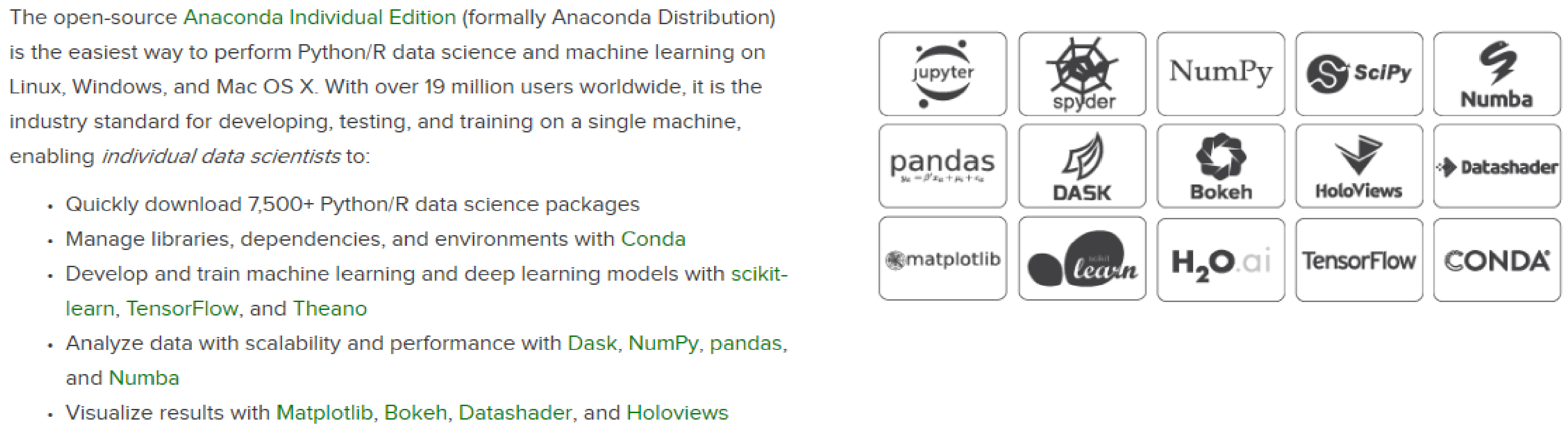
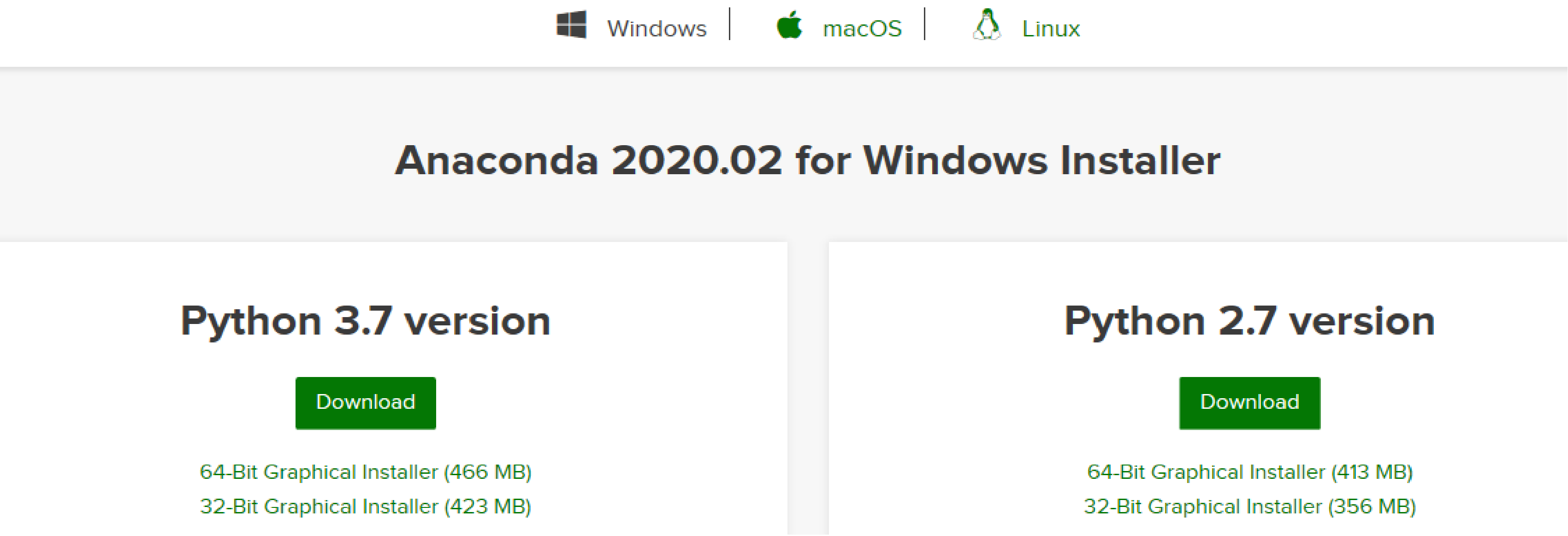
Installing Python on Windows



The open-source Anaconda Individual Edition (formally Anaconda Distribution) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 19 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 7,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews



Anaconda 2020.02 for Windows Installer

Python 3.7 version

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (423 MB)

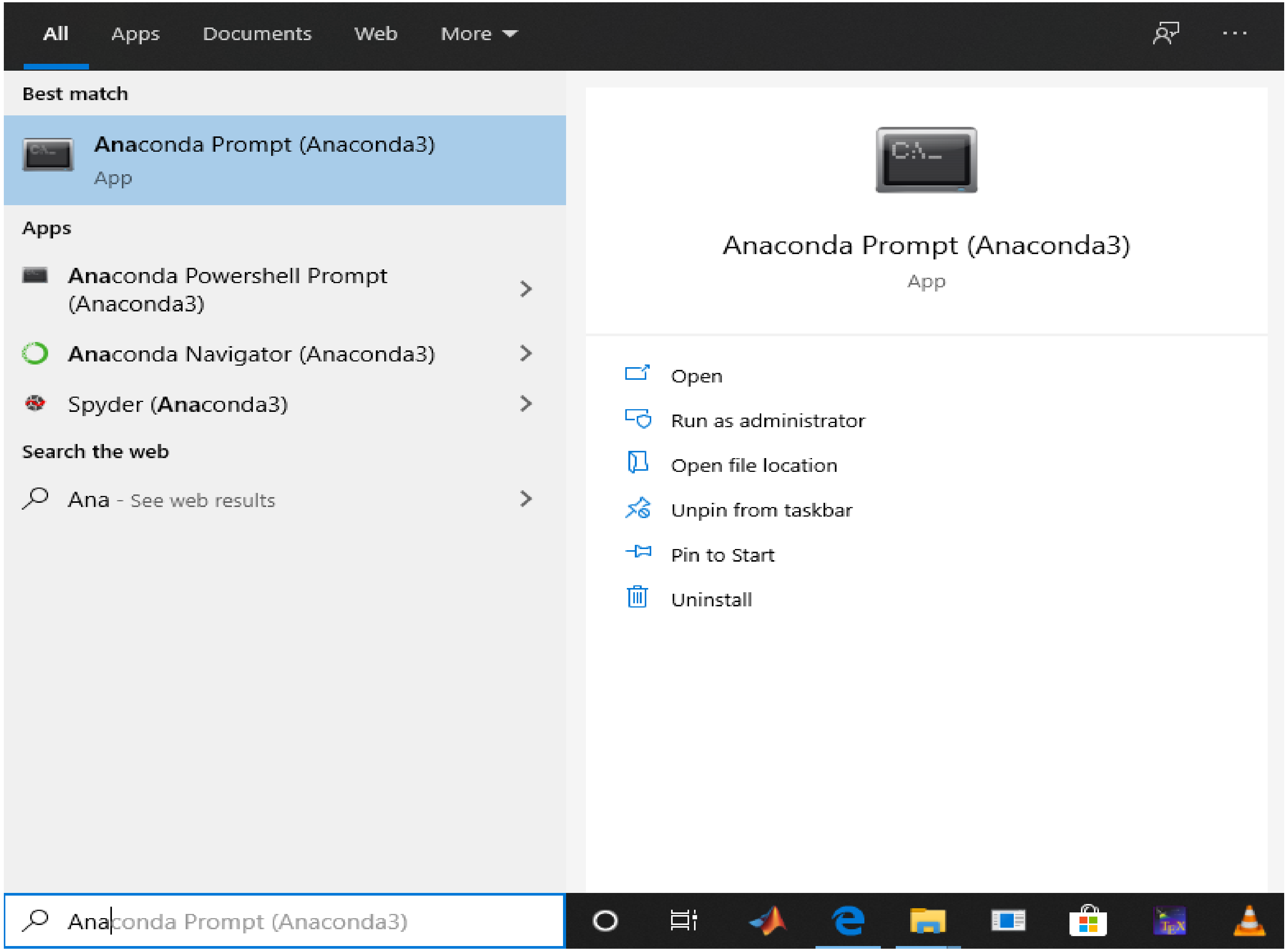
Python 2.7 version

64-Bit Graphical Installer (413 MB)

32-Bit Graphical Installer (356 MB)

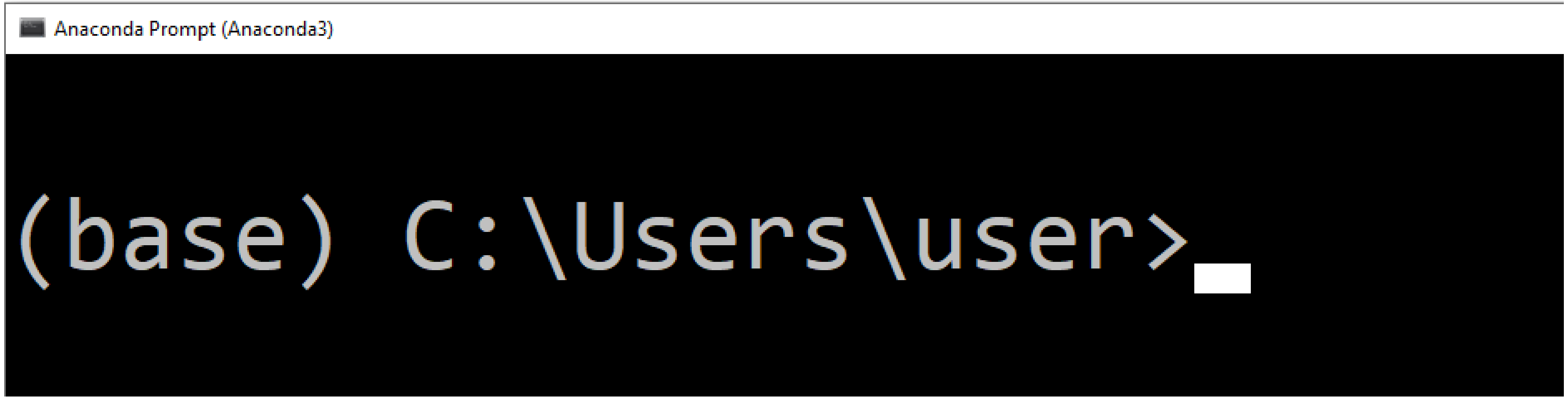
https://www.anaconda.com/distribution/

Launch Jupyter

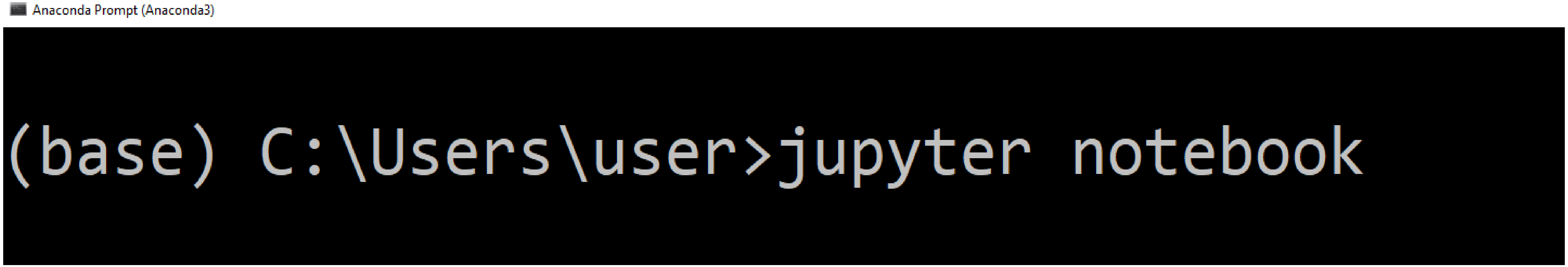


Anaconda Prompt (Anaconda3)

Launch Jupyter

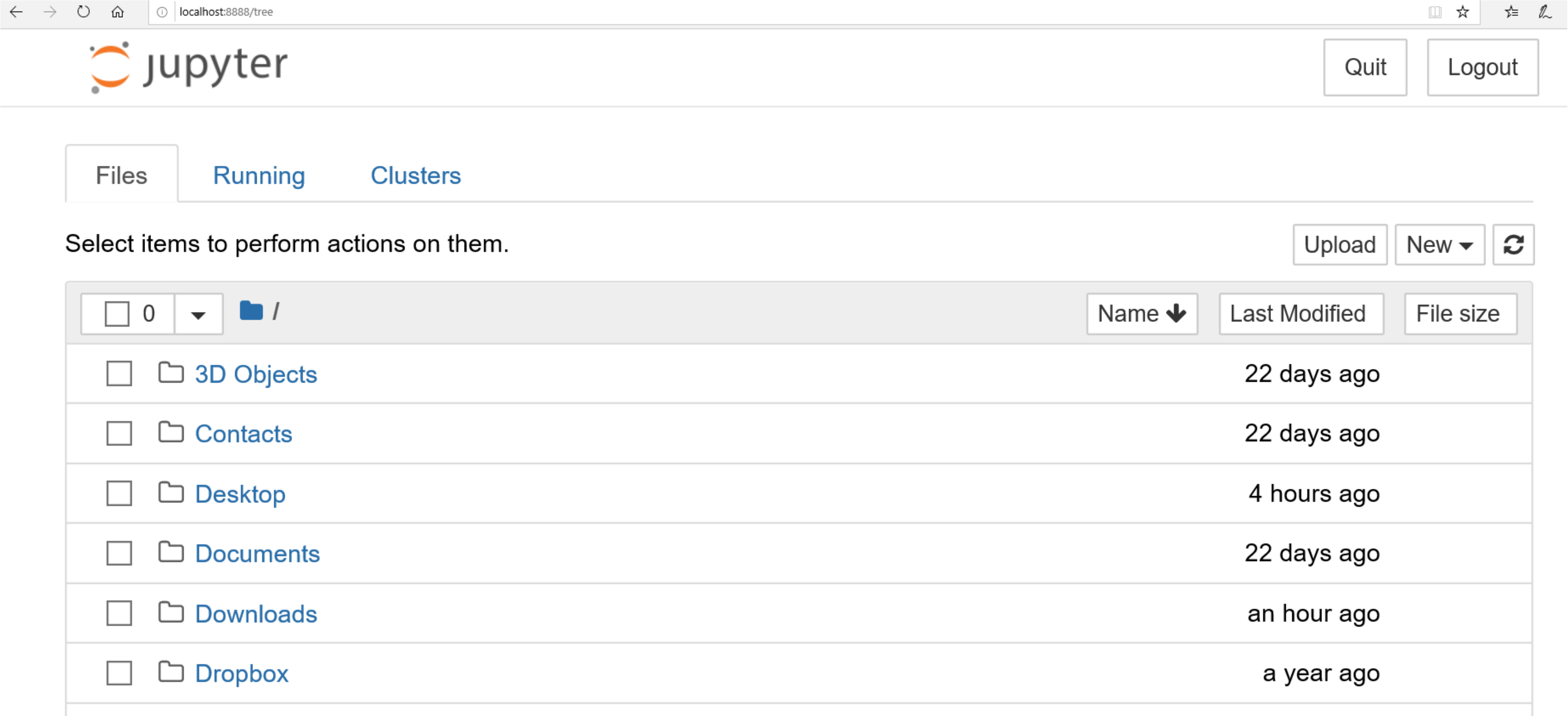


```
(base) C:\Users\user>_
```



```
(base) C:\Users\user>jupyter notebook
```

Launch Jupyter

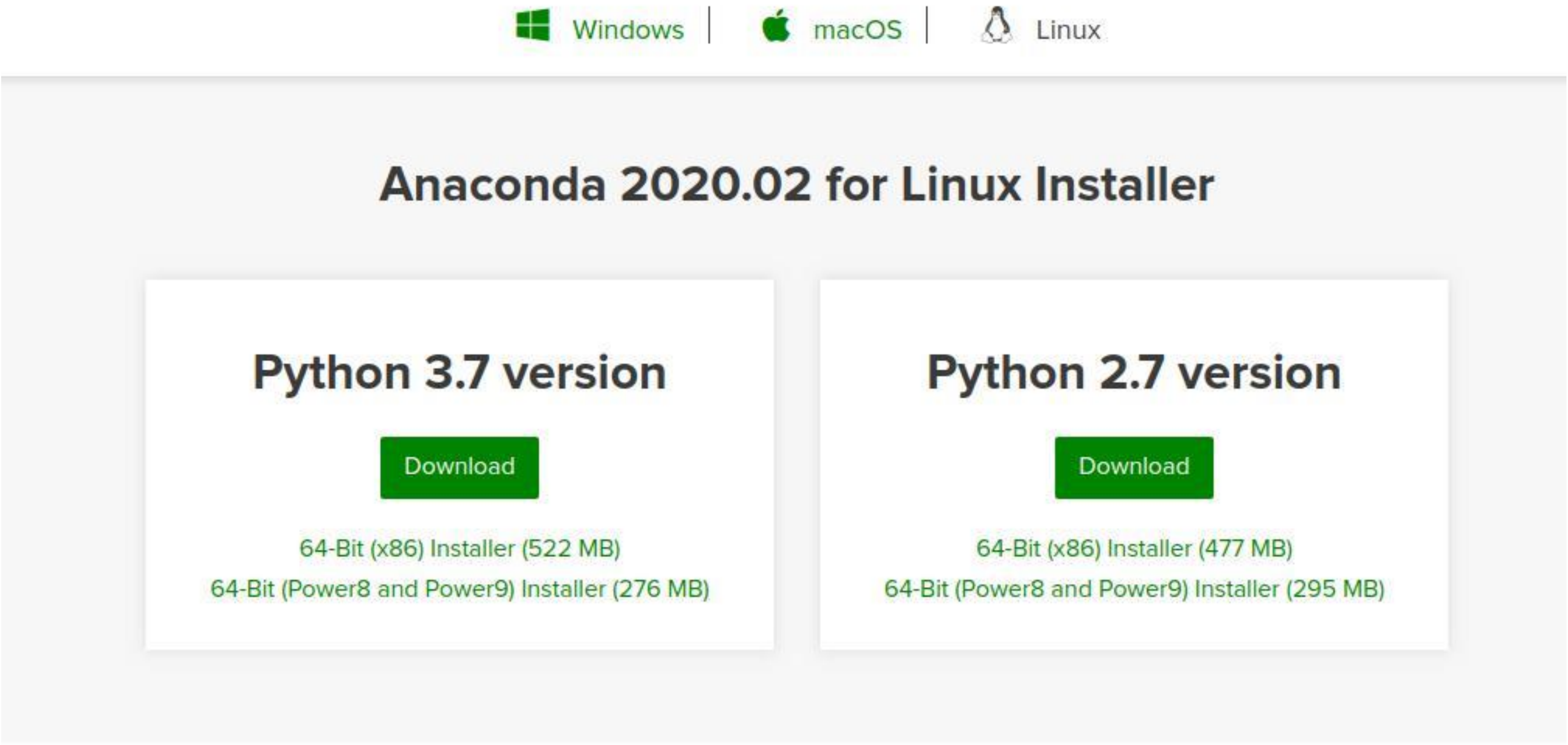


localhost:8888/tree

Select items to perform actions on them.

	Name	Last Modified	File size
□	3D Objects	22 days ago	
□	Contacts	22 days ago	
□	Desktop	4 hours ago	
□	Documents	22 days ago	
□	Downloads	an hour ago	
□	Dropbox	a year ago	

Installing Python on Ubuntu



Anaconda 2020.02 for Linux Installer

Python 3.7 version

64-Bit (x86) Installer (522 MB)

64-Bit (Power8 and Power9) Installer (276 MB)

Python 2.7 version

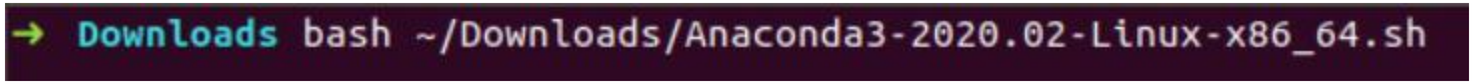
64-Bit (x86) Installer (477 MB)

64-Bit (Power8 and Power9) Installer (295 MB)

https://www.anaconda.com/distribution/#linux

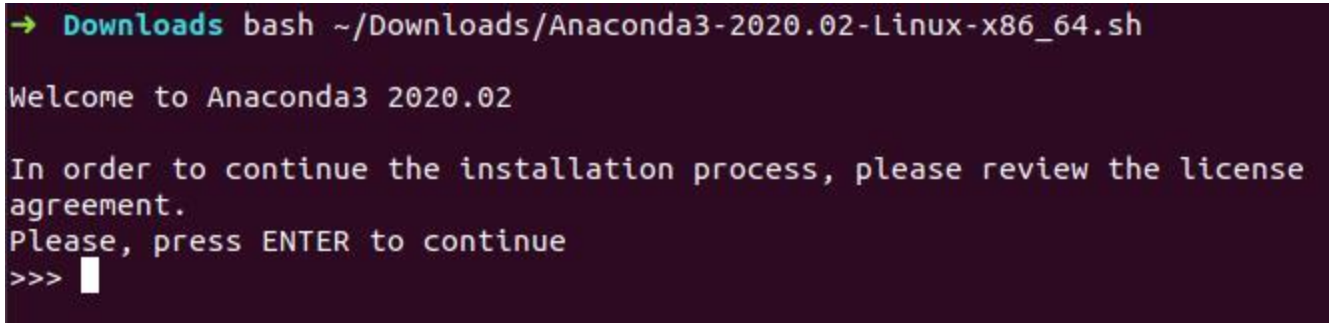
Installing Python on Ubuntu

1. Open a terminal and Enter the following to install Anaconda for Python 3.7:



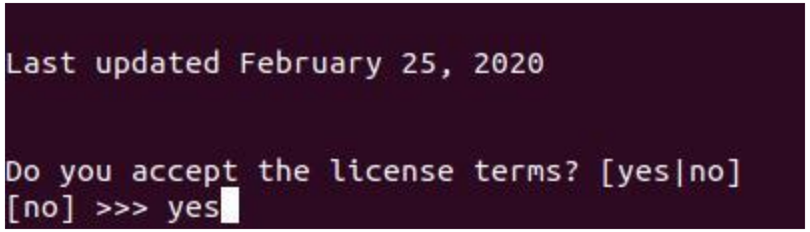
```
➜ Downloads bash ~/Downloads/Anaconda3-2020.02-Linux-x86_64.sh
```

1. The installer prompts “In order to continue the installation process, please review the license agreement.” Click Enter to view license terms.



```
➜ Downloads bash ~/Downloads/Anaconda3-2020.02-Linux-x86_64.sh

Welcome to Anaconda3 2020.02

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> 
```

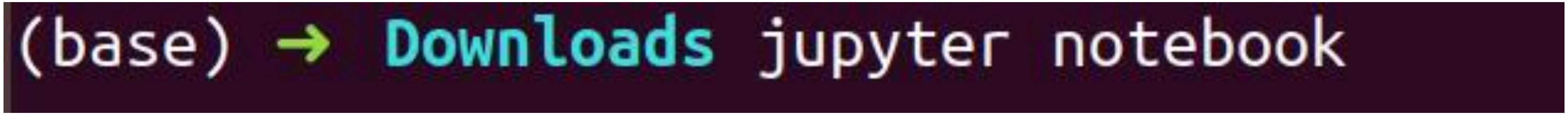
1. Scroll to the bottom of the license terms and enter “Yes” to agree.



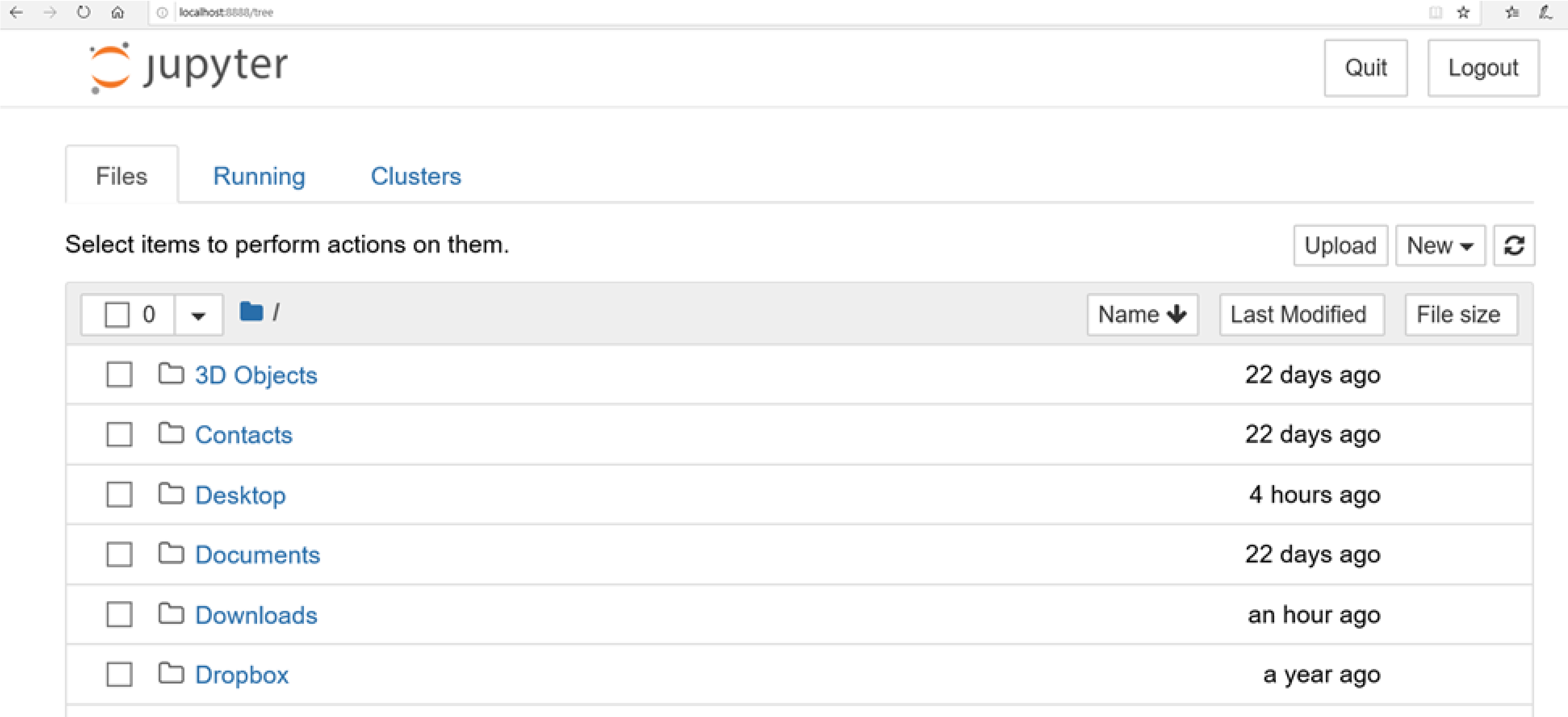
```
Last updated February 25, 2020


Do you accept the license terms? [yes|no]
[no] >>> yes
```

1. The installer finishes and displays “Thank you for installing Anaconda!”

Launch Jupyter On Ubuntu



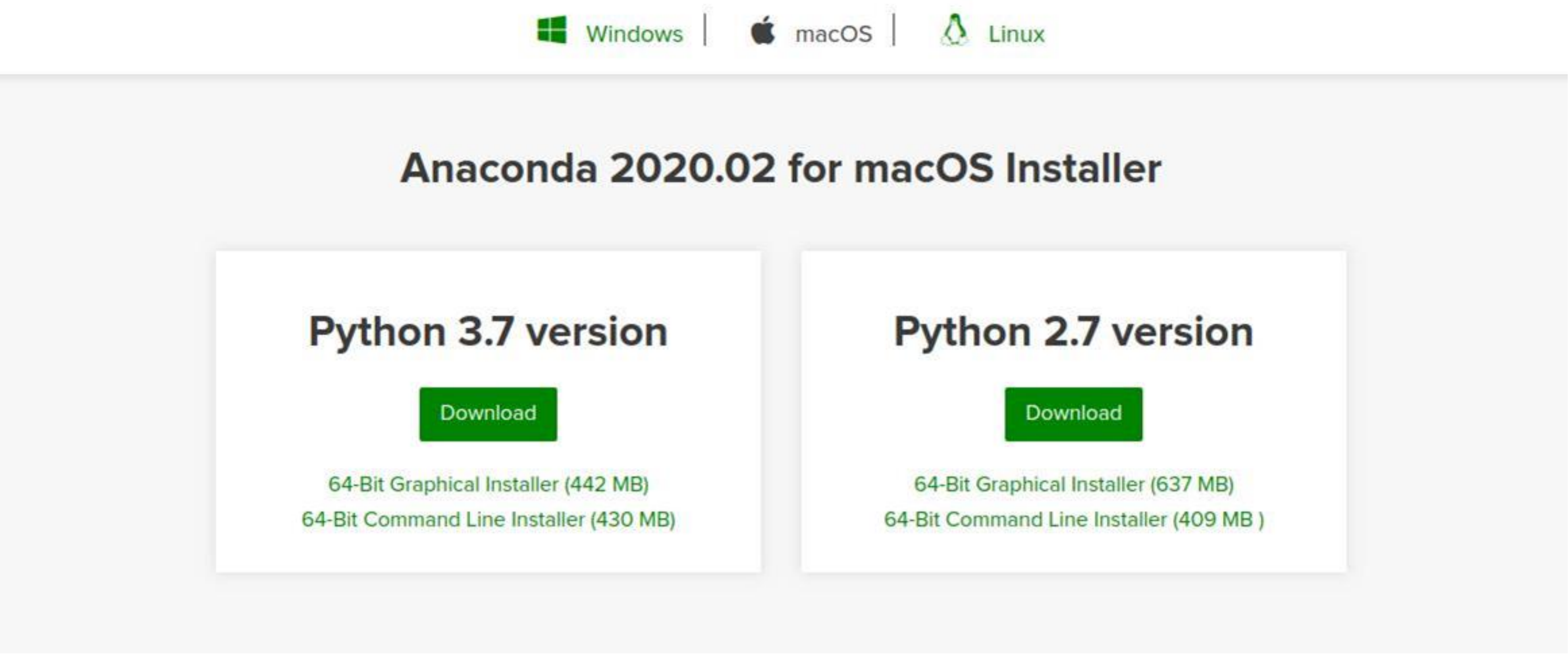
```
(base) → Downloads jupyter notebook
```



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	22 days ago	
Contacts	22 days ago	
Desktop	4 hours ago	
Documents	22 days ago	
Downloads	an hour ago	
Dropbox	a year ago	

Installing Python on macOS



Anaconda 2020.02 for macOS Installer

Python 3.7 version

64-Bit Graphical Installer (442 MB)

64-Bit Command Line Installer (430 MB)

Python 2.7 version

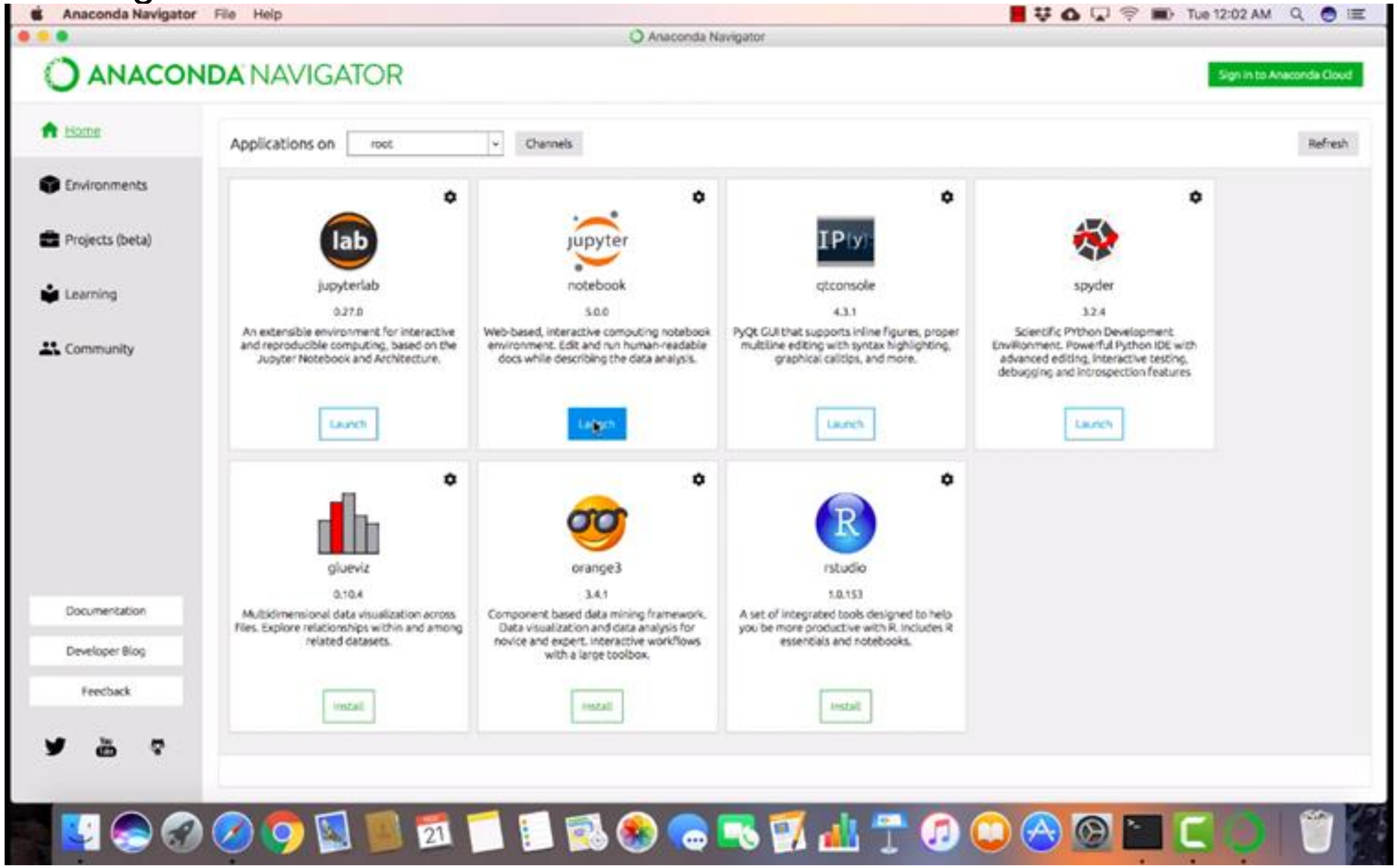
64-Bit Graphical Installer (637 MB)

64-Bit Command Line Installer (409 MB)

https://www.anaconda.com/distribution/#macos

Launch Jupyter On macOS

- You'll find Anaconda Navigator in Launchpad (and also in the Applications folder). Drag it to the Dock if you want to have it readily available. Then, click the Launch button below the notebook icon on the Navigator



jupyterlab

0.27.0

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

notebook

5.0.0

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

qtconsole

4.3.1

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

spyder

3.2.4

Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz

0.10.4

Multidimensional data visualization across files. Explore relationships within and among related datasets.

orange3

3.4.1

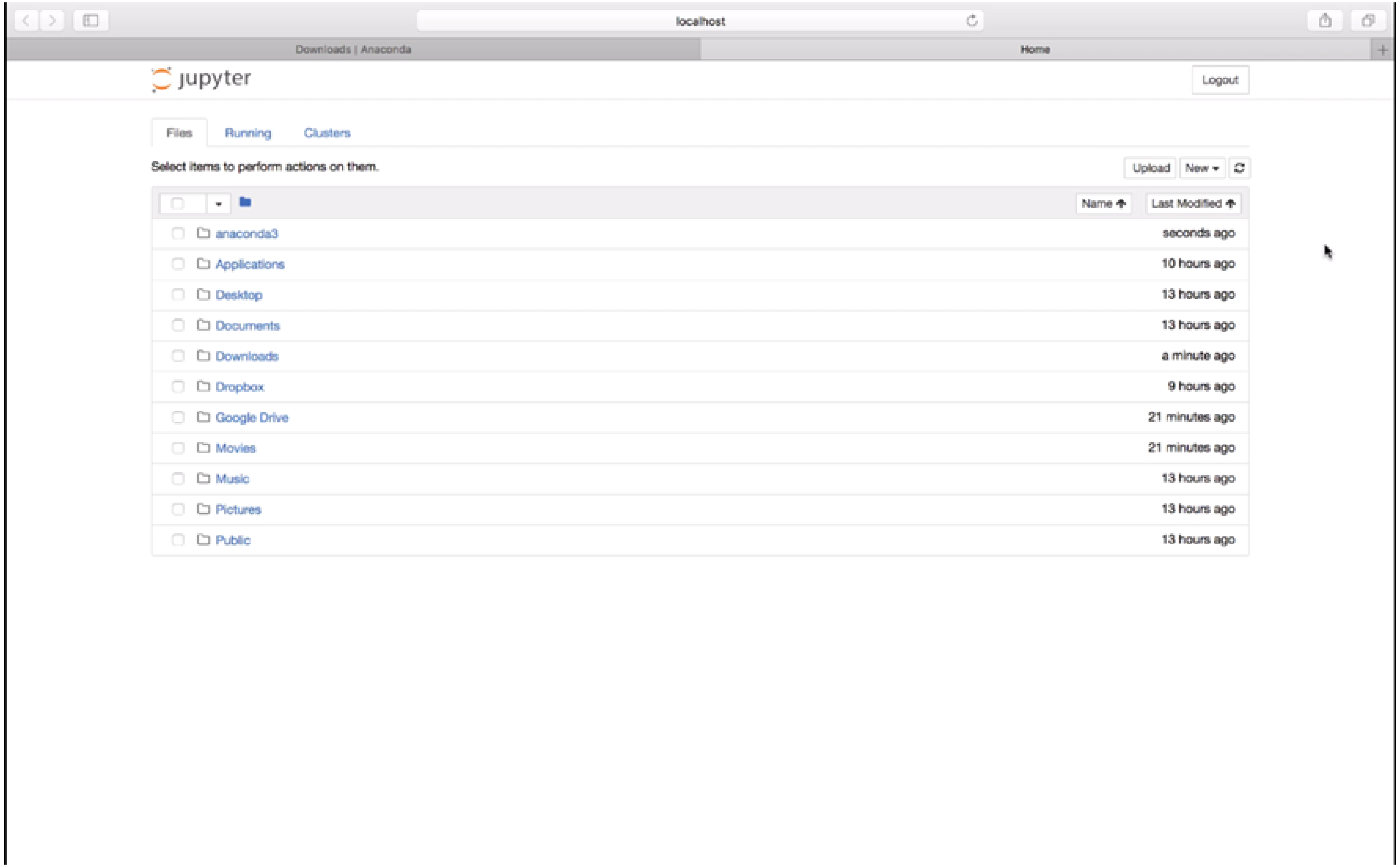
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

rstudio

1.0.153

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Launch Jupyter On macOS



Select items to perform actions on them.

Name	Last Modified
□ anaconda3	seconds ago
□ Applications	10 hours ago
□ Desktop	13 hours ago
□ Documents	13 hours ago
□ Downloads	a minute ago
□ Dropbox	9 hours ago
□ Google Drive	21 minutes ago
□ Movies	21 minutes ago
□ Music	13 hours ago
□ Pictures	13 hours ago
□ Public	13 hours ago

Variables and operators

```
x = 2          # int          print(type(a))     # <class 'int’>
y = 5          # int          Print(type(b))     # <class ‘float'>
xy = 7.2       # float

a,b = 4,5.0    # multiple assignment
```

```
del x          # deletes x
print(x)       # error
```

Variables and operators

Symbol	Task Performed
+	Addition
-	Subtraction
/	division
%	mod
*	multiplication
//	floor division
**	to the power of

Can a variable name start with a digit i.e. 3x ?

Type bool and Comparisons

Symbol	Task Performed
==	True, if it is equal
!=	True, if not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Type bool and Comparisons

print((not(2!=3) and True) or (False and True))

Some Useful functions

- **round( )** function rounds the input value to a specified number of places or to the nearest integer.

```
print (round(5.6231) )
print (round(4.55842, 3))
```

Some Useful functions

- **divmod(x,y)** outputs the quotient and the remainder in a tuple (We will see what a tuple is)

```
divmod(27,5)
```

Some Useful functions

- **isinstance( )** returns True, if the first argument is an instance of that class. Multiple classes can also be checked at once.

```
print(isinstance(1, int))
print(isinstance(1.0,int))
print(isinstance(1.0,(int,float)))
```

Some Useful functions

- **pow(x,y,z)** x raise to the power y and remainder by z

Some Useful functions

- **Input()** a = input("Enter something")

Control Flow

```
a = int(input())
b = int(input())
```

```
Task:
  print only the bigger number
```

Control Flow (If condition)

```
a = int(input())
b = int(input())
if b > a:
  print("b is greater than a")
```

Control Flow (If-Else)

```
a = int(input())
b = int(input())
if b > a:
  print("b is greater than a")
else:
  print(“b is not greater than a")
```

Control Flow (If-Elif-Else)

```
a = float(input(“Enter first number : ”))
b = float(input(“Enter second number : ”))
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Control Flow (short hand if)

```
a = float(input(“Enter first number : ”))
b = float(input(“Enter second number : ”))
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a = 9
b = 10
print(“A") if a > b else print("=") if a == b else print("B")
```

Control Flow (Nested If)

```
x = int(input(“Enter a number : “))

if x > 10:
    print(“Your Number is above ten, ")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

Control Flow (Indentation!)

```
x = int(input(“Enter a number : “))

if x > 10:
 print(“Your Number is above ten, ")
 if x > 20:
  print("and also above 20!")
 else:
  print("but not above 20.")
```

```
x = int(input(“Enter a number : “))

if x > 10:
 print(“Your Number is above ten, ")
 if x > 20:
  print("and also above 20!")
else:
 print(“Not above 10.")
```

Control Flow (Loops)

```
n = input(‘Max iterations :’)
i = 1
while i < n:
  print(i)
  i += 1
print(‘done’)
```

Control Flow (Loops)

```
n = input(‘Max iterations :’)
i = 1
while i < n:
  if i%2 == 0:
     print(i)
  else
     pass
  i += 1
print(‘done’)
```

Control Flow (Loops: break, continue)

```
i = 1
while True:
    if i%17 == 0:
        print(‘break’)
        break
    else
        i += 1
        continue
    print(‘I am inside the loop’)
print(‘done’)
```

Control Flow (for Loop)

```
L = []
for i in range(10):
    print(i+1)
    L.append(i**2)
```

Control Flow (else in for Loops)

```
S = {"apple", 4.9, "cherry“}
for x in S:
  print(x)
else
  print(‘Loop completes its iterations’)
```

Control Flow (Exploring a Dictionary)

```
D = {"apple“:44,"cherry“:”game”}
for x in D:
  print(x,D[x])
```

Functions

```
def printSuccess():
   print(“The task was successful")
   print(“Moving to the next task")
   print(“Send me the next task")
```

Functions(Doc string)

```
def printSuccess():
  “”” this function does nothing”””
  print(“The task was successful")
```

Functions(input arguments)

```
def printMessage(msg):
  print(msg)
```

Functions(input arguments)

```
def printMessage(msg1,msg2):
  print(msg1,msg2)
```

Functions(Order of input arguments)

```
def f(c2, c1, c3):
  print(c1,c2,c3)


f(c1 = “A", c2 = “B", c3 = “C")
```

Functions(variables inside)

```
def add(x,y):
  c = x+y  # I need value of c?
```

Functions(return statement)

```python
def add(x,y):
  return x+y
```

Functions(variable number of input arguments)

```
def add(*args):
  sum = 0
  for i in range(len(args)):
      sum+=args[i]
  return sum
```

add(3,4,5,6,7)

Functions(variable number of input arguments)

```
def f(**c):
  for x in c:
     print(x,c[x])


f(c1 = “A", c2 = “B", c3 = “C")
```

Functions(Default values)

```python
def f(sum = 0):
    print(sum)
```

Modules

D:/mymodules/myfuncs.py

```
def printMe(msg='No message was supplied’):
    print(msg)

def printList(L=[]):
    for x in L:
        print(x)
```

```
import sys
sys.path.append('D:/mymodules/')
import myfuncs as f
f.printMe(‘hellow’)
```

Practice Functions

String

“python” is same as ‘python’ # be careful and don’t mix ‘python” or “python’

```
s = “python is the best language for data science”
t = ‘in this course we are going to learn python. ‘
print(s + ' and, ' +t)
```

String(Multi line String)

```
mulitineString = """This is the first line of the string,
                    and here is the second line,
                    and here is the third and final line."""
```

String Cont…

Indexing and Slicing

Negative index

```
a = “Game of programming"
print(a[3:8])
print(a[-8:-3])

print(len(a))
print(len(a[3:8]))
```

String Cont...

```
a = "      A lot OF Spaces at The     beGinning and end     "
b = a.strip()
print(b.lower())
print(b.upper())
print(a.replace(" ", ","))
L = "game,and,no".split(",") # returns List (We will see Lists later on)
```

String Cont…

```
a = "      A lot OF Spaces at The     beGinning and end     "
b = a.strip()
print(b.lower())
print(b.upper())
print(a.replace(" ", ","))

L = "game,and,no".split(",") # returns List (We will see Lists later on)

print(“at”  in “asdfaatdea” )

print(“at”  not in “asdfaatdea” )

"We are learning \"Strings\" here."
‘We are learning "Strings" here.’

 print(“c:\drive\name” )

 print(r”c:\drive\name” )
```

Data Structures

•List	Ordered,	changeable,	duplicates
•Tuple	Ordered,	unchangeable,	duplicates
•Set	Unordered,	addable/removable	no duplicates
•Dictionary	Unordered,	changeable,	no duplicate

List	Tuple	Set	Dictionary
L= [12, "banana", 5.3]	T= (12, "banana", 5.3)	S= {12, "banana", 5.3}	D={“Val”:12,”name”:”Ban”}
L[1]	T[2]	X in S	D[“Val”]

L = L + [“game”]
L[2] = “orange”
	Immutable
	T3 = T1+T2
		S.add(“new Item”)
		S.update({“more”,”items”})
			D[“Val”] = newValue
			D[“newkey”] = “newVal”

del L[1] del L	Immutable del T	S.Remove(“banana”) del S	del D[“Val”] del D
L2 = L.copy()	T2 = T	S2 = S.copy()	D2 = D.copy()
….	….	….	….

Numpy

Why Numpy?

Quick Answer: Numpy is Faster

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])
b = np.array((1, 6, 3, 4, 9))
print(a)
print(b)
```

Numpy(Dimensions)

```
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.ndim)

b = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(b.ndim)

print(b.shape[0],b.shape[1],b.shape[2])

print(b[1,0,2])
```

Numpy(np.arrange,reshape,random)

- np.arange
- np.random.permutation
- np.reshape

Numpy(Slicing)

```
A[start:end:step]

a[1:5] # index 1 till 5 but not 5

a[:5] # index 0 till 5 but not 5

a[:5] # index 0 till 5 but not 5

a[2:] # index 2 till end including last element

a[::-1] # from end till start (reverse the array)

a[::2] # from start till end every other element

a[::2,:] ?
```

Numpy(More Indexing)

```
A[index_array]

a[[1,4,6]] # index 1, 4 and 6 elements

a[[True,True,False,False,True,True,True,False]]
```

Assuming array has 8 elements, the above returns all the elements corresponding to True index

```
a[a<8]

a[a<8 & a>4] # difference between(and,&)?
a[a<8 and a>4] ?
```

Copy vs View

Numpy(Broadcasting)

A = A+5

Numpy(hstack,vstack,sort(axis=0)

```
np.hstack
np.vstack
np.sort
```

Numpy(Seed: ufuncs)

```
b = np.random.rand(1000000)
%timeit sum(b)
%timeit np.sum(b)
```

```
%%timeit
s = 0
for x in b:
    s+=x
```

Pandas

```
data = pd.Series([0.25, 0.5, 0.75, 1.0],index=['a', 'b', 'c', 'd'])

data.values


data.index
```

Pandas(Series)

```
grades_dict = {‘A’: 4,‘A-’: 3.5,‘B’: 3,‘B-’: 2.5,‘B’: 2}
grades = pd.Series(grades_dict)

marks_dict = {‘A’: 85,‘A-’: 80,‘B’: 75,‘B-’: 70,‘B’: 65}
marks = pd.Series(marks_dict)
```

Pandas(DataFrame)

```
grades_dict = {‘A’: 4,‘A-’: 3.5,‘B’: 3,‘B-’: 2.5,‘B’: 2}
grades = pd.Series(grades_dict)

marks_dict = {‘A’: 85,‘A-’: 80,‘B’: 75,‘B-’: 70,‘B’: 65}
marks = pd.Series(marks_dict)

rs = pd.DataFrame({‘grades’:grades,’marks’:marks})
```

Pandas(values)

```
grades_dict = {‘A’: 4,‘A-’: 3.5,‘B’: 3,‘B-’: 2.5,‘B’: 2}
grades = pd.Series(grades_dict)

marks_dict = {‘A’: 85,‘A-’: 80,‘B’: 75,‘B-’: 70,‘B’: 65}
marks = pd.Series(marks_dict)

rs = pd.DataFrame({‘grades’:grades,’marks’:marks})
rs['percentage'] = rs['marks']/100
rs.index
rs.columns
rs.values[2,:]
```

Pandas(indexing)

```
grades_dict = {‘A’: 4,‘A-’: 3.5,‘B’: 3,‘B-’: 2.5,‘B’: 2}
grades = pd.Series(grades_dict)

marks_dict = {‘A’: 85,‘A-’: 80,‘B’: 75,‘B-’: 70,‘B’: 65}
marks = pd.Series(marks_dict)

rs = pd.DataFrame({‘grades’:grades,’marks’:marks})
```

```
rs = rs[rs[‘marks’]>60]
```

Pandas(NaN)

```
pd.DataFrame([{'a': 1, 'b': 2}, {'b': 3, 'c': 4}])
```

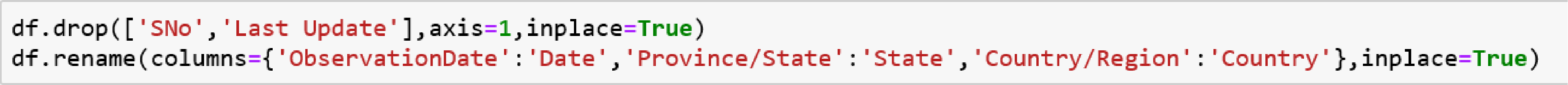
```
     a    b    c
0  1.0    2  NaN
1  NaN    3  4.0
```

Pandas (Indexing)

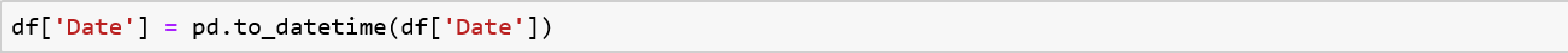
```
data = pd.Series(['a', 'b', 'c'], index=[1, 3, 5])

data[1] # explicit index  , use loc instead
data[1:3] # implicit index , use iloc instead

rs.iloc[2,3]
```

Pandas (csv files)

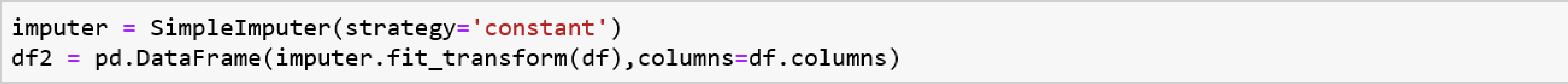
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
df = pd.read_csv('covid_19_data.csv’)



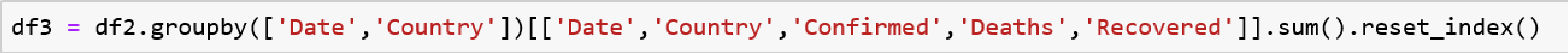
```
df.drop(['SNo','Last Update'],axis=1,inplace=True)
df.rename(columns={'ObservationDate':'Date','Province/State':'State','Country/Region':'Country'},inplace=True)
```



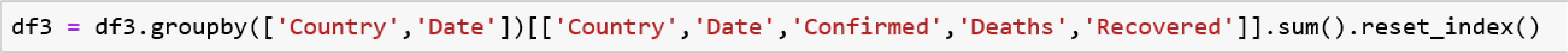
```
df['Date'] = pd.to_datetime(df['Date'])
```



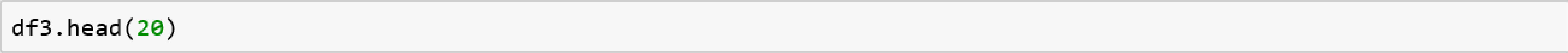
```
imputer = SimpleImputer(strategy='constant')
df2 = pd.DataFrame(imputer.fit_transform(df),columns=df.columns)
```



```
df3 = df2.groupby(['Date','Country'])[['Date','Country','Confirmed','Deaths','Recovered']].sum().reset_index()
```



```
df3 = df3.groupby(['Country','Date'])[['Country','Date','Confirmed','Deaths','Recovered']].sum().reset_index()
```



```
df3.head(20)
```

Matplotlib

```
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
plt.plot(x, np.sin(x));
```

Matplotlib

```
plt.plot(x, np.sin(x - 0), color='blue’)
plt.plot(x, np.sin(x - 1), color='g’)
plt.plot(x, np.sin(x - 2), color=‘r’)
```

Matplotlib

```
plt.plot(x, x + 0, '-g') # solid green
plt.plot(x, x + 1, '--c') # dashed cyan
plt.plot(x, x + 2, '-.k') # dashdot black
plt.plot(x, x + 3, ':r'); # dotted red
```