Installing Python

<https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/python.html>

<https://www.python.org/downloads/>

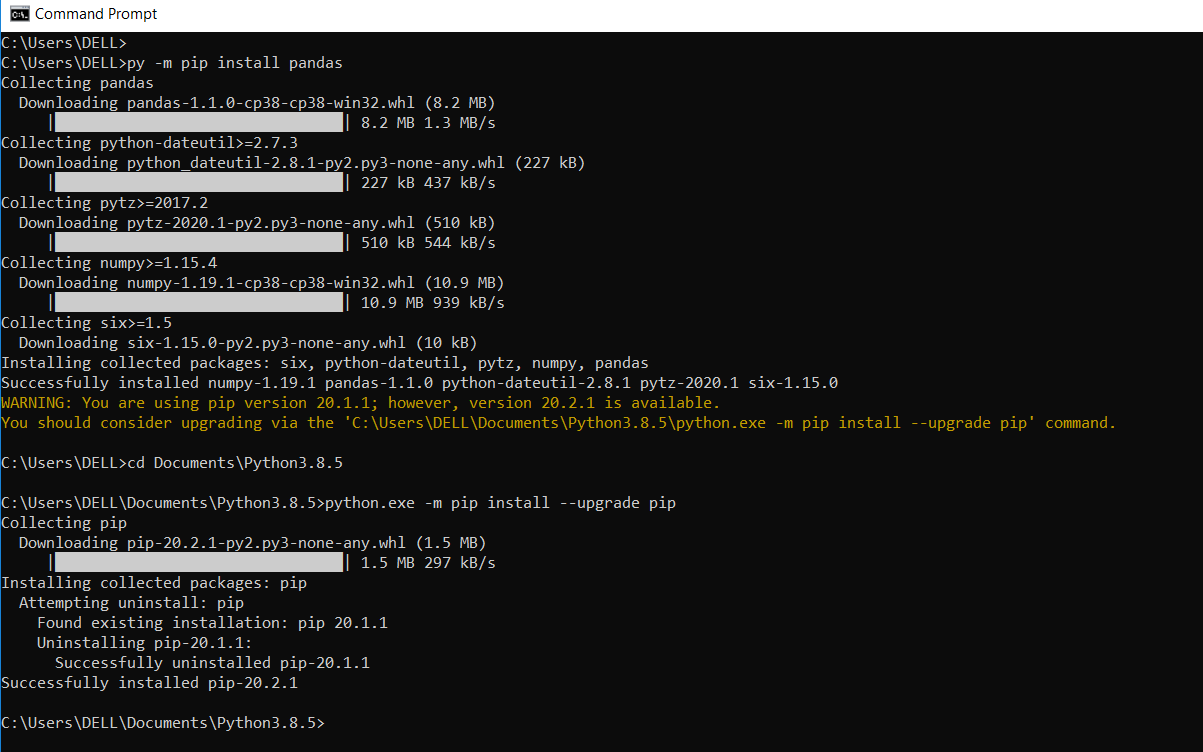
Python Tutorial

<https://docs.python.org/3.8/tutorial/index.html>

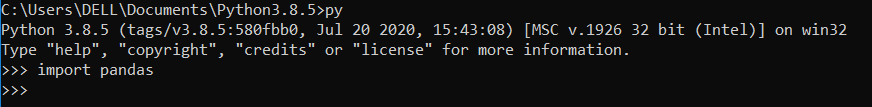
Python Documentation 3.8

<https://docs.python.org/3.8/index.html>

installing Panda Libraries on Python – (C:\Users\DELL>py -m pip install pandas)



Verify if pandas installed correctly on python -



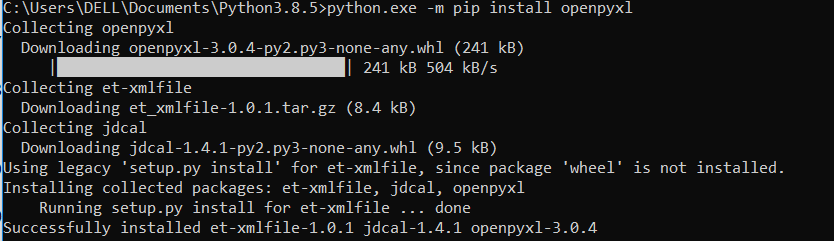
Download Text editor – VisualStudio Code

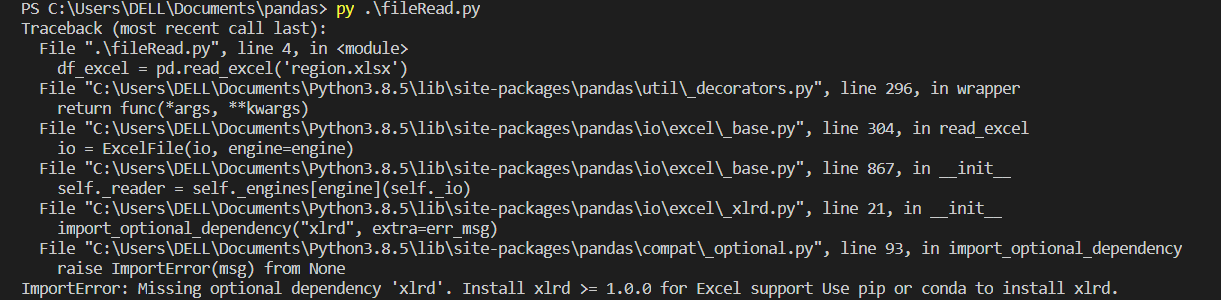
<https://code.visualstudio.com/download>

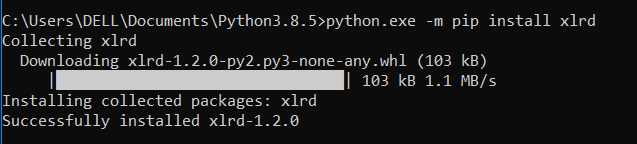
Module openpyxl – working with Excel not installed

<https://pythonexamples.org/modulenotfounderror-no-module-named-openpyxl/>

pip install openpyxl







Reading from Excel through python, pandas, openpyxl

import pandas as pd

from openpyxl  import Workbook

df\_excel = pd.read\_excel('regions.xlsx')

df\_csv = pd.read\_csv('Names.csv')

df\_txt = pd.read\_csv('data.txt')

print(df\_excel)

PS C:\Users\DELL\Documents\pandas> py .\fileRead.py

Region Units Sales Export

0 South 54 332 100

1 North 20 110 50

2 East 36 224 85

3 West 60 400 110

4 West 50 226 65

5 North 84 470 150

PS C:\Users\DELL\Documents\pandas>

Error debugging –

<https://pythonexamples.org/modulenotfounderror-no-module-named-openpyxl/>

import pandas as pd

from openpyxl  import Workbook

df\_excel = pd.read\_excel('regions.xlsx')

df\_csv = pd.read\_csv('Names.csv',header=None)

df\_txt = pd.read\_csv('data.txt', delimiter='\t')

df\_csv.columns = ['First','Last','Address','City','State','Area Code','Salary']

df\_csv.to\_excel('modified.xlsx')

#print(df\_csv)

df = pd.read\_csv('Names.csv', header=None)

df.columns = ['First','Last','Address','City','State','Area Code','Salary']

#print(df.columns)

#print(df['Last'])

#print(df[['State','Area Code']])

#print(df['First'][0:3])

#print(df.iloc[1])

#print(df)

#print(df.iloc[5,3])

wanted\_values = df[['First','Last','State']]

stored = wanted\_values.to\_excel('State\_location.xlsx',index=None)

import pandas as pd

from openpyxl.workbook import Workbook

df = pd.read\_csv('Names.csv', header=None)

df.columns = ['First','Last','Address','City','State','Area Code','Income']

#print(df)

#print(df.loc[df['City'] == 'Riverside'])

#print(df.loc[(df['City'] == 'Riverside') & (df['First'] == 'John')])

df['Tax %'] = df['Income'].apply(lambda x: .15 if 10000 < x < 40000 else .2 if 40000 < x < 80000 else .25)

#print(df)

df['Taxes Owed'] = df['Income'] \* df['Tax %']

print (df['Taxes Owed'])

to\_drop = ['Area Code', 'First', 'Address']

df.drop(columns=to\_drop, inplace=True)

#print(df)

df['Test col']= False

df.loc[df['Income'] < 60000, 'Test col'] = True

#print(df)

#print(df.groupby(['Test col']).mean())

print(df.groupby(['Test col']).mean().sort\_values('Income'))

groupby

sort\_values

mean

to\_drop

apply

lambda

working in Jupyter Notebook

==============================

1. print("Hello!")
2. for i in range(10): print(i)
3. 4\*sum(pow(-1,k)/(2\*k+1) for k in range(10000))
4. import matplotlib.pyplot as plt
5. import numpy as np
6. import pandas as pd
7. greeting = "Hello Jupyter!"
8. greeting

'Hello Jupyter!'

1. words=greeting.split(" ")

word\_length = [len(w) for w in words]

plt.bar(words,word\_length)

plt.show()

2 modes of working

Command Mode – Blue Border – Used for action taken in the context of the notebook – outside the scope of any individual cell

Keyboard Shortcuts

A – insert cell above

B – insert cell below

d d – delete cell

x – cut

c – copy

v – paste

and

Edit mode – Green Border

import numpy as np

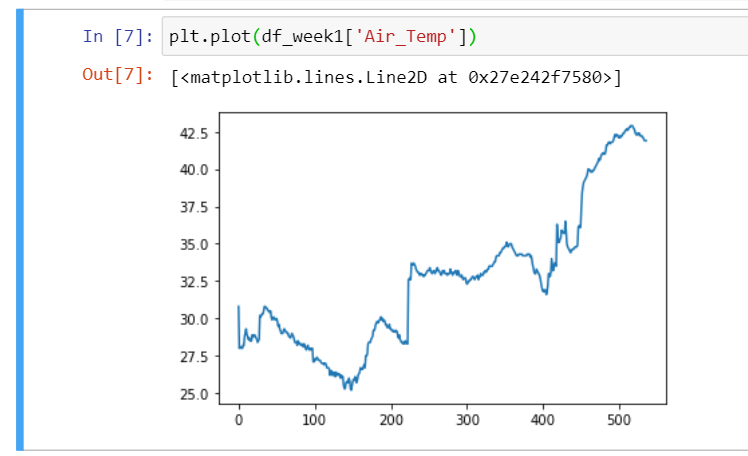
import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('../../inputs/Environmental\_Data\_Deep\_Moor\_2013.csv')

df\_week1 = df[df['date'].between('2013\_01\_01','2013\_01\_07')]

plt.plot(df\_week1['Air\_Temp'])



Ctrl + (UpArrow Key) – go to beginning of line

Ctrl + / - inserting comment

Ctrl + (Left Arrow Key/Right arrow Key) – one word move back and fwd

Ctrl + z – Undo

Ctrl +Shift + z – Redo

Ctrl + Backspace – delete upto beginning

17 May 2021

<https://www.datacamp.com/community/tutorials/python-excel-tutorial>

<https://pythonexamples.org/pandas-write-dataframe-to-excel-sheet/>

Example 2: Write DataFrame to a specific Excel Sheet

You can write the DataFrame to a specific Excel Sheet. The step by step process is:

1. Have your DataFrame ready.
2. Create an Excel Writer with the name of the desired output excel file.
3. Call to\_excel() function on the DataFrame with the writer and the name of the Excel Sheet passed as arguments.
4. Save the Excel file using save() method of Excel Writer.

15 Apr 2022

**Types in Python**

C:\Users\DELL\Downloads\\_\_0.Python.Leanings\\_PythonPractice>python

Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> t=()

>>> type(t)

<class 'tuple'>

>>> list=[12,133,1444]

>>> type(list)

<class 'list'>

>>> d={}

>>> type(d)

<class 'dict'>

>>> x=44

>>> type(x)

<class 'int'>

>>> y=33.44

>>> type(y)

<class 'float'>

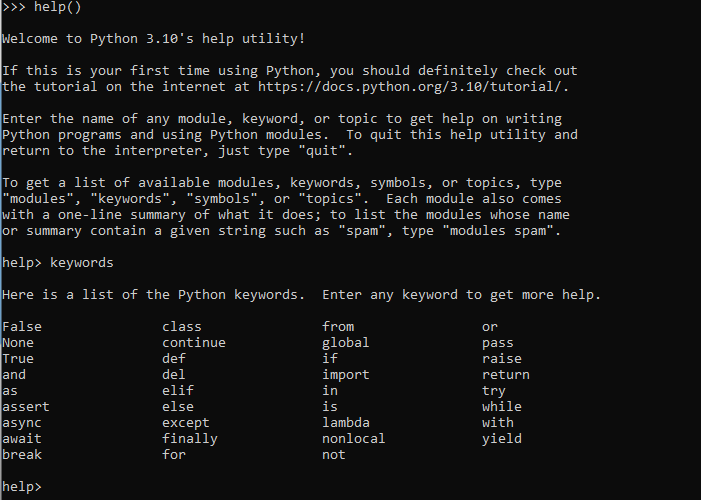
>>> lang='python'

>>> type(lang)

<class 'str'>

>>>

**Help in Python**



**Variable Names Rules –**

1. A variable name must start with a letter or the underscore character
2. A variable name cannot start with a number
3. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
4. Variable names are case-sensitive (age, Age and AGE are three different variables)

Can be defined in a single line like

a=b=c=11

can be defined like

a,b,c=34,45,56

**Print function in python**

print(object(s), sep=separator, end=end, file=file, flush=flush)

>>> print("my Name","is","pankaj",sep='-')

my Name-is-pankaj

>>>

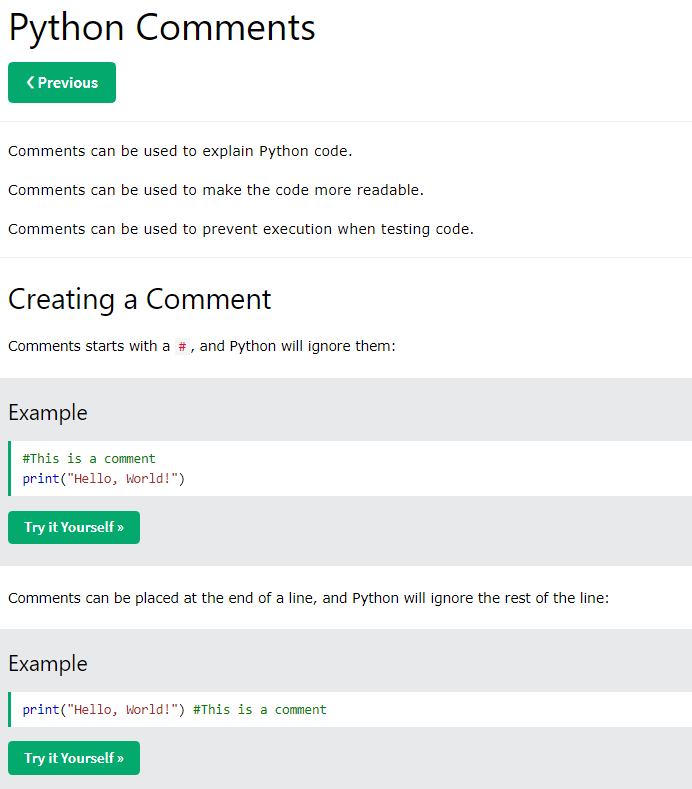
>>> print("my Name","is","pankaj",sep='-',end='\*\*')

my Name-is-pankaj\*\*>>>

>>>

>>>

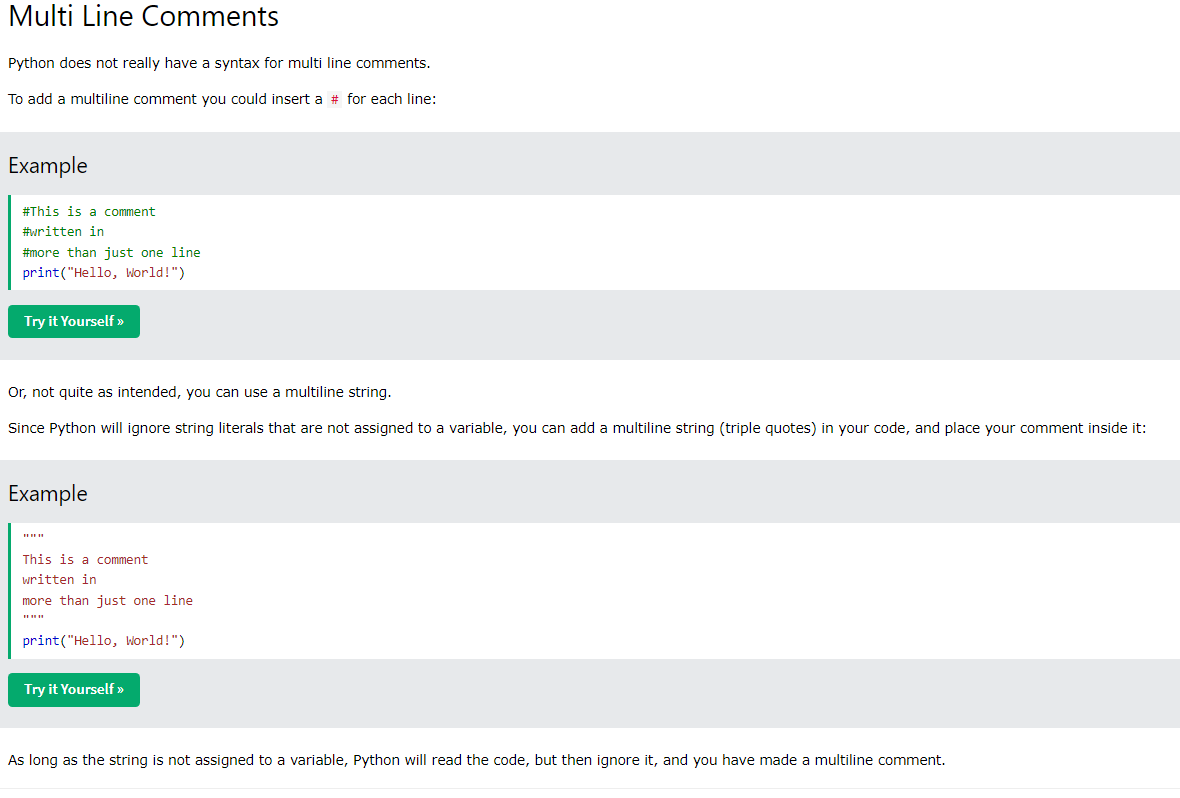
Comments in Python



Ctrl+/ - toggle to comment and uncomment a line in python

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code

Multiline comments



## **Python Keywords**

Python keywords are special reserved words that have specific meanings and purposes and can’t be used for anything but those specific purposes. These keywords are always available—you’ll never have to import them into your code.

Python keywords are different from Python’s [built-in functions and types](https://docs.python.org/3/library/functions.html). The built-in functions and types are also always available, but they aren’t as restrictive as the keywords in their usage.

An example of something you can’t do with Python keywords is assign something to them. If you try, then you’ll get a SyntaxError. You won’t get a SyntaxError if you try to assign something to a built-in function or type, but it still isn’t a good idea. For a more in-depth explanation of ways keywords can be misused, check out [Invalid Syntax in Python: Common Reasons for SyntaxError](https://realpython.com/invalid-syntax-python/#misspelling-missing-or-misusing-python-keywords).

>>> 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

>>> import keyword

>>> keyword.kwlist

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

>>> x=keyword.kwlist

>>> print(x)

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

>>> x

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

>>>

>>> len(x)

35

>>>

**Rules for creating Identifiers**

1. An Identifier can be combination of alphabets and digits but it cannot start from digit
2. No Special Character except underscore (\_) can be used in identifiers.
3. No Keywords can be used as Identifiers
4. Python is Case-Sensitive
5. It should be small and meaningful Name although no rule for length for variable

**Validity of identifier** – isidentifier() function can be used

“12number”.isidentifier()

False

**line indentation**

Python does not use {} to indicate block of code for class and function definitions or flow control. Block of code in python are denoted by line indentation.

No semicolon in python code

**Datatypes in Python:**

1. Datatype represent a type of value hold by variables
2. Python does not allow us to explicitly declare datatype while creating variables it does automatic for us.
3. Python is Dynamic type programming language

**Numeric Data Types (int, float, complex)** –

1. Complex datatype – x+yj

Where x is real part and y is imaginary part and j represent sqrt-1

4+5j can be stored in a variable like

Var=4+5j

If c1=3+8j and c2=8+9j

S=c1+c2=11+17j

s.real=11.0

s.imag=17.0

real and imag are property of complex datatype to know the real and imaginary part of complex number.

Python does not have limits for integers or floats. It can store very very big or small numbers or floats in your code.

**String type in Python**

1. Sequence of Character placed with in single, double or triple quotes is string
2. Strings are immutable (cannot be changed)
3. No Char datatype available in Python, everything in quotes is string
4. Operations performed on Strings – String Handling
   1. Reverse
   2. Concatenation
   3. Upper/Lower/Proper
   4. Slice operator

Indexing and Slice Operator –

1. [start:end:step] – Slice operator can be used with list and tuple as well along with string
2. Start – start index, end – end index (not included i.e. excluded),, step – by default it is 1 if not specified
3. E.g. x=’python’

0 1 2 3 4 5

X -> p y t h o n

Size of string x is 6 which are indexed from 0 to 5.

S=len(x)

print(s) – 6

print(x[0:5]) – pytho

print(x[0:5:2]) – pto

print(x[4: ]) - on

print(x[ :6]) – python