	hace 10 meses
<input type="checkbox"/>	Archivos en Jupyter	hace 6 meses
<input type="checkbox"/>	Automatizacion	hace 2 días
<input type="checkbox"/>	BD	hace 6 meses
<input type="checkbox"/>	build	hace 4 días
<input type="checkbox"/>	Calibre Library	hace un mes
<input type="checkbox"/>	Carpeta de pruebas	hace 21 días
<input type="checkbox"/>	Contacts	hace 3 meses
<input type="checkbox"/>	Desktop	hace 3 meses
<input type="checkbox"/>	dist	hace 4 días
<input type="checkbox"/>	Documents	hace 9 meses

Escritorio2/Pavillion-Rocio/Cl

localhost:8888/tree/Escritorio2/Pavillion-Rocio/Cursos impartidos Presenciales/Progra

jupyter

QuitLogout

FilesRunningClusters

Select items to perform actions on them.

0

/ Escritorio2 / Pavillion-Rocio / Cursos impartidos Presenciales / Programacion I MCD / Scripts

UploadNew

Notebook:

Python 3 (ipykernel)

Other:

Text File

Folder

Terminal

..		
build		
dist		
categorias de corredores.csv		
Histograma.jpg	hace 5 días	29.6 kB
Histogramas en Python.py	hace 4 días	1.51 kB
Histogramas en Python.spec	hace 4 días	1.32 kB
Histogramas en Python2.py	hace 4 días	1.54 kB
Histogramas en Python2.spec	hace 4 días	1.18 kB
Python en terminal.py	hace 5 días	652 B



In []:

Texto

Escritorio2/Pavillion-Rocio/C

Untitled - Jupyter Notebook

+

←

→

↺

🏠

🛡️📄localhost:8888/notebooks/Escritorio2/Pavillion-Rocio/Cursos impartidos Presenciales/☆

📧

⬇️

≡

📖

🔥

📌

☰

jupyter

Untitled

Last Checkpoint: hace 5 minutos (autosaved)

Python 3 (ipykernel)

🔴

Logout

File

Edit

View

Insert

Cell

Kernel

Widgets

Help

Trusted

New Notebook

Open...

Make a Copy...

Save as...

Rename...

Save and Checkpoint

Ctrl-S

Revert to Checkpoint

Print Preview

Download as

Trusted Notebook

Close and Halt

▶ Run

■

↺

▶▶

Code

▼

🖨️

localhost:8888/notebooks/Escritorio2/Pavillion-Rocio/Cursos impartidos Presenciales/Programacion I MCD/Scripts/Untitled.ipynb?kernel_name=python3#



Cut Cells X

Copy Cells C

Paste Cells Above Shift-V

Paste Cells Below V

Paste Cells & Replace

Delete Cells D, D

Undo Delete Cells Z

Split Cell Ctrl-Shift-Minus

Merge Cell Above

Merge Cell Below

Move Cell Up

Move Cell Down

Edit Notebook Metadata

Find and Replace

Cut Cell Attachments

Copy Cell Attachments

Paste Cell Attachments

Insert Image



Code v





Toggle Header

Toggle Toolbar

Toggle Line Numbers Shift-L

Cell Toolbar ▶



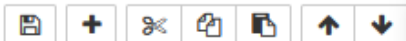
Code



In []:



In []:



In []:

Run Cells Ctrl-Enter

Run Cells and Select Below Shift-Enter

Run Cells and Insert Below Alt-Enter

Run All

Run All Above

Run All Below

Cell Type

Current Outputs

All Output



Escritorio2/Pavillion-Rocio/Cl



Untitled - Jupyter Notebook



localhost:8888/notebooks/Escritorio2/Pavillion-Rocio/Cursos impartidos Presenciales/



jupyter

Untitled

Last Checkpoint: hace 8 minutos (autosaved)



Logout

File

Edit

View

Insert

Cell

Kernel

Widgets

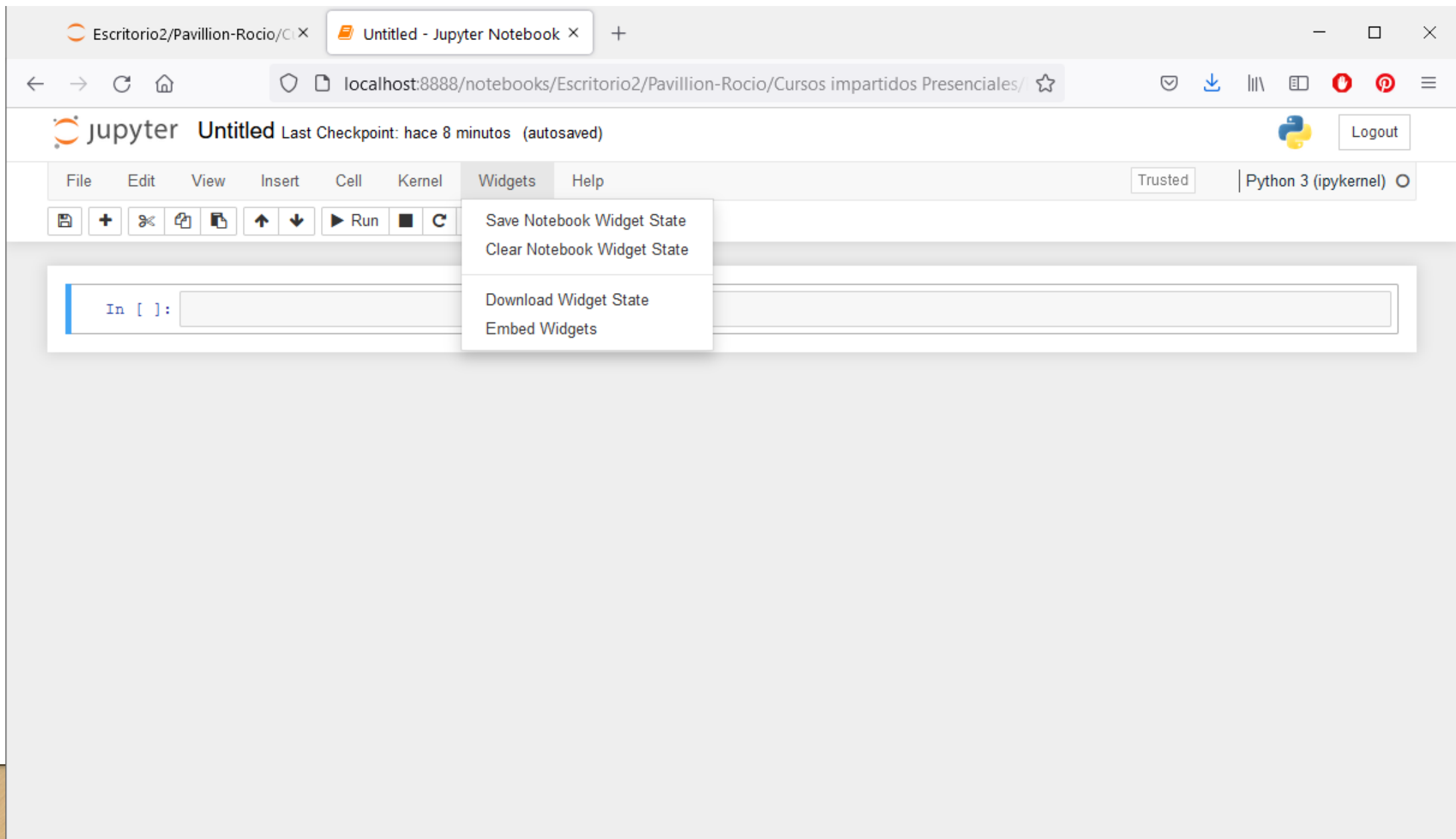
Help

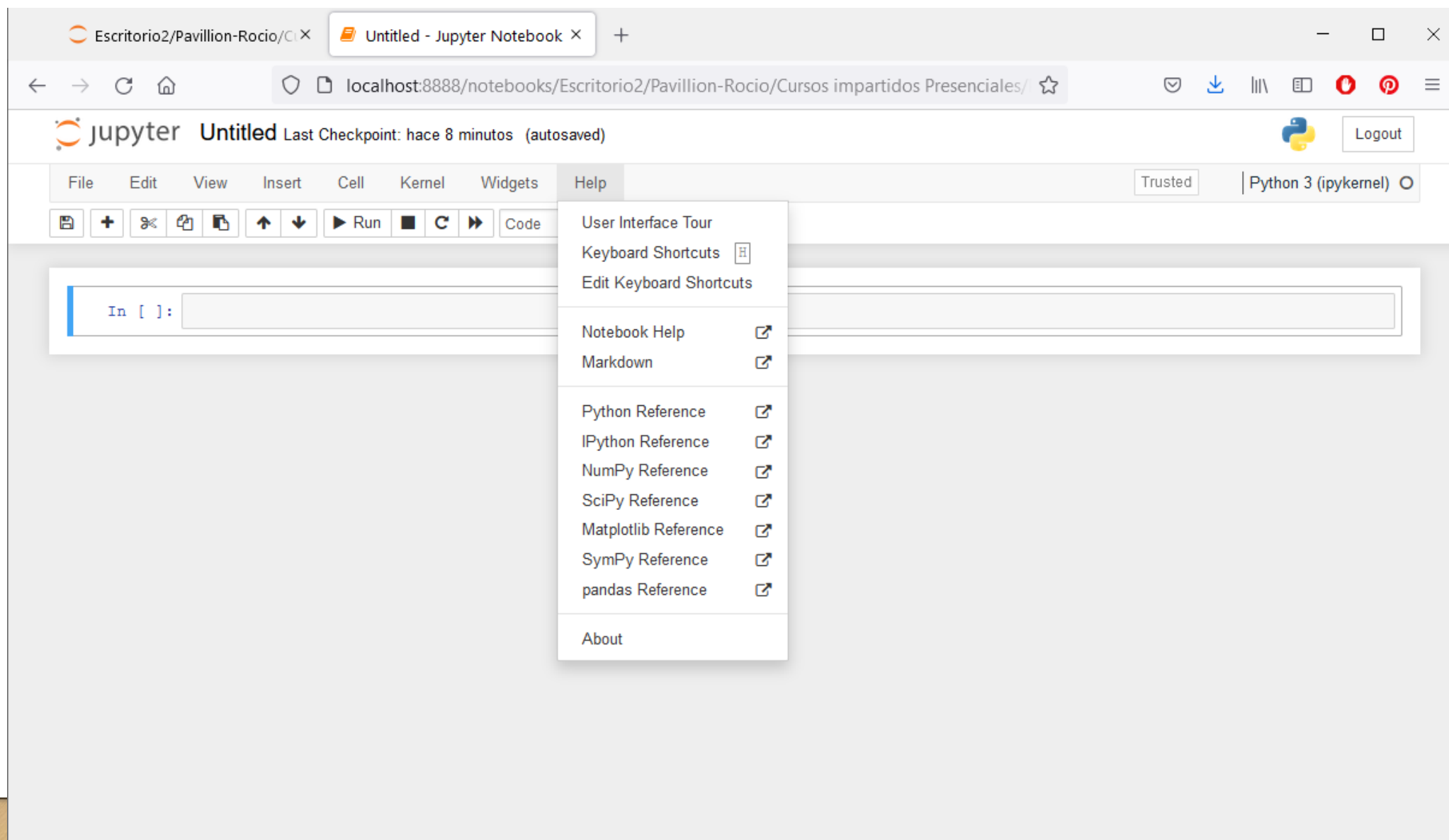
Trusted

Python 3 (ipykernel)

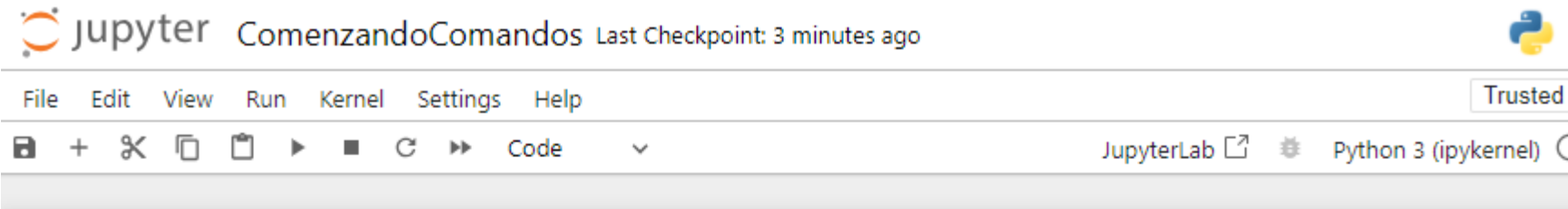


In []:





Comenzamos con Algunos Comandos



Conociendo la versión de python

```
[1]: import sys
      !{sys.executable} --version
```

Python 3.11.9

Conociendo la versión de una librería

```
[2]: import pandas
      pandas.__version__
```

[2]: '2.2.2'

[]:



*Copiamos el código del archivo
Histogramas en Python.py”*

Jupyter Untitled Last Checkpoint: hace una hora (unsaved changes) Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Script Histogramas en Python2.py

```
In [24]: import os
import sys

#Detectar la ruta del ejecutable
ruta_aplicacion= os.path.dirname(sys.executable)
print(ruta_aplicacion)

#Importamos los paquetes que vamos a necesitar
import pandas as pd #Contiene funciones que nos ayudan en el análisis de datos
import matplotlib.pyplot as plt #Nota: Fue necesario instalar matplotlib 2 con conda install matplotlib=2
#y comentar la linea 4 del archivo _classic_test_patch.mplstyle

#Leemos el archivo a analizar
atletas = pd.read_csv(f'{ruta_aplicacion}/categorias de corredores.csv', index_col=0) #Con index_col=0 le indicamos que las f
atletas.info()

#Vemos las primeras filas del archivo
atletas.head()
```

C:\Users\rocio\Anaconda3
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 3475 to 8527
Data columns (total 6 columns):
Column Non-Null Count Dtype

0 Lugar 1000 non-null int64
1 Genero 1000 non-null object
2 Edad 1000 non-null int64
3 Pais 999 non-null object
4 Tiempo 1000 non-null float64
5 Velocidad 1000 non-null object
dtypes: float64(1), int64(2), object(3)
memory usage: 54.7+ KB

Out[24]:

	Lugar	Genero	Edad	Pais	Tiempo	Velocidad
Corredor						
3475	3592	Male	52	GBR	217.483333	Regular
13594	13853	Female	40	NY	272.550000	Regular
12012	12256	Male	31	FRA	265.283333	Regular

*Copiamos el código del archivo
Histogramas en Python.py"*

ook × Glossary — Python Packagin × ¿Qué es el Kernel y para qué × +

888/notebooks/Escritorio2/Pavillion-Rocio/Cursos impartidos Presenciales/Programacion I MCD/Scripts/Untitled.ipynb?kernel_na

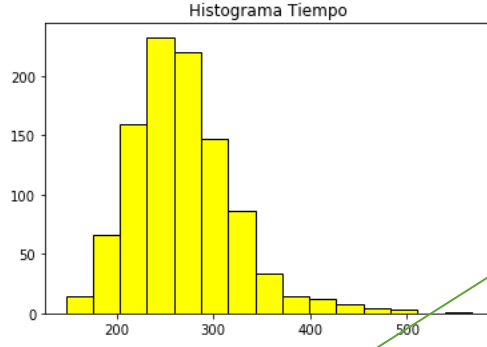
jupyter Untitled Last Checkpoint: hace una hora (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run

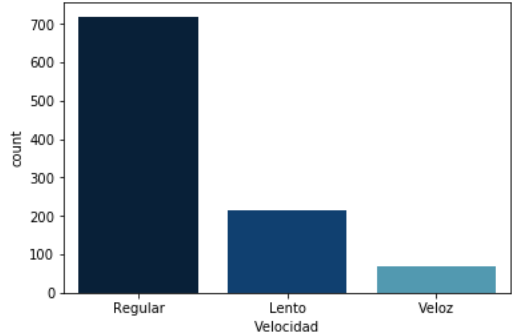
```
In [25]: #Creamos el histograma de la variable Tiempo
plt.figure(1)
plt.hist(atletas['Tiempo'], 15, color="yellow", ec="black")
plt.title("Histograma Tiempo")
```

Out[25]: Text(0.5,1,'Histograma Tiempo')



Si dividimos el código de esta manera, no es necesaria la función plt.figure() para obtener los gráficos sin que estos se traslapen, pero si estamos

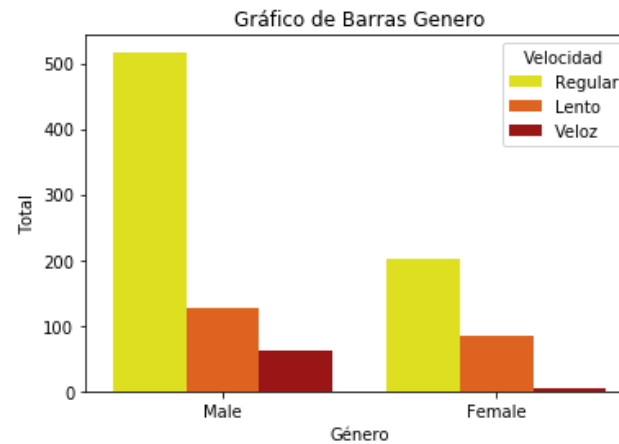
```
In [26]: #Para conocer la frecuencia de una variable que es categórica
import seaborn as sns
plt.figure(2)
sns.countplot(x=atletas['Velocidad'], palette = 'ocean')
plt.savefig(f'{ruta_aplicacion}/Velocidades.jpg')
```



```
In [27]: #Si queremos saber las velocidades en hombres y en mujeres
plt.figure(3)
grafico3 = sns.countplot(x = 'Genero', hue = 'Velocidad', palette = 'hot r', data = atletas)
```

*Copiamos el código del archivo
Histogramas en Python.py"*

```
In [27]: #Si queremos saber las velocidades en hombres y en mujeres
#plt.figure(3)
grafico3 = sns.countplot(x = 'Genero', hue = 'Velocidad', palette = 'hot_r', data = atletas)
grafico3.set(title = 'Velocidades por Género',
             xlabel = 'Género', ylabel = 'Total')
plt.title("Gráfico de Barras Genero")
plt.savefig(f'{ruta_aplicacion}/Genero.jpg')
plt.show()
```



En Jupyter
notebook no es
necesaria la función
plt.show()

Ejecutando un Script Creado en Notebook desde el prompt

```
Anaconda Powershell Prompt
(base) PS C:\Users\rocio\Escritorio2\Pavillion-Rocio\Cursos impartidos Presenciales\Programacion I MCD\scripts> python "Histogramas desde Notebook.ipynb"
Traceback (most recent call last):
  File "Histogramas desde Notebook.ipynb", line 254, in <module>
    "execution_count": null,
NameError: name 'null' is not defined
(base) PS C:\Users\rocio\Escritorio2\Pavillion-Rocio\Cursos impartidos Presenciales\Programacion I MCD\scripts> _
```

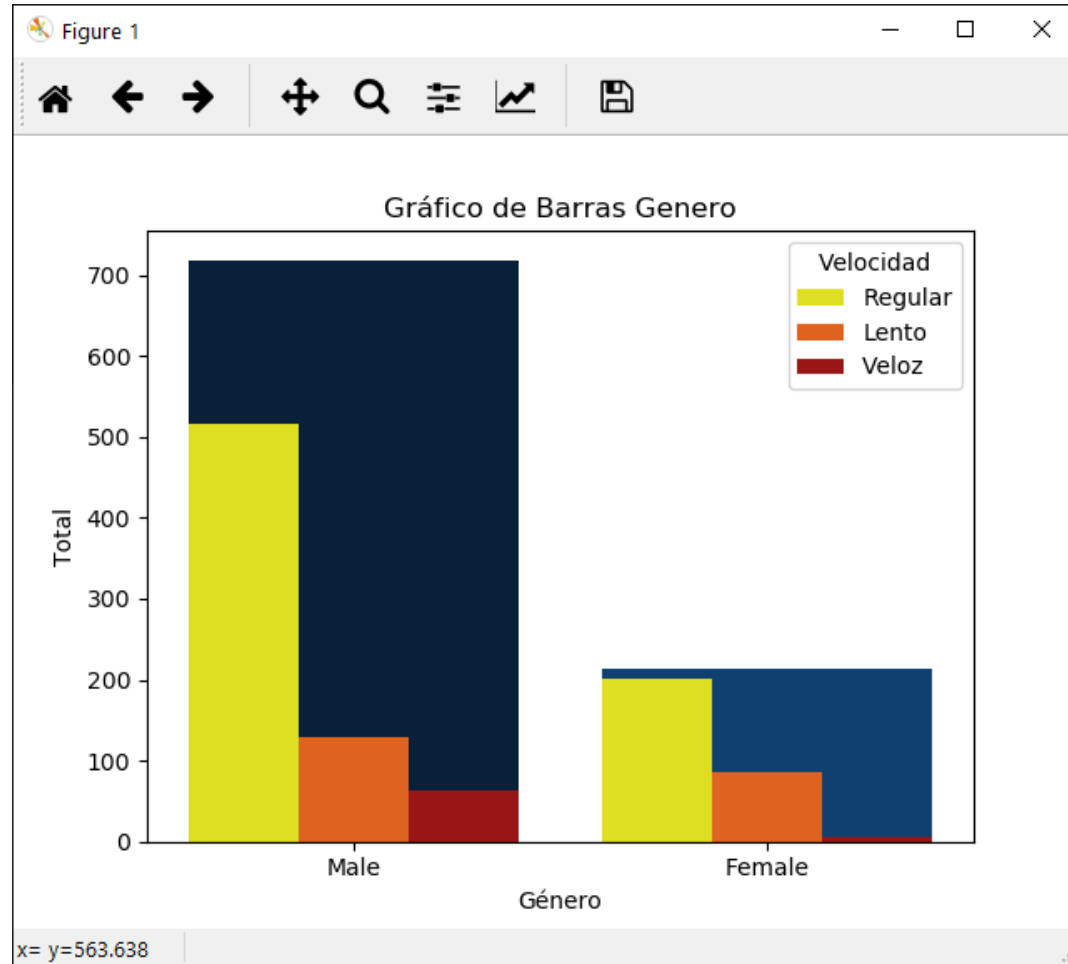
>jupyter nbconvert --to script "Histogramas desde Notebook.ipynb"
>dir

Ejecutamos el Script “Histogramas desde Notebook.py”

```
Anaconda Powershell Prompt
(base) PS C:\Users\rocio\Escritorio2\Pavillion-Rocio\Cursos impartidos Presenciales\Programacion I MCD\scripts> python "Histogramas desde Notebook.py"
C:\Users\rocio\Anaconda3
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 3475 to 8527
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0    Lugar      1000 non-null   int64
1    Genero      1000 non-null   object
2    Edad        1000 non-null   int64
3    Pais        999 non-null    object
4    Tiempo      1000 non-null   float64
5    Velocidad   1000 non-null   object
dtypes: float64(1), int64(2), object(3)
memory usage: 54.7+ KB
Attribute Qt::AA_EnableHighDpiScaling must be set before QCoreApplication is created.
(base) PS C:\Users\rocio\Escritorio2\Pavillion-Rocio\Cursos impartidos Presenciales\Programacion I MCD\scripts>
```

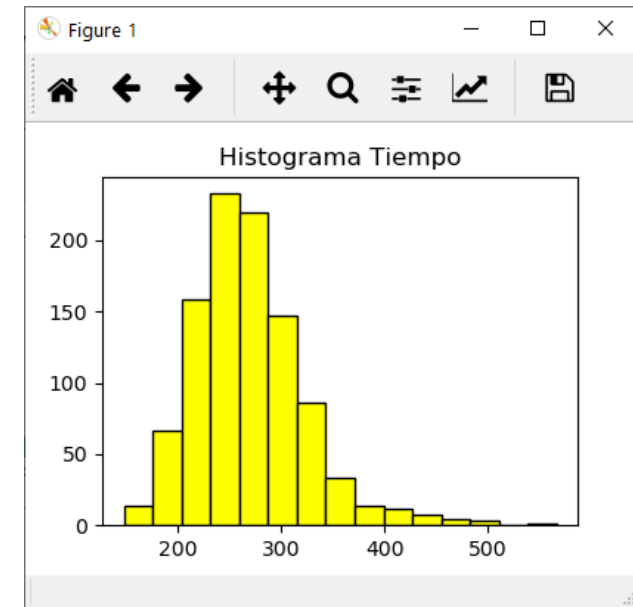
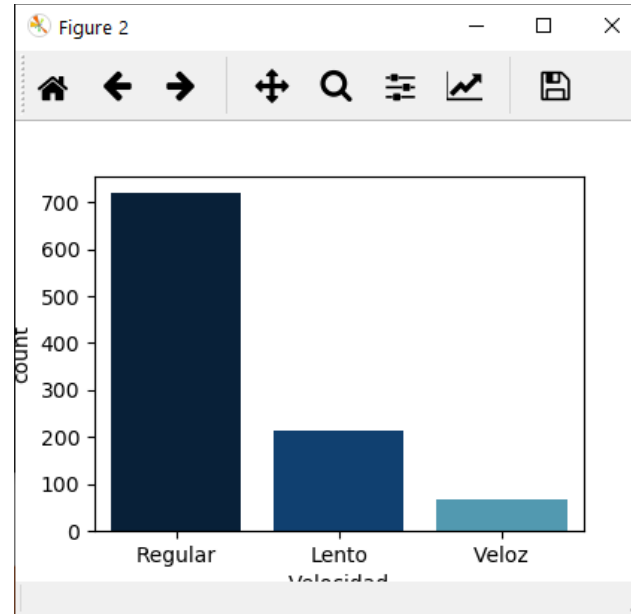
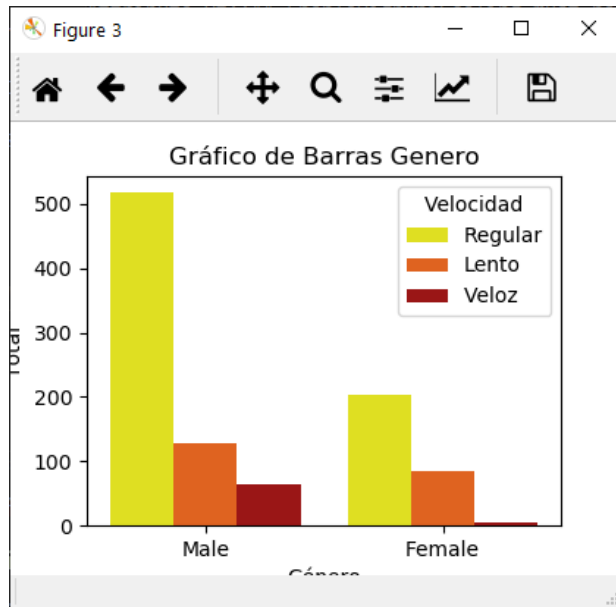
No despliega los gráficos debido a que no se le incluyó la función plt.show()

Ejecutamos el Script .py incluyendo la función plt.show()



Despliega todas las gráficas en una sola figura debido a que no se incluyeron las funciones plt.figure() para cada una de ellas

Ejecutamos el Script .py incluyendo las funciones `plt.figure()`



Unidad 2.

Fundamentos de Programación en Python

Nombres de Variables

Pueden contener letras, dígitos y guiones bajos

No pueden comenzar con un dígito

Python es case sensitive

Palabras Reservadas

```
>>> help()
```

```
Welcome to Python 3.7's help utility!
```

Anaconda Prompt - python

```
help> keywords
```

```
Here is a list of the Python keywords. Enter any keyword to get more help.
```

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

Utilizar ctrl+c para salir de la ayuda

Indentación

La indentación es importante en Python

Para indicarle al sistema que vamos a comenzar un bloque que es parte de otro bloque, suelen dejarse cuatro espacios en blanco en relación al bloque del cual forma parte



Sentencias

Instrucciones que el intérprete de Python puede ejecutar

Ejemplos:

```
x = 3
```

Sentencia de Asignación

```
print(x)
```

```
x=5
```

```
if x >0:
```

```
    print('x es un número positivo')
```

Sentencia if.. elif.. else

```
elif x<0:
```

```
    print('x es un número negativo')
```

```
else:
```

```
    print('x es igual a cero')
```

Sentencias Iterativas o Bucles

Iterar es realizar una determinada acción varias veces

Cada vez que se realiza una acción es una iteración

-Ciclo While

En ejecución se determina la finalización del bucle

- Ciclo For

Previo a la ejecución se define el número de veces que se ejecutará el bucle

Ciclo While (Mientras)

Se basa en repetir un **bloque** a partir de evaluar una **condición lógica**, siempre que esta sea **TRUE**

```
c = 0
```

```
while c <= 5:
```

```
    c+=1
```

```
    print("c vale", c)
```

Salida:

c vale 1

c vale 2

c vale ...

c vale 6

Ciclo while con un else

```
c = 0
```

```
while c <= 5:
```

```
    c+=1
```

```
    print("c vale", c)
```

```
else:
```

```
    print("Se ha completado toda la  
iteración y c vale:", c)
```

Salida:

c vale 1

c vale 2

c vale 5

c vale 6

Se ha completado toda la
iteración y c vale 6

Ciclo while con un if y break

```
c = 0
```

```
while c <= 5:
```

```
    c+=1
```

```
    if c==3:
```

```
        break
```

```
    print("c vale", c)
```

Salida:

c vale 1

c vale 2

Ciclo while con un if y continue

```
c = 0
```

```
while c <= 5:
```

```
    c+=1
```

```
    if c==3 or c==4:
```

```
        continue
```

```
    print("c vale", c)
```

Salida:

c vale 1

c vale 2

c vale 5

c vale 6

Ciclo for

Útil para recorrer listas de una manera más fácil que el ciclo while

Utilizando un ciclo while

```
numeros = [1,2,3,4,5,6]
indice = 0
while(indice < len(numeros)):
    print(numeros[indice])
    indice += 1
```

Salida:

1
2
3
4
5
6

Utilizando un ciclo for

```
numeros = [1,2,3,4,5,6]

for numero in numeros:
    print(numero)
```

Salida:

1
2
3
4
5
6

Ciclo for con enumerate

Utilizando solamente el ciclo for

```
indice = 0
numeros = [1,2,3,4,5,6]
for numero in numeros:
    numeros[indice] *= 10
    indice += 1
numeros
```

Salida:

10
20
30
40
50
60

Utilizando for con enumerate

```
numeros = [1,2,3,4,5,6]
for indice,numero in enumerate(numeros):
    numeros[indice] *= 10
numeros
```

Salida:

10
20
30
40
50
60

Ciclo for i in range

```
for i in range(10):  
    print(i)
```

Salida:

0
1
2
3
4
5
6
7
8
9

Entrada por Teclado

```
cantidad = int(input('Cantidad de pesos a convertir a dólares: '))  
print(f"{cantidad} pesos son {round(cantidad / 20.37, 2)} dólares")
```

Salida:

Cantidad de pesos a convertir a dólares: 30
30 pesos son 1.47 dólares

Tipos de Números

int: Número Entero 345

```
type(345)
```

```
int
```

Float: Número en punto Flotante 984987.87643

```
type(984987.87643)
```

```
float
```

Podemos convertir de entero a float y viceversa

```
type(float(345))
```

```
float
```

```
int(984987.87643)
```

```
984987
```



Cuidado: Se pierden los decimales, no redondea

Operaciones Aritméticas

Suma

$3+2.0$

5.0

División

$5/2$

2.5

Potencia

$3^{**}2$

9

Resta

$3-2$

1

Solo Cociente

$5//2$

2

ó

$\text{pow}(3, 2)$

9

Multiplicación

$3.0*2$

6.0

Módulo

$5\%2$

1

Órden de las Operaciones Aritméticas

- 1) Calcular lo que se encuentre dentro de paréntesis
- 2) Potencias
- 3) Productos o divisiones (De izquierda a derecha)
- 4) Sumas y restas (De izquierda a derecha)

Ejemplos:

$$6+2*8/4-2**3 = 2.0$$

$$6+2*8/4-8 = 2.0$$

$$6+4-8 = 2$$

$$10-8$$

$$2.0$$

$$(6+2)*(8/(4-2))**3 = 512.0$$

$$8*(8/2)**3 = 512.0$$

$$8*4**3 = 512$$

$$8*64 = 512$$

$$512.0$$

Agrega el .0 debido a que hay una división

Números Complejos o Imaginarios

Cuando tenemos un número negativo y obtenemos su raíz cuadrada

Tienen dos partes: una parte real 3 o 7.5 y una imaginaria 5j o -7j

$\text{parte_real} + \text{parte_imaginaria} * j$ (i de imaginario)

#Creando un número complejo

$c1 = 4 + 3j$ o $(4 - 3j)$

$(4 + 3j)$

`type(c1)`

`complex`

Dado un número complejo $z = (a + bj)$

#Magnitud (valor absoluto) o Módulo

(distancia que existe entre la parte real y la imaginaria) dentro del plano cartesiano, tienes a los números reales en un eje y a los imaginarios en otro.

$$|z| = \sqrt{a^2 + b^2}$$

$$|c1| = \sqrt{4^2 + 3^2} = 5$$

Operaciones con Números Complejos

Si tenemos dos números complejos $(a+bj)$ y $(c+dj)$

#Suma

$$c1 = 4+3j$$

$$c2 = 3+2j$$

$$c1+c2 = (a+c, (b+d)j)$$

$$(7+5j)$$

#Resta

$$c1 = 4+3j$$

$$c2 = 3+2j$$

$$c1-c2 = (a-c, (b-d)j)$$

$$(1+j)$$

#Producto Escalar

$$c1 = 4+3j$$

$$3*c1$$

$$(12+9j)$$

#División

$$c1 = 4+3j$$

$$c2 = 3+2j$$

$$c1/c2 = \left(\frac{a*c+b*d}{c^2+d^2}, \frac{b*c-a*d}{c^2+d^2}j \right)$$

$$(1.38+0.07j)$$

#Multiplicación

$$c1 = 4+3j$$

$$c2 = 3+2j$$

$$c1*c2 = (a*c-b*d, (a*d+b*c)j)$$

$$(6+17j)$$

Unidad imaginaria i es definida como la raíz cuadrada de -1

$$i^2 = (0+1j)*(0+1j)$$

$$i^2 = -1$$

Cadena o String

Cadena ordenada de caracteres

Se declaran utilizando ya sea comillas dobles o sencillas

Ejemplos:

“Este es un texto”

‘Este es un texto’

Cuando queremos citar algo de manera textual dentro de un string, se utilizan ambos tipos de comillas

```
print(“Y entonces la maestra dijo: ‘Jovenes Ilustres’..” )
```

O bien

```
print(‘Y entonces la maestra dijo: “Jovenes Ilustres”..’ )
```

*Es mejor evitar los acentos, ya
que no todos los sistemas operativos
Pueden desplegarlos sin problemas*

Literales de Cadena

```
print('\\ Para incluir un backslash en un texto ')
```

```
print('\ ' Comilla simple')
```

```
print('\ " Comillas dobles')
```

```
print('\n Salto de línea')
```

```
print('\t Tabulación horizontal')
```

```
print("Y entonces la maestra dijo: \"Jovenes Ilustres\"..")
```

Y entonces la maestra dijo: "Jovenes Ilustres"..

```
print("Y entonces la maestra dijo: \n \"Jovenes Ilustres\"..")
```

Y entonces la maestra dijo:
"Jovenes Ilustres"..

Concatenación de Strings

s1="Hola! "

s2="Como estas?"

s1 + s2

'Hola! Como estas?'

Repetición de Strings

```
s1="Hola! "
```

```
s1*3
```

```
Hola! Hola! Hola!
```


Función Print

```
x="Programación I"
```

```
print("Bienvenidos a la clase de " + x + "!")
```

Bienvenidos a la clase de Programacion I!

Cuando utilizamos el símbolo + es necesario agregar espacios entre los strings que se están concatenando

```
x="Programacion I"
```

```
print("Bienvenidos a la clase de ", x, "!")
```

Bienvenidos a la clase de Programacion I!

Cuando utilizamos comas no es necesario agregar espacios entre los strings que se están concatenando

Función str()

Con la función str(), podemos concatenar strings y variables de cualquier tipo dentro de un print

```
nombre="Maria"
```

```
edad=35
```

```
print("Mi prima " + nombre + " tiene " + str(edad) + "años")
```

Mi prima Maria tiene 35 años

Método .format()

Se utiliza para concatenar strings y variables de cualquier tipo dentro de un print.

Mediante llaves le indicamos el lugar en el que queremos que coloque el resultado de las variables y con el método .format() el orden de las mismas

```
nombre="Maria"
```

```
edad=35
```

```
print("Mi prima {} tiene {} años".format(nombre, edad))
```

```
Mi prima Maria tiene 35 años
```

Índices en Strings

```
nombre="Maria"
```

```
nombre[2] #La primer posición es 0
```

```
'r'
```

```
nombre[-1] #Último caracter de la cadena
```

```
'a'
```

```
nombre[-2] #Penúltimo caracter de la cadena
```

```
'i'
```

```
nombre[::-1] #Voltea la cadena de caracteres
```

```
'airaM'
```

Slicing en Strings

```
nombre="Maria"
```

```
nombre[0:2] #Penúltimo caracter de la cadena
```

```
'Ma'
```

La posición 2 no la toma en cuenta, solamente la 0 y la 1

```
nombre= "h" + nombre[1:]
```

```
'haria'
```

Agrega la "h" a las letras ubicadas de la posición 1 en adelante

Listas

En Python, una lista puede contener más de un tipo de dato

```
datos_varios=[3,5,7, "escritorio","borrador", 100]
```

```
datos_varios
```

```
[3, 5, 7, 'escritorio', 'borrador', 100]
```

```
type(datos_varios)
```

```
list
```

```
datos_varios[0]
```

```
3
```

```
datos_varios[-2]
```

```
'borrador'
```

Concatenar dos Listas

```
datos_varios= datos_varios + [1000, 2000]
```

```
datos_varios
```

```
[3, 5, 7, 'escritorio', 'borrador', 100, 1000, 2000]
```

Sustituir Elementos de una Lista

```
datos_varios[3] = "pizarrón"
```

```
datos_varios
```

```
[3, 5, 7, 'pizarrón', 'borrador', 100, 1000, 2000]
```

Métodos que podemos aplicar a las listas

`append()`

`extend()`

`remove()`

`index()`

`count()`

`reverse()`

Agregar elementos a una Lista con append()

```
datos_varios.append("Curso de programación I")
```

```
datos_varios
```

```
[3, 5, 7, 'escritorio', 'borrador', 100, 'Curso de Programacion I']
```

Nota: Cuando aplicamos un método, no es necesario guardar el resultado de nuevo en el objeto original:

```
datos_varios = datos_varios.append("Curso de programación I")
```

Esto haría que nuestra lista deje de ser una lista y nos arrojaría un error

Agregando una lista a otra Lista

```
datos_varios.append([7,8])
```

```
[3, 5, 7, 'escritorio', 'borrador', 100, 'Curso de Programacion I', [7, 8]]
```

Con el método append(), los elementos de las lista se agregan como un solo elemento

Agregando una lista a otra Lista con extend()

```
datos_varios.extend([20,50])
```

```
datos_varios
```

```
[3, 5, 7, 'escritorio', 'borrador', 100, 'Curso de Programacion I', 1000, 2000, 20, 50]
```

Con el método extend(), los elementos de las lista se agregan como elementos separados

Eliminar elementos a una Lista con remove()

```
datos_varios.remove("Curso de programación I")
```

```
datos_varios
```

```
[3, 5, 7, 'escritorio', 'borrador', 100, 1000, 2000, 20, 50]
```

Obtener la posición de un elemento en una Lista con index()

```
[3, 5, 7, 'escritorio', 'borrador', 100, 1000, 2000, 20, 50]
```

```
datos_varios.index(20)
```

```
8
```

Obtener la cantidad en la que un elemento se encuentra en una Lista con count()

```
[3, 5, 7, 'escritorio', 'borrador', 100, 1000, 2000, 20, 50]
```

```
datos_varios.count(20)
```

1

```
datos_varios= datos_varios + [1000, 2000]
```

```
[3, 5, 7, 'escritorio', 'borrador', 100, 1000, 2000, 20, 50, 1000, 2000]
```

```
datos_varios.count(1000)
```

2

Invertir el orden de los elementos en una Lista con reverse()

```
[3, 5, 7, 'escritorio', 'borrador', 100, 1000, 2000, 20, 50, 1000, 2000]
```

```
datos_varios.reverse()  
datos_varios
```

```
[2000, 1000, 50, 20, 2000, 1000, 100, 'borrador', 'escritorio', 7, 5, 3]
```



Tarea 2

Crear un script en el que lleves a cabo un ejemplo de cada uno de los comandos que vimos el día de hoy

Subir el script al classroom