Programming in Python Lecture 8- Matplotlib & TKinter

Plan for today

- Matplotlib
- Tkinter

Reminder

- Numpy
- Pandas

NumPy

- The **NumPy** (Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.
- NumPy are open-source, and therefore provide a free Matlab alternative.
- The package are popular among scientists, researchers and engineers who want to apply various mathematical methods on large datasets.

NumPy's main object is the **Array**

- NumPy's main object is the homogeneous multidimensional **array**.
- It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
- In Numpy dimensions are called ***axes***.
- The number of axes is ***rank***.
  - *A **vector** is an array of rank 1*
  - *A 2D **matrix** is an array of rank 2.*

The Array object – Creation

```
>>> import numpy as np
>>> a = np.array([0, 1, 2])
array([0,1,2])
```

 # 1 x 3 array

0	1	2

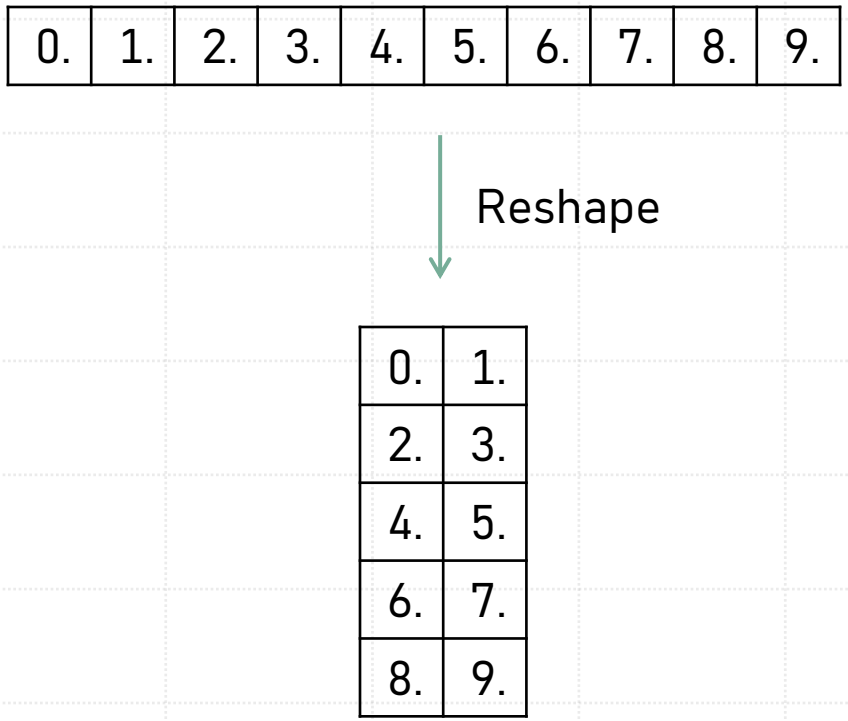
```
>>> b = np.array([[0, 1, 2], [3, 4, 5]])
array([[0,1,2],
       [3,4,5]])
```

 # 2 x 3 array

0	1	2
3	4	5

Reshaping an Array

```
>>> Mat
array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])

>>> Mat = a.reshape((5, 2))
Mat array([[ 0., 1.],
       [ 2., 3.],
       [ 4., 5],
       [ 6., 7.],
         [ 8., 9.]])
>>> Mat.shape
(5, 2)
```



Array Arithmetic

Arithmetic operators on arrays apply *elementwise*. A new array is created and filled with the result.

```
>>> x = np.array([1,5,2])
>>> y = np.array([7,4,1])
>>> x + y
array([8, 9, 3])
>>> x * y   # element by element multiplication ! Use np.dot(x,y) for matrix multiplication.
array([ 7, 20,  2])
>>> x - y
array([-6,  1,  1])
>>> x / y
array([0, 1, 2])
>>> x % y
array([1, 1, 0])
```

Note that here, x and y have the same size

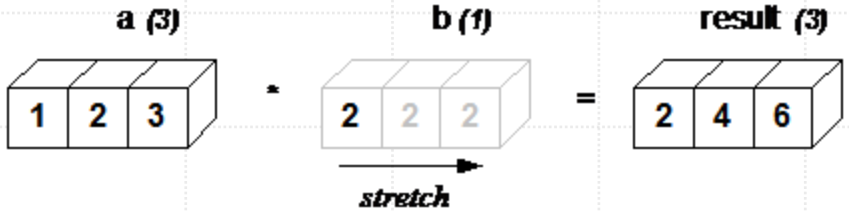
Broadcasting

- NumPy operations are usually done element-by-element which requires two arrays to have exactly the same shape:

```
>>> a = array([1.0,2.0,3.0])
>>> b = array([2.0,2.0,2.0])
>>> a * b
array([ 2.,  4.,  6.])
```

- But this will also work:

```
>>> a = array([1.0,2.0,3.0])
>>> b = 2.0
>>> a * b
array([ 2.,  4.,  6.])
```



Comparisons and Boolean operations

```
>>> a = np.random.random_integers(0, 10,
15)
array([ 4,  9,  1,  7,  0,  9,  8, 10,  1,  0,  7,  7,  9,
5,  1])
```

Comparisons and Boolean operations

```
>>> a = np.random.random_integers(0, 10, 15)
>>> b = np.random.random_integers(0, 10, 15)
>>> comp1 = a==b
>>> comp1
array([False, False, False, False, False,  True, False, False, False,   False,
False, False, False, False, False], dtype=bool)
>>> comp1.any()
True
>>> comp1.all()
False
```

Any is true?

Are all true?

Get the True indices

Fancy indexing: Logical indexing

```
>>> a = np.random.random_integers(0, 20,
  15)
>>> (a % 3 == 0)
```

```
array([False, True, False, True, False, False, False, True,
False,True, True, False, True, False, False], dtype=bool)
```

Sorting along an axis

```
>>> a = np.array([[4, 3, 5], [1, 2, 1]])
>>> b = np.sort(a, axis=1)
>>> b
array([[3, 4, 5],
[1, 1, 2]])
```

Pandas

- **Pandas** is a library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Read CSV-head()

```
>>> chiptole=pd.read_csv("chipotle.tsv", sep = '\t')
```

```
>>> print(chiptole)
```

```
      order_id  ...  item_price
0            1  ...        2.39
1            1  ...        3.39
2            1  ...        3.39
3            1  ...        2.39
4            2  ...       16.98
         ...  ...         ...
4617      1833  ...       11.75
4618      1833  ...       11.75
4619      1834  ...       11.25
4620      1834  ...        8.75
4621      1834  ...        8.75
```

```
>>> print(chiptole.head(5))
```

```
   order_id  ...  item_price
0         1  ...        2.39
1         1  ...        3.39
2         1  ...        3.39
3         1  ...        2.39
4         2  ...       16.98

[5 rows x 5 columns]
```

Read CSV-tail()

```
>>> chiptole=pd.read_csv("chipotle.tsv", sep = '\t')
```

```
>>> print(chipotle.tail())
```

```
      order_id  ...  item_price
4617      1833  ...       11.75
4618      1833  ...       11.75
4619      1834  ...       11.25
4620      1834  ...        8.75
4621      1834  ...        8.75

[5 rows x 5 columns]
```

```
>>> print(chiptole.tail(1))
```

```
      order_id  ...  item_price
4621      1834  ...        8.75

[1 rows x 5 columns]
```

Read CSV-select

>>> chiptole[["order_id", "quantity", "choice_description"]]

	order_id	quantity	choice_description
0	1	1	NaN
1	1	1	[Clementine]
2	1	1	[Apple]
3	1	1	NaN
4	2	2	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...
...	...		...
4617	1833	1	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...
4618	1833	1	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...
4619	1834	1	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...
4620	1834	1	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...
4621	1834	1	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...

[4622 rows x 3 columns]

>>>chiptole[["order_id","quantity","choice_description"]].head()

	order_id	quantity	choice_description
0	1	1	NaN
1	1	1	[Clementine]
2	1	1	[Apple]
3	1	1	NaN
4	2	2	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...

Read CSV-Calculation

```
>>> chiptole["item_price_Add"] = chiptole["item_price"].add(0.85)
>>> chiptole["item_price_Add"].head()
```

```
0     3.24
1     4.24
2     4.24
3     3.24
4    17.83
Name: item_price_Add, dtype: float64
```

```
>>> chiptole["item_price_Sub"] = chiptole["item_price"].sub(0.85)
>>> chiptole[["item_price","item_price_Sub"]].head()
```

0	2.39	1.54
1	3.39	2.54
2	3.39	2.54
3	2.39	1.54
4	16.98	16.13

Read CSV-Insert column

```
>>> chiptole.insert(loc = 4,
 column = "diff",
 value = chiptole["item_price"] - chiptole["item_price_discount"])
>>> chiptole[["item_price","item_price_discount","diff"]].head()
```

	item_price	item_price_discount	diff
0	2.39	2.0315	0.3585
1	3.39	2.8815	0.5085
2	3.39	2.8815	0.5085
3	2.39	2.0315	0.3585
4	16.98	14.4330	2.5470

Read CSV-Assigning/Rename

```
>>>chiptole2=chiptole.assign(pct = chiptole["item_price_discount"]
/ chiptole["item_price"] * 100).head()
```

```
>>> print(chiptole2[["pct","diff"]])
```

	pct	diff
0	85.0	0.3585
1	85.0	0.5085
2	85.0	0.5085
3	85.0	0.3585
4	85.0	2.5470

```
>>>chiptole[["order_id", "quantity", "item_price_discount"]]\
.rename(columns = {"item_price_discount":"ip_dsc"})\
.head()
```

	order_id	quantity	ip_dsc
0	1	1	2.0315
1	1	1	2.8815
2	1	1	2.8815
3	1	1	2.0315
4	2	2	14.4330

Read excel

```
>>> products=pd.read_excel(r"products.xlsx")
```

```
>>> print(products.head())
```

	ProductID	ProductName	UnitPrice	CategoryID
0	1	Chai	18.00	1
1	2	Chang	19.00	1
2	3	Aniseed Syrup	10.00	2
3	4	Chef Anton's Cajun Seasoning	22.00	2
4	5	Chef Anton's Gumbo Mix	21.35	2

To excel/csv

```
>>> DF1=np.random.randint(100,200,8)
>>> DF2=np.random.randint(100,200,8)

>>> df = pd.DataFrame([DF1[0:len(DF1)], DF2[0:len(DF2)]],
                      index=['X', 'Y'],
                      columns=range(1, len(DF2)+1))

>>> df.to_excel('Data1.xlsx')
>>> df.to_csv('Data1.xlsx')
```

Homework

1. Create a Python script that take a list of dictionaries and create an excel file that including the id ,Date and S_Id columns by the dictionary's keys(use the .item() to extract the values from the dictionaries)

Questions?

Plan for today

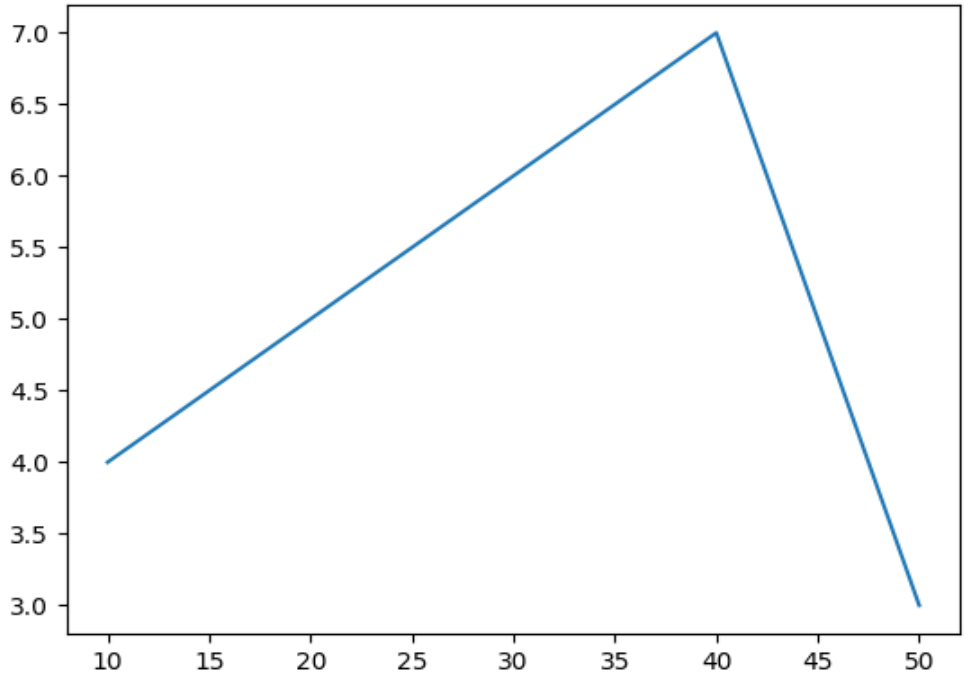
- **Matplotlib**
- Tkinter

Matplotlib

- The **Matplotlib** package is a comprehensive library for creating static, animated, and interactive visualizations in Python
- Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, IPython shell, web application servers, and various graphical user interface toolkits.

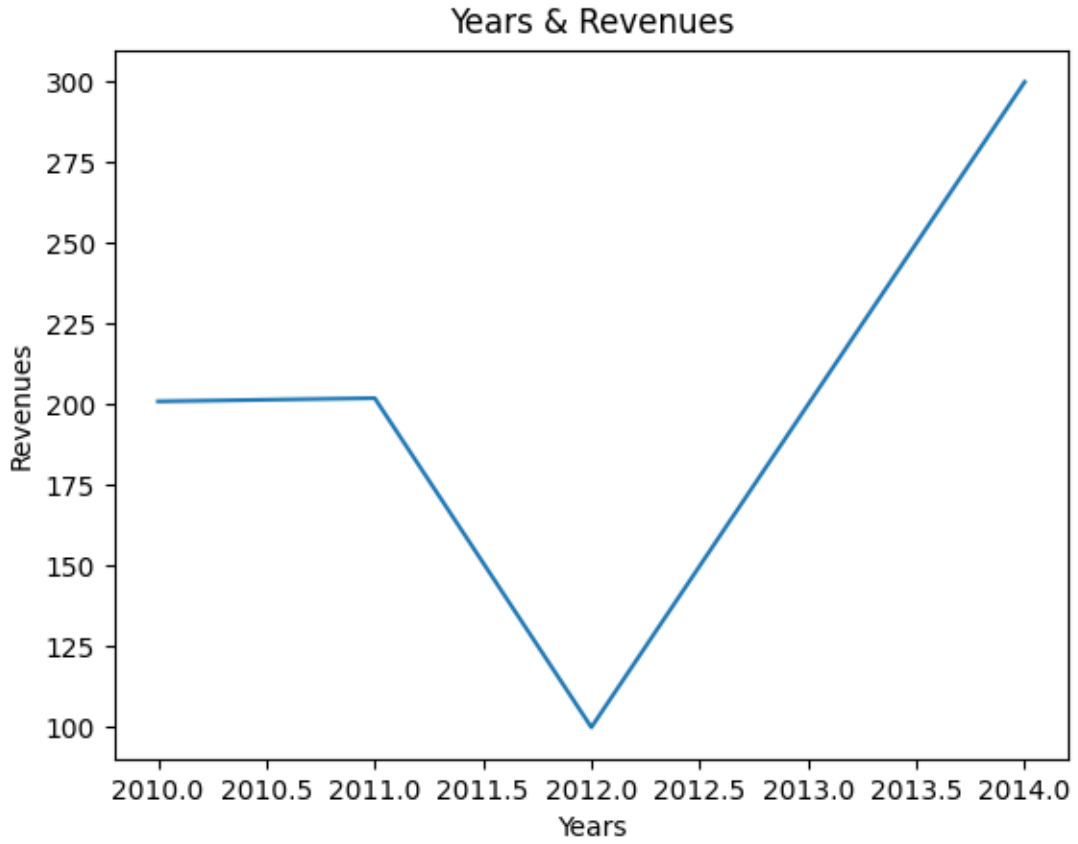
Plotting-Line plot

```
>>> import numpy as np
>>>import matplotlib.pyplot as plt

>>>plt.plot([10,20,30,40,50], [4,5,6,7,3])
>>> plt.show()
```



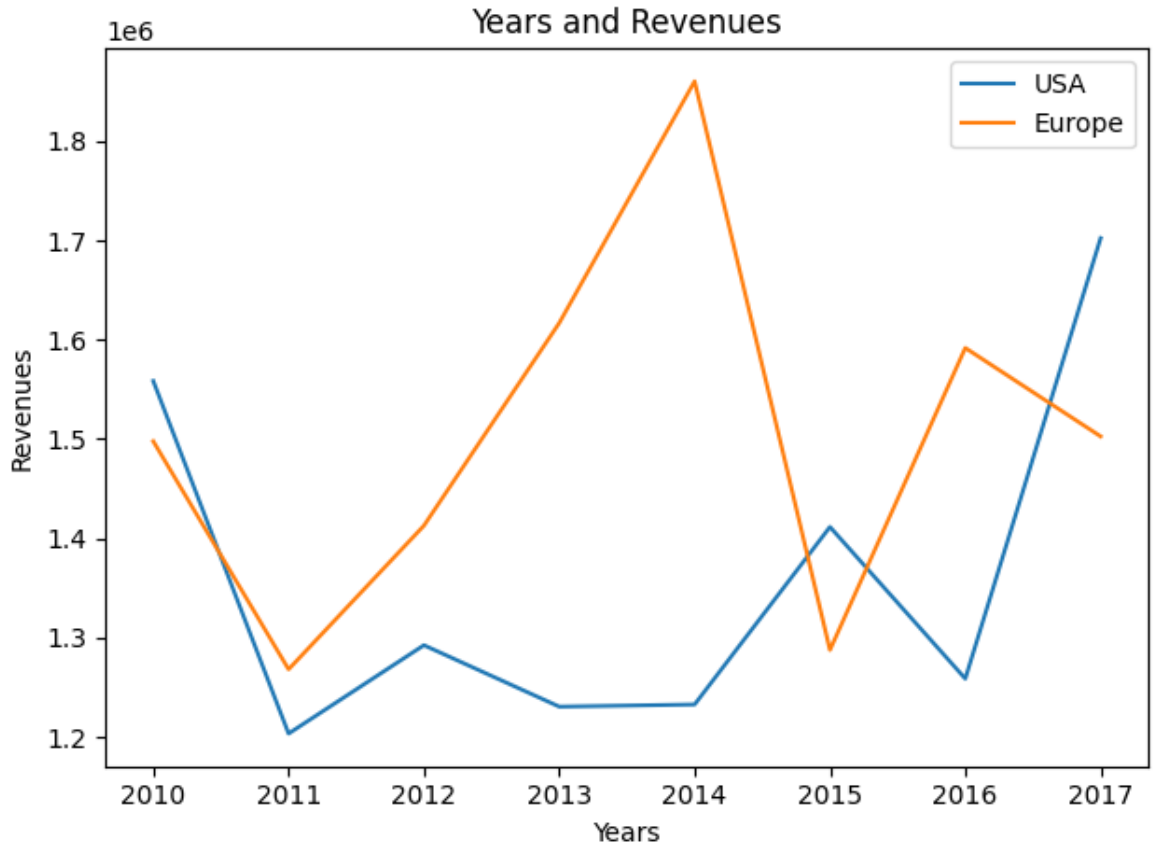
Plotting-labels

```
>>> def Plot(a,b,c,d,e):
    x_axis =[2010,2011,2012,2013,2014]
    y_axis =[a,b,c,d,e]
    plt.plot(x_axis, y_axis)
    plt.xlabel('Years')
    plt.ylabel('Revenues')
    plt.title('Years & Revenues')
    plt.show()
>>>Plot(201,202,100,200,300)
```



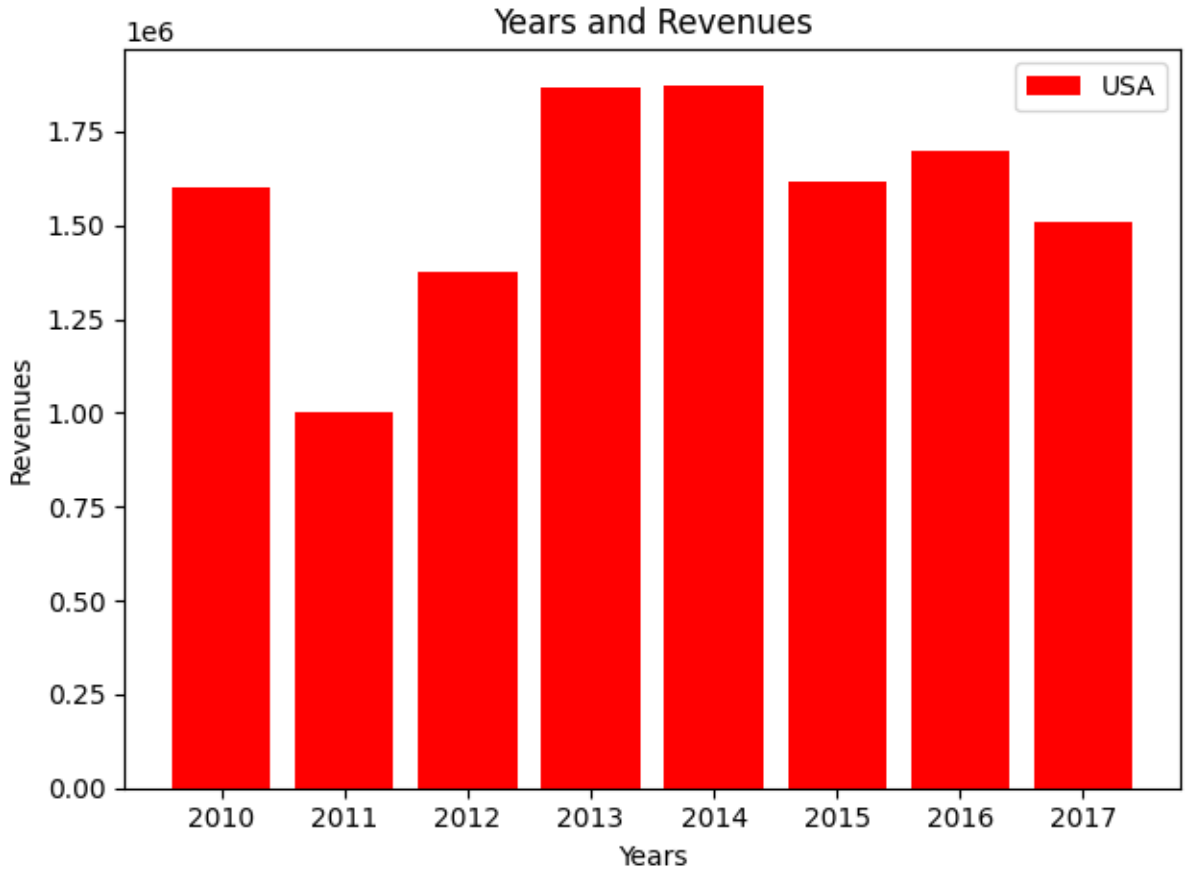
Plotting-legends

```
>>> x_axis = np.arange(2010,2018)
>>> y_axis_1 = np.random.randint(1000000,2000000,8)
>>> y_axis_2 = np.random.randint(1000000,2000000,8)
plt.plot(x_axis, y_axis_1, label='USA', )
plt.plot(x_axis, y_axis_2, label='Europe')
plt.xlabel('Years')
plt.ylabel('Revenues')
plt.title('Years and Revenues')
plt.tight_layout()
plt.legend()
>>> plt.show()
```



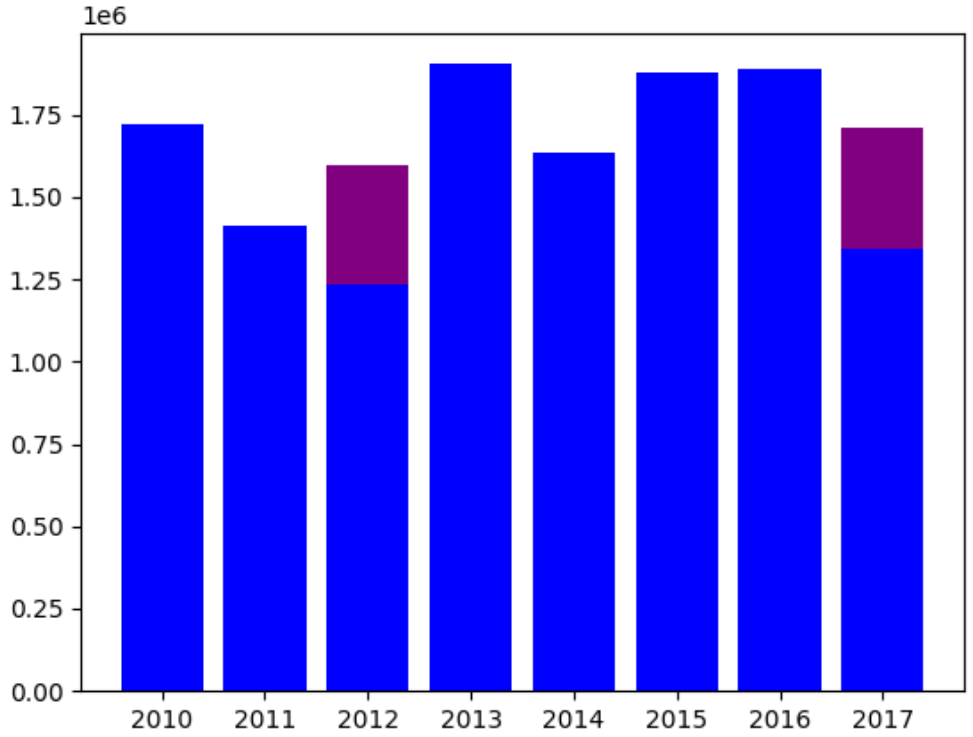
Plotting-Bar Charts

```
>>> x_axis_1 = np.arange(2010,2018)
>>> y_axis_1 = >>> np.random.randint(1000000,2000000,8)
plt.bar(x_axis_1, y_axis_1, label="USA", color='red')
plt.xlabel('Years')
plt.ylabel('Revenues')
plt.title('Years and Revenues')
plt.legend()
plt.tight_layout()
>>> plt.show()
```



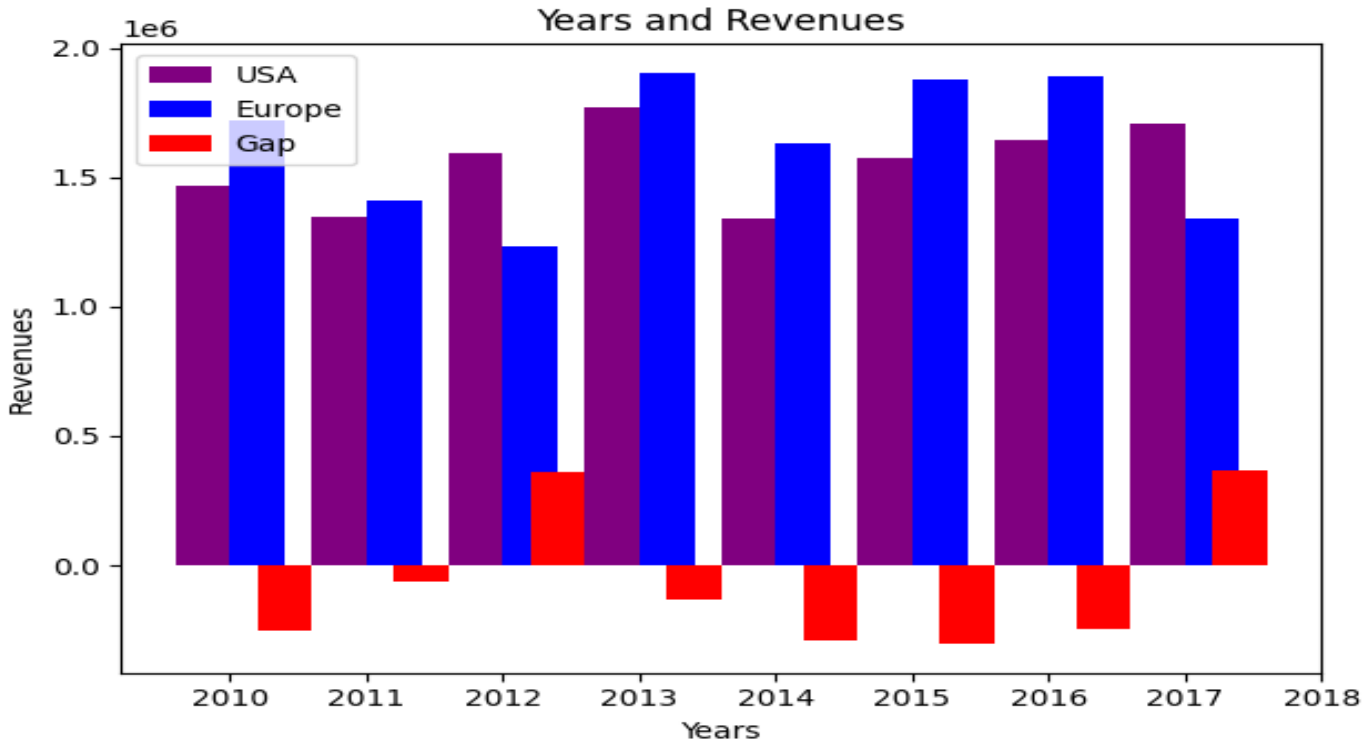
Plotting-Bar Charts-overlapping values

```
>>> Bar Charts-overlapping values
x_axis_1 = np.arange(2010,2018)
y_axis_1 = np.random.randint(1000000,2000000,8)
y_axis_2 = np.random.randint(1000000,2000000,8)
y_axis_3=y_axis_1-y_axis_2
plt.bar(x_axis_1, y_axis_1,  label="USA", color='purple')
plt.bar(x_axis_1, y_axis_2,  label="Europe", color='blue')
plt.xlabel('Years')
plt.ylabel('Revenues')
plt.title('Years and Revenues')
plt.legend()
plt.tight_layout()
>>>plt.show()
```



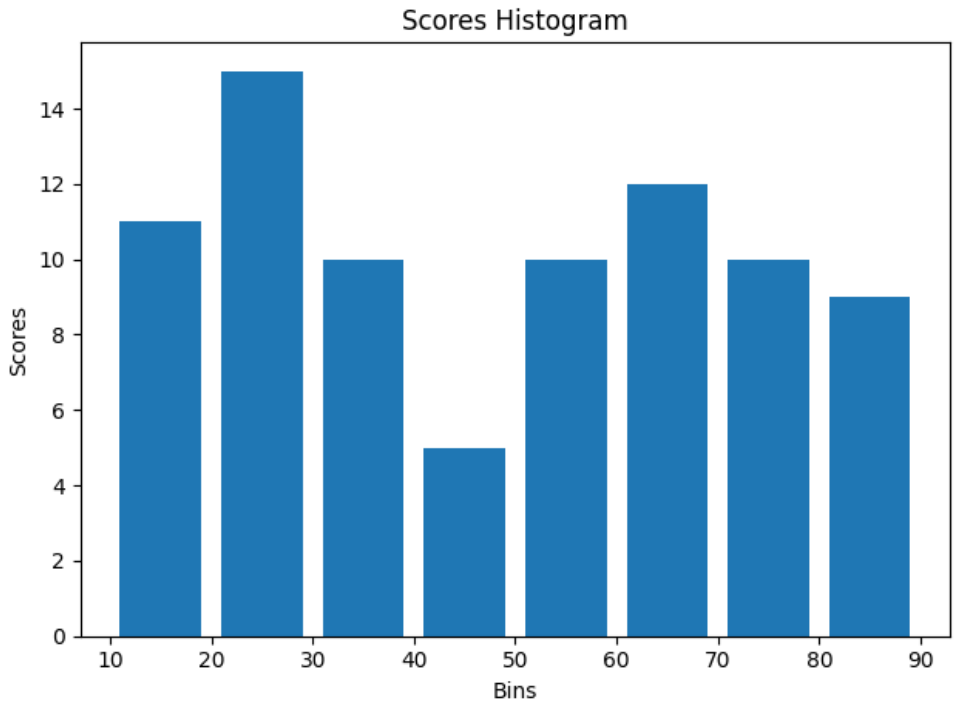
Plotting-Bar Charts-overlapping values-columns

```
>>> plt.bar(x_axis_1-0.2, y_axis_1, width=0.4, label="USA",color='purple')
plt.bar(x_axis_1+0.2, y_axis_2, width=0.4, label="Europe", color='blue')
plt.bar(x_axis_1+0.4, y_axis_3, width=0.4, label="Gap", color='red')
plt.xlabel('Years')
plt.ylabel('Revenues')
plt.title('Years and Revenues')
plt.legend()
plt.tight_layout()
>>> plt.show()
```



Plotting- Histograms

```
>>>bins = np.arange(10,100, 10)
>>> scores = np.random.randint(0,100,100)


plt.hist(scores, bins, histtype='bar', rwidth=0.8)
plt.xlabel('Bins')
plt.ylabel('Scores')
plt.title('Scores Histogram')
plt.tight_layout()
>>> plt.show()
```



Plotting-Scatter Plots

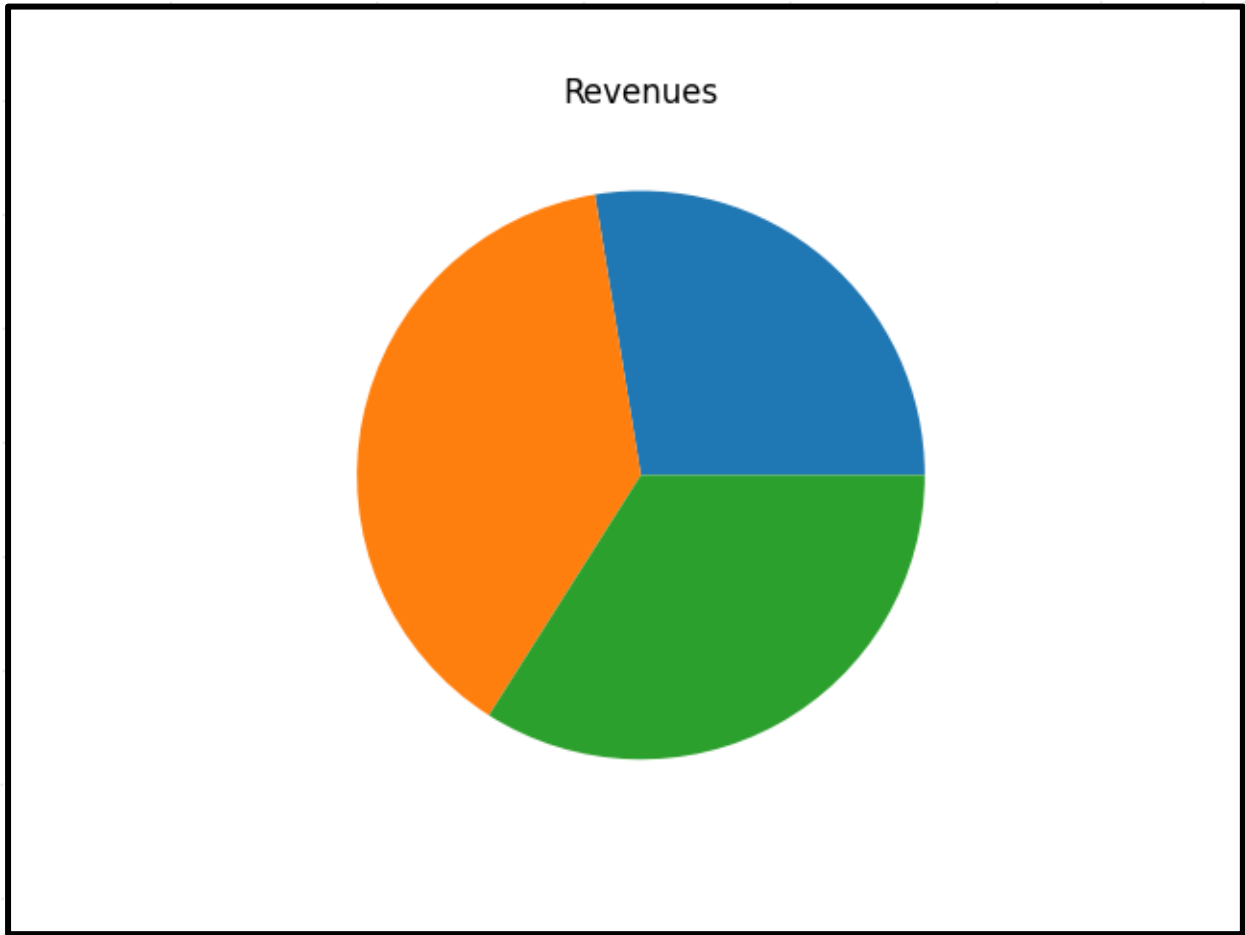
```
>>>scores  = np.random.randint(0,100,100)
>>> time_spent = np.random.randint(30,60,100)
```

```
plt.scatter(time_spent, scores)
```

```
plt.title('Test scores vs Time Spent')
plt.xlabel('Time Spent on Test')
plt.ylabel('Test Score')
>>> plt.show()
```

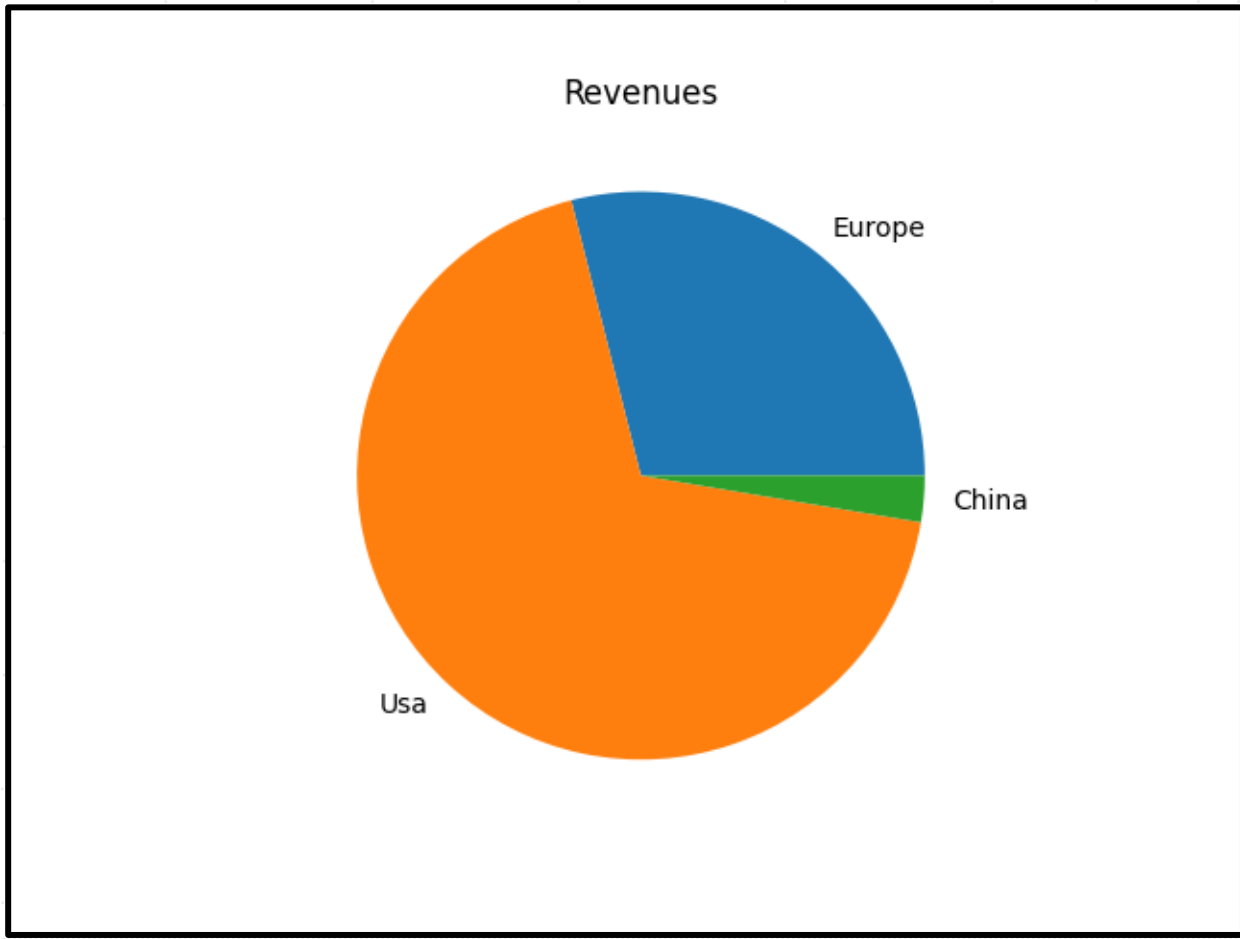
Plotting-Pie chart

```
>>> p_size = np.random.randint(0,100,3)
plt.pie(p_size)
plt.title('Revenues')
>>>plt.show()
```



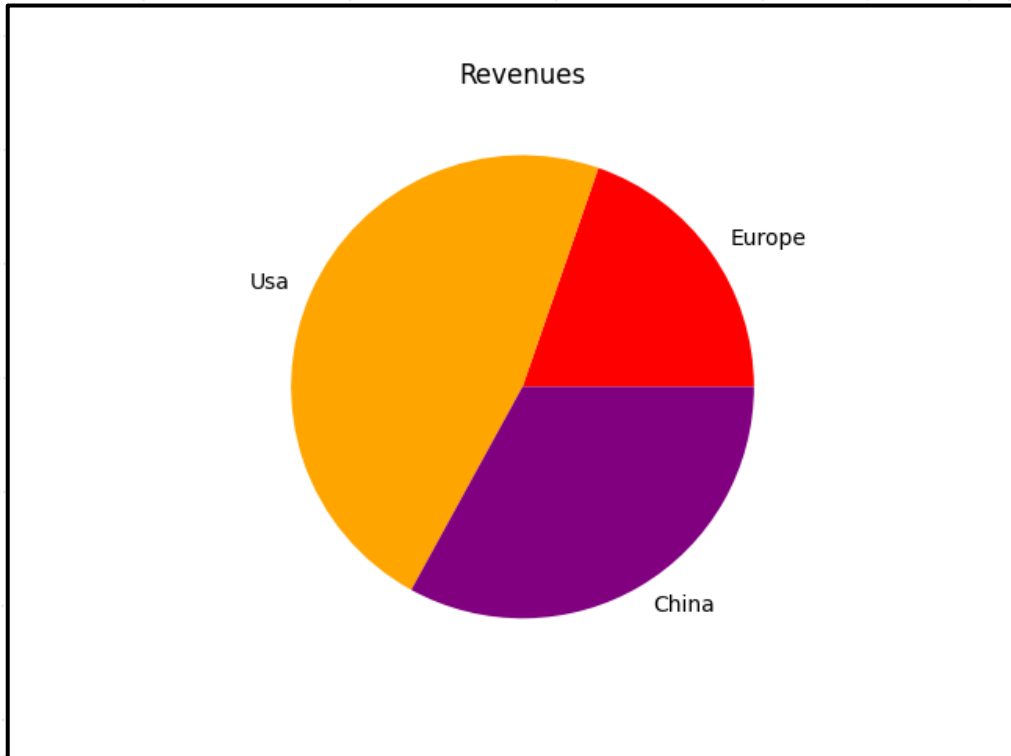
Plotting-Pie chart labels

```
>>> p_size = np.random.randint(0,100,3)
plt.xlabel('Years')
plt.ylabel('Revenues')
plt.title('Years & Revenues')
 >>> plt.show()
```



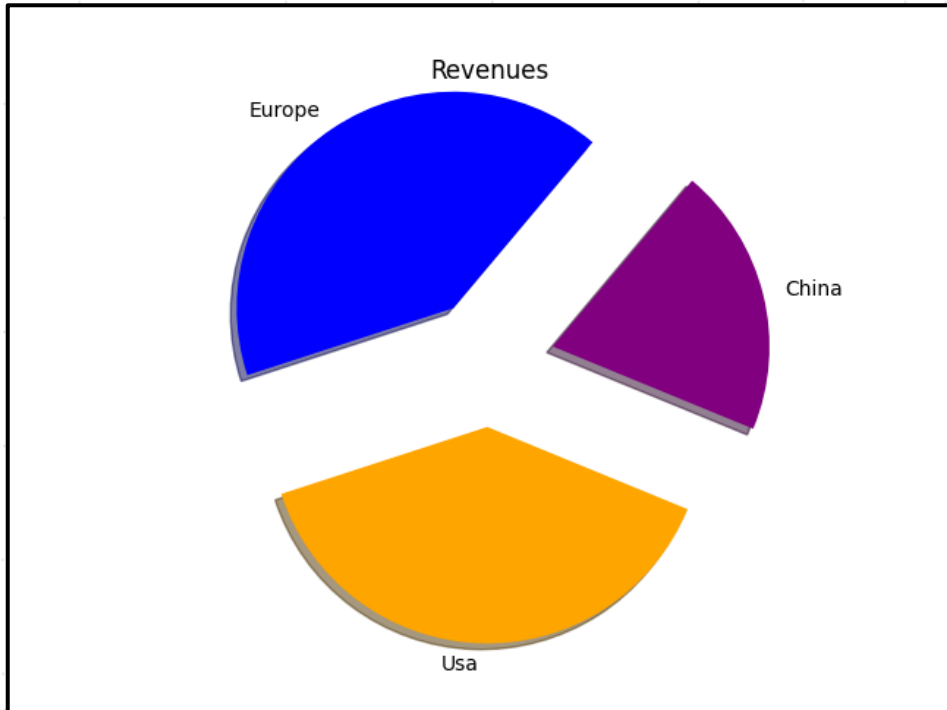
Plotting-Pie chart labels colors

```
>>> p_size =np.random.randint(0,100,3)
>>> p_labels = 'Europe', 'Usa', 'China'
>>> p_colors = ['red','orange','purple']
plt.pie(p_size, labels=p_labels, colors=p_colors)
plt.title('Revenues')
>>>plt.show()
```



Plotting-Pie chart explode

```
>>> p_size = np.random.randint(0,100,3)
>>> p_labels = 'Europe', 'Usa', 'China'
>>> p_colors = ['blue','orange','purple']
plt.pie(p_size, labels=p_labels, colors=p_colors,
startangle=50, explode = (0.3, 0.3, 0.3),
shadow=True)
plt.title('Revenues')
>>> plt.show()
```



Questions?

Hands On

Plan for today

- **Matplotlib**
- **Tkinter**

TKinter

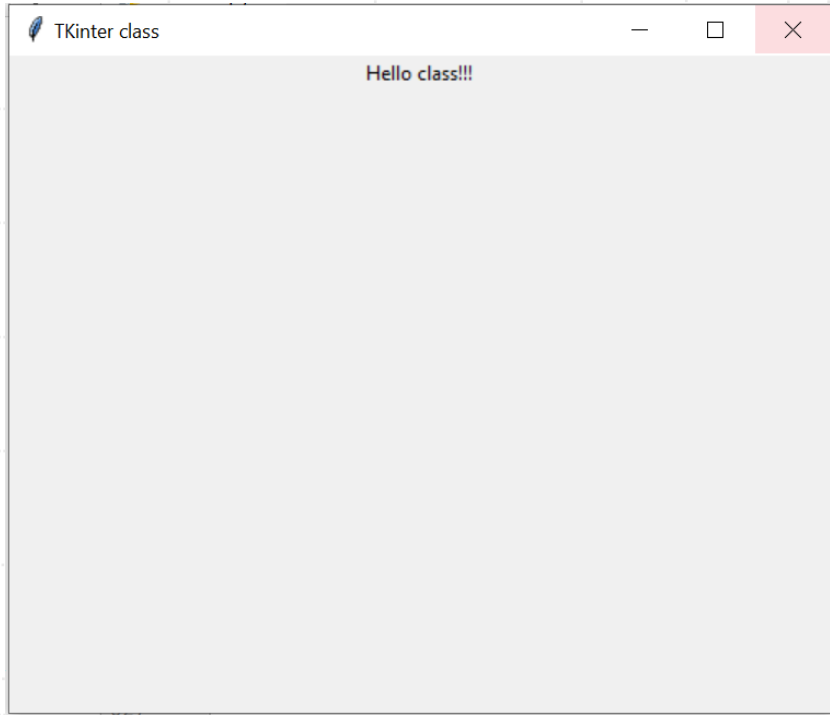
- Python has a lot of GUI(graphical user interface) frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

- Although Tkinter is considered the de facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

- However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to quickly build something that's functional and cross-platform.

Creating a window

```
>>> from tkinter import *

>>>     window = Tk()
        window.title("TKinter class")
        window.minsize(500, 400)
        my_label = Label(text="Hello class!!!")
        my_label.pack()
        window.mainloop()
```



Left top corner

```
>>> root = tk.Tk()
root.title('TKinter geomatry left')
root.geometry('600x400+50+50')
root.mainloop()    plt.show()
>>>Plot(201,202,100,200,300)
```

Right top corner

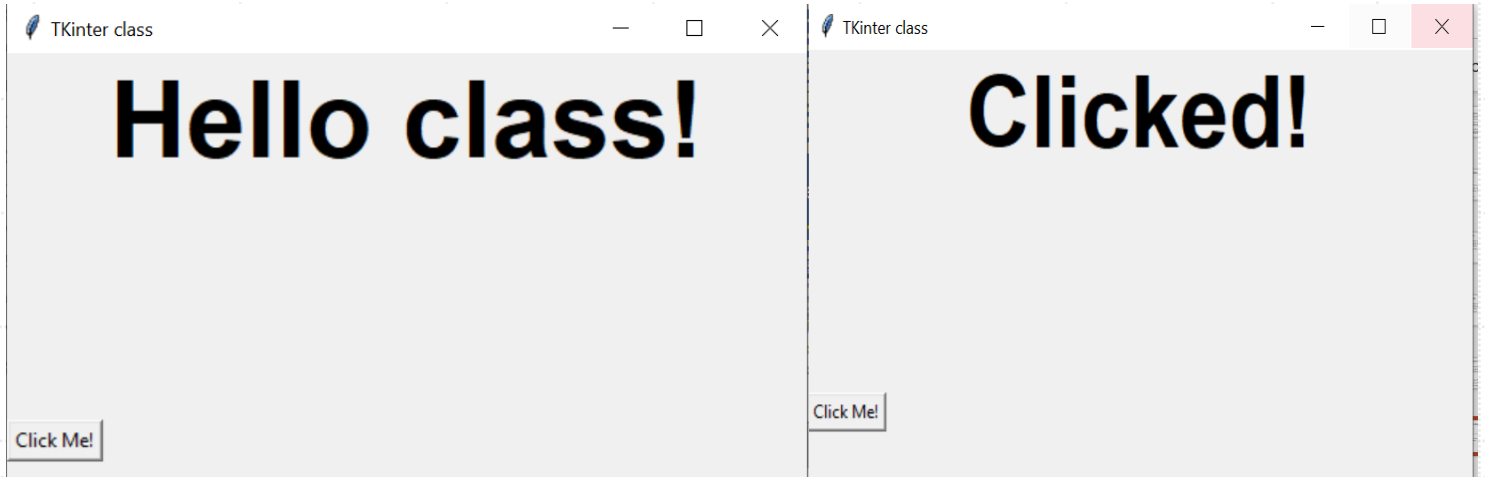
```
>>> root = tk.Tk()
root.title(‘TKinter geomatry right')
root.geometry('600x400-50-50')
root.mainloop()    plt.show()
>>>Plot(201,202,100,200,300)
```

Center top corrner

```
>>> root = tk.Tk()
root.title(TKinter geomatry center')
sw = root.winfo_screenwidth()
sh = root.winfo_screenheight()
window_width = 600
window_height = 400
dw = (sw - window_width) // 2
dh = (sh - window_height) // 2
root.geometry(f'{window_width}x{window_height}+{dw}+{dh}')
root.mainloop()
```

Labels, Buttons & commend

```
>>>window = Tk()
window.title("TKinter class")
window.minsize(500, 400)
my_label = Label(text="Hello, World!", font=("Arial", 50, "bold"))
my_label.pack()
>>>button = Button(text="Click Me!", command=lambda:
my_label.config(text="Clicked!"))
button.pack(side="left")
window.mainloop()
button.config()
```



Labels, Buttons & commend-with function

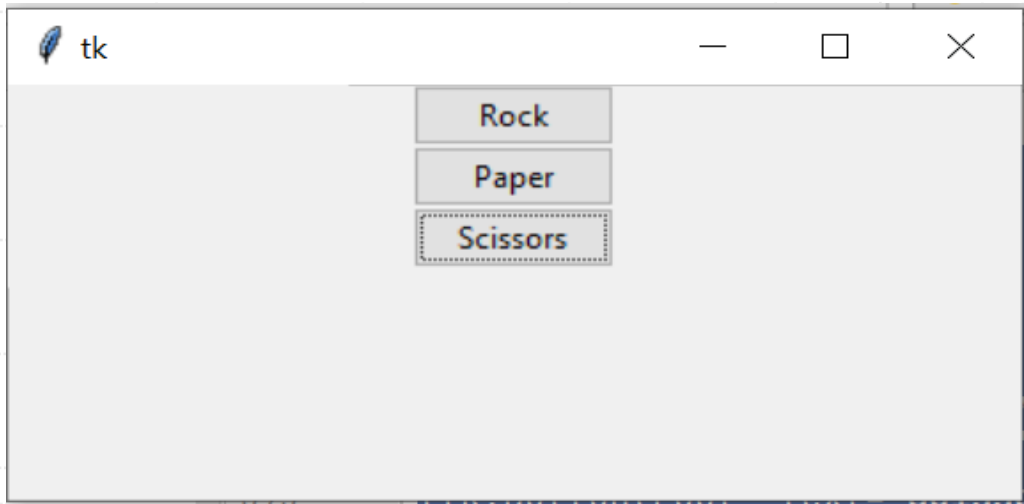
```
>>>window = Tk()
window.title("TKinter class")
window.minsize(500, 400)
my_label = Label(text="Hello class!", font=("Arial", 50, "bold"))
my_label.pack()
>>>def action():
 print("Do something")
>>>button = Button(text="Click Me!", command=lambda:
my_label.config(text="Clicked!"))
button.pack(side="left")
window.mainloop()
button.config()
```

Labels, Buttons & commend-with function

```
>>>window = Tk()
window.title("TKinter class")
window.minsize(500, 400)
my_label = Label(text="Hello class!", font=("Arial", 50, "bold"))
my_label.pack()
>>>def action():
 print("Do something")
>>>button = Button(text="Click Me!", command=lambda:
my_label.config(text="Clicked!"))
button.pack(side="left")
window.mainloop()
button.config()
```

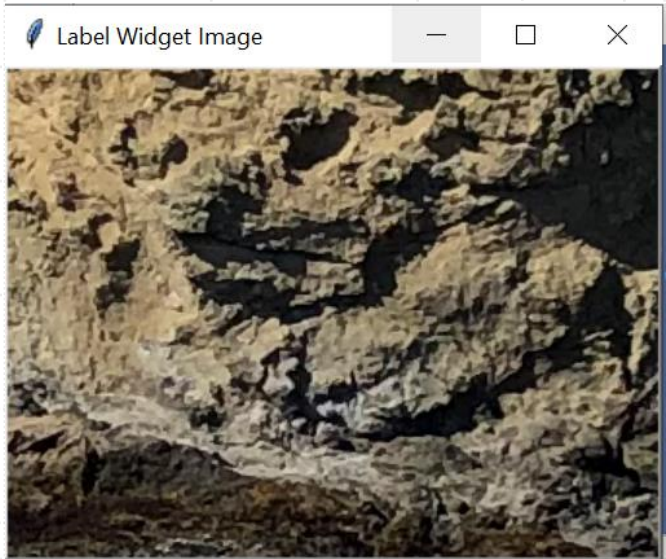
Multipale options

```
>>>from tkinter import ttk
>>>root = tk.Tk()
def select(option):
 print(option)
ttk.Button(root, text='Rock', command=lambda: select('Rock')).pack()
ttk.Button(root, text='Paper',command=lambda: select('Paper')).pack()
ttk.Button(root, text='Scissors', command=lambda:select('Scissors')).pack()
root.mainloop()
```



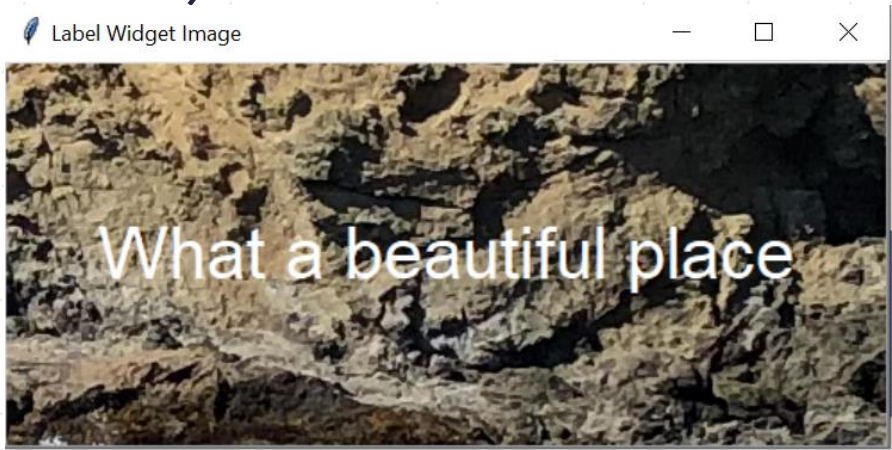
Insert a photo

```
>>>root = Tk()
root.geometry('300x200')
root.title('Label Widget Image')
photo = PhotoImage(file='IMG_5382.png')
image_label =
Button(root,image=photo,text='Python',fg="orange",font=("Helvetica", 30),
compound='center')
image_label.config(text ="Welcome to Data Science Learner!")
image_label.pack()
mainloop()
```



Insert text to a photo

```
>>>root = Tk()
root.geometry('300x200')
root.title('Label Widget Image')
photo = PhotoImage(file='IMG_5382.png')
image_label = Button(root,image=photo,text='Python',fg="orange",font=("Helvetica",
30),
compound='center')
image_label.config(text ="Welcome to Data Science Learner!")
image_label.config(text ="What a beautiful place“
,foreground = "white")
image_label.pack()
mainloop()
```

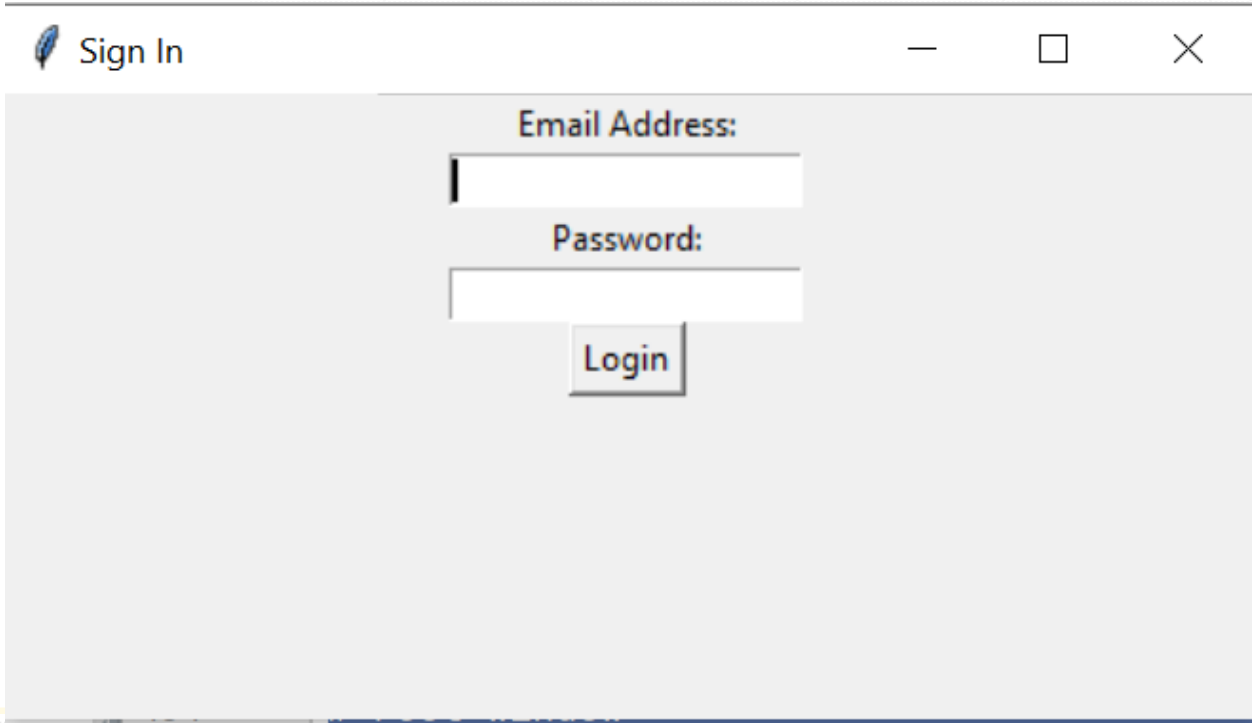


Entry

```
window = Tk()
window.minsize(500, 400)
window.title('Entry')

# Set the text color to white

my_label = Label(text="Please enter ausername", foreground="white",
 background="black", font=("Arial", 50, "bold"))
my_label.pack()
label = Label(text="username")
label.pack(side="right")
entry = Entry()

entry.pack(side="right")

button = Button(text="Click Me!", command=lambda: my_label.config(text="Clicked!"))
button.config(command=lambda: {my_label.config(text=entry.get())})
button.pack(side="left")
window.mainloop()
```

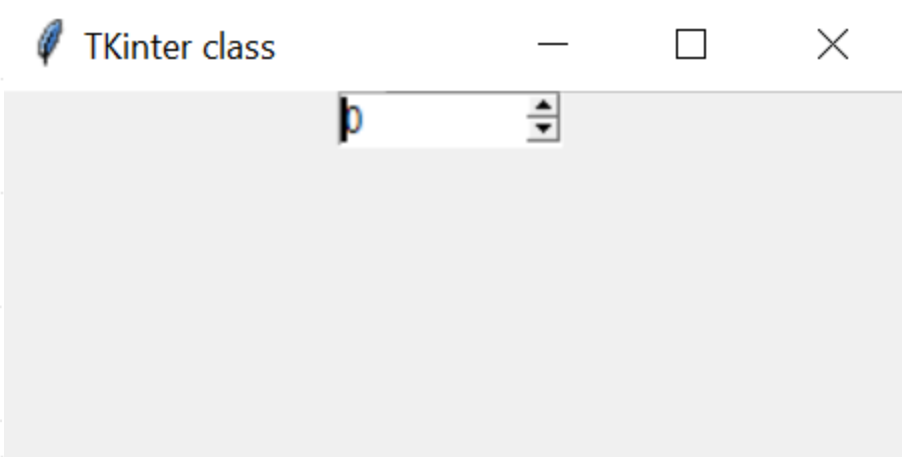
Password

```
from tkinter.messagebox import showinfo
root = Tk()
root.title('Sign In')
email = StringVar()
password = StringVar()
def login_clicked():
 print(password_entry.get())
 msg = f'You entered email: {email.get()} and password: {password.get()}'
 showinfo(
 title='Information',
 message=msg)
email_label = Label(text="Email Address:")
email_label.pack()
email_entry = Entry(textvariable=email)
email_entry.pack()
email_entry.focus()
password_label = Label(text="Password:")
password_label.pack()
password_entry = Entry(textvariable=password, show="*")
password_entry.pack()
login_button = Button(text="Login", command=login_clicked)
login_button.pack()
root.mainloop()
```



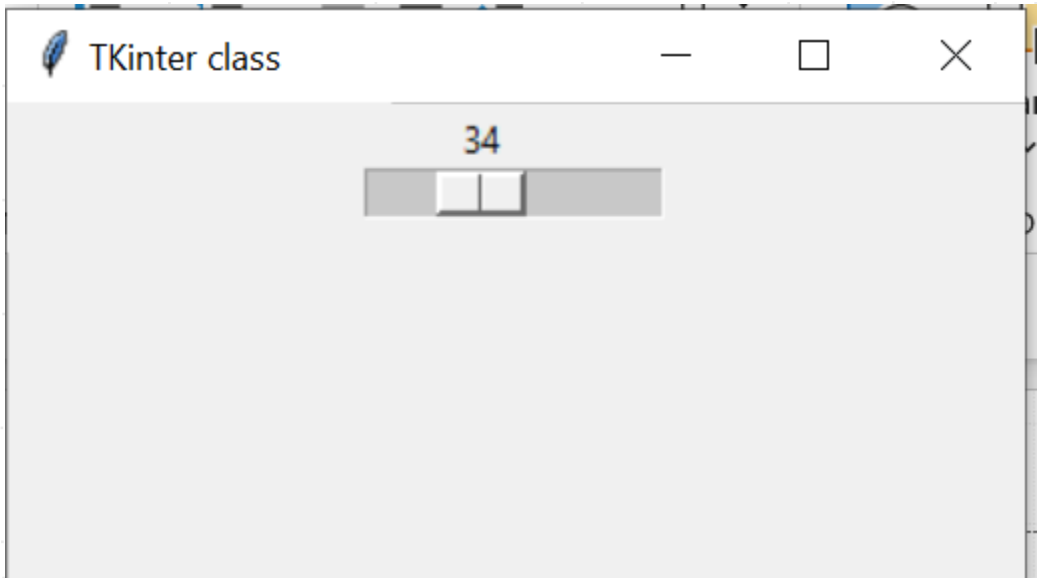
Spinbox

```
>>>window = Tk()
window.title("TKinter class")
>>> def spinbox_used():
print(spinbox.get())
>>> spinbox = Spinbox(from_=0, to=100, width=10, command=spinbox_used)
spinbox.focus()
spinbox.pack()
window.mainloop()
```



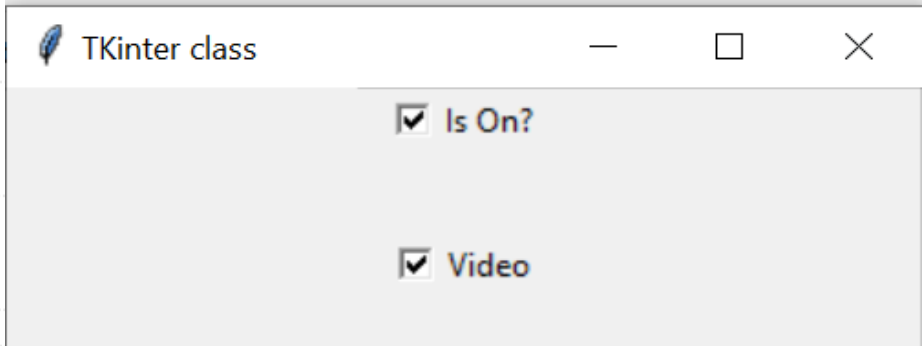
Scale

```
>>>window = Tk()
window.title("TKinter class")
>>> def scale_used(value):
 print(value)
 print(scale.get())
>>> scale = Scale(from_=0, to=100, command=scale_used, orient=HORIZONTAL)
scale.pack()
window.mainloop()
```



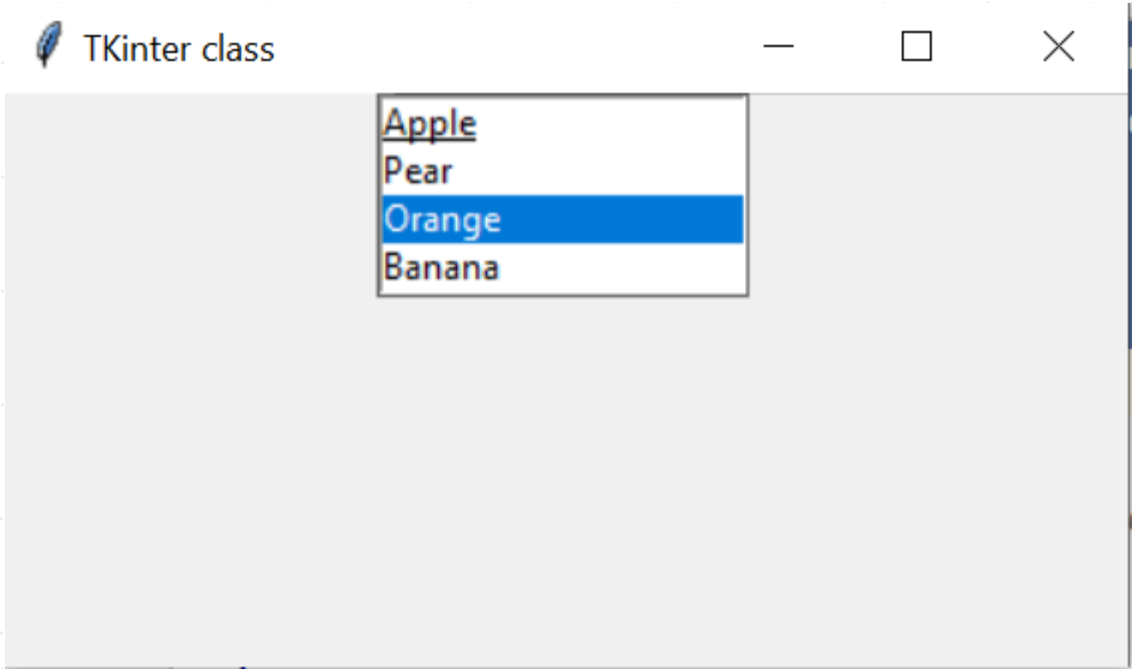
Checkbutton

```
>>>window = Tk()
window.title("TKinter class")
>>> def checkbutton_used():
 print(checked_state.get())
>>> checked_state = IntVar()
checkbutton = Checkbutton(text="Is On?",variable=checked_state,command=checkbutton_used)
checked_state.get()
checkbutton.pack()
checkVar1 = IntVar()
c1 = Checkbutton(window, text="Video", variable=checkVar1,onvalue=1, offvalue=0, height=5, width=10)
c1.pack()
window.mainloop()
```



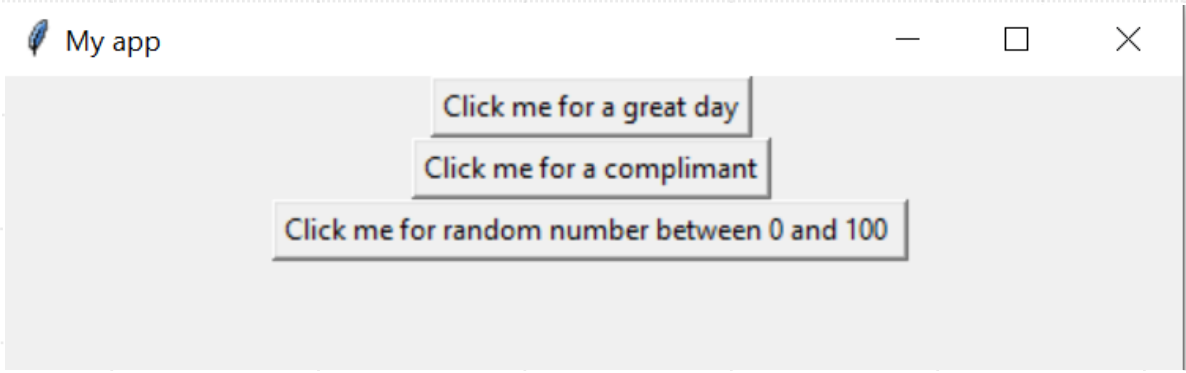
Listbox

```
>>>window = Tk()
window.title("TKinter class")
>>> def listbox_used(event):
 print(listbox.get(listbox.curselection()))
fruits = ["Apple", "Pear", "Orange", "Banana"]
listbox = Listbox(height=len(fruits))
>>> for item in fruits:
 listbox.insert(fruits.index(item), item)
listbox.bind("<<ListboxSelect>>", listbox_used)
listbox.pack()
listbox.focus()
listbox.select_set(2)
window.mainloop()
```



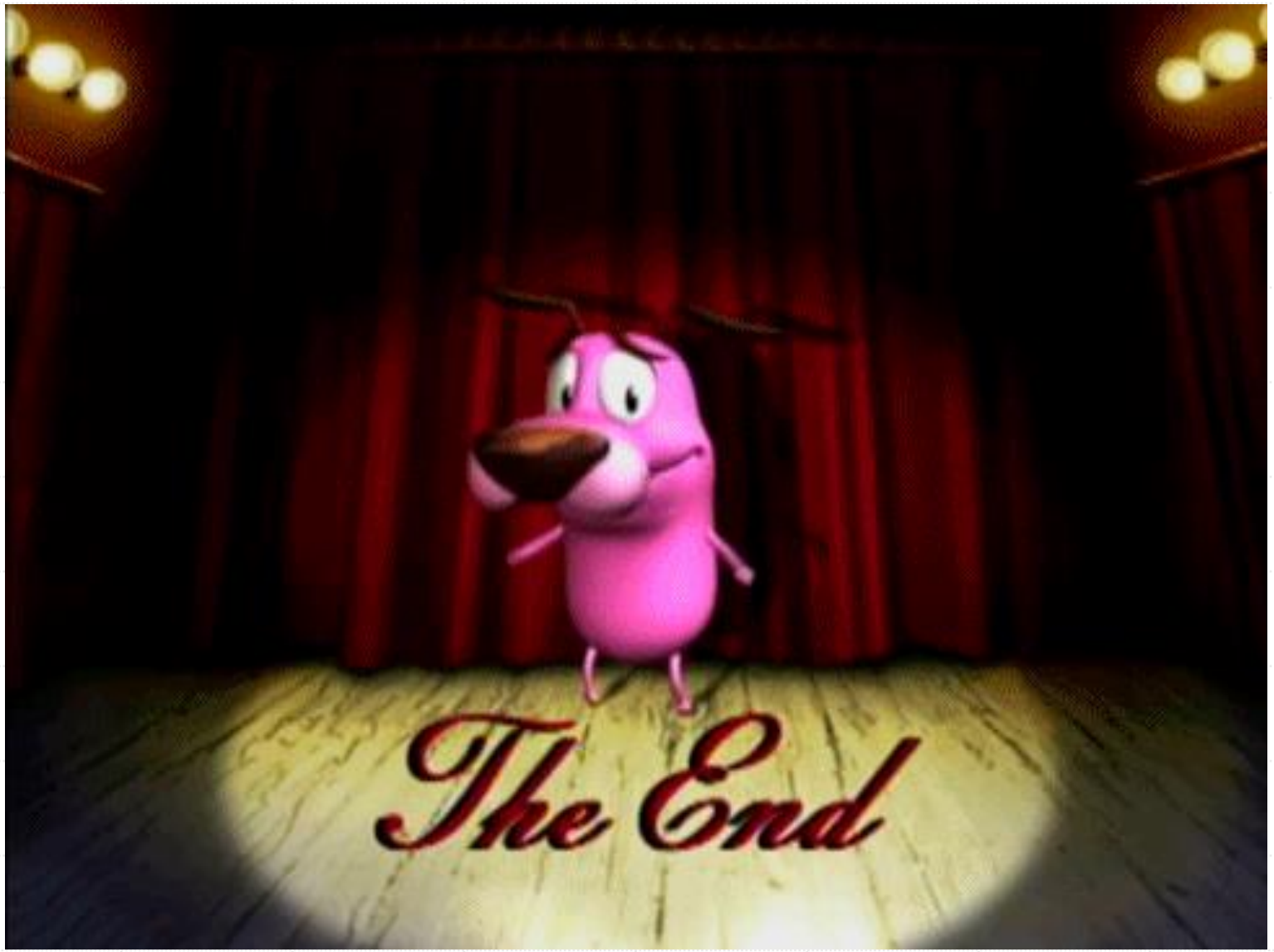
Window and configurations

```
>>>import random as rd
>>> window = Tk()
window.title("My app")
window.minsize(width=500, height=500)
button = Button(text="Click me for a great day", command=lambda: print("It's a great day to be alive"))
button.pack(side="top")
button = Button(text="Click me for a complimant" , command=lambda: print("You're amazing!!!"))
button.pack(side="top")
button = Button(text="Click me for random number between 0 and 100 "
                , command=lambda: print(rd.randint(0,100)))
button.pack(side="top")
window.mainloop()
```



Questions?

Hands On


The End