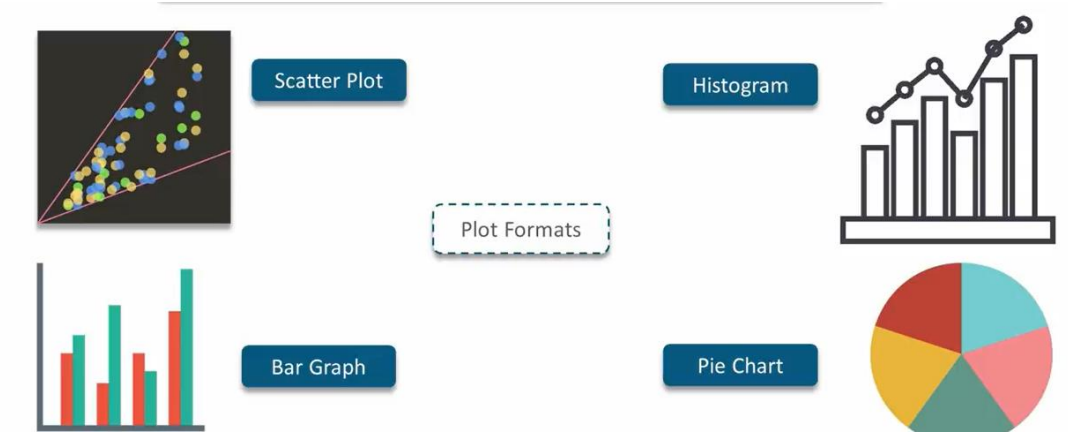
MTE204

INTRODUCTION TO DATA ANALYSIS IN PYTHON



1.0. Introduction to Python

1.1.Why Python

Python is simple and easy to learn, read, and write. It is a Free/Libre and Open Source Software (FLOSS). Meaning one can distribute copies freely, read its source code, modify it, etc. it is a high-level language and portable: meaning it is supported by Linux, Windows, FreeBSD, Macintosh, Solaris, BeOS, OS/390, PlayStation, and Windows CE platforms. Python supports procedure-oriented programming as wells as object-oriented programming. It can also invoke C and C++ libraries can be called from and C++ programs, can integrate with Java and NET components. Python has been used by many of the big companies known today such as: YouTube, Google, Dropbox, RaspberryPi, BitTorrent, NASA and NETFLIX.

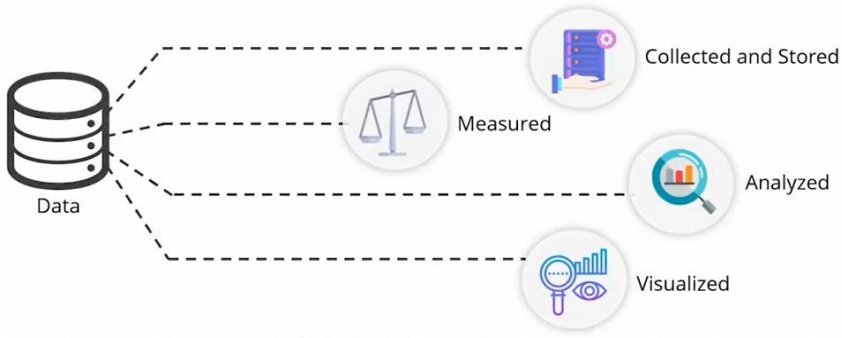
Python application includes:

1. Web Scrapping
2. Automation Testing
3. Web Development
4. Data Analysis (Our focus in this course)

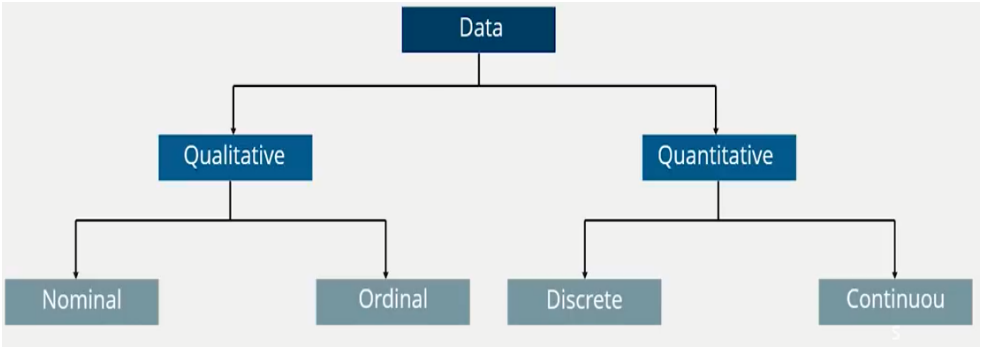
After the installation of Python, the next step is to start working on python. We will discuss some important attributes then move to data analysis with python.

1.2. What is Data

Data in terms of statistics and probability refers to facts and statistics collected together for reference or analysis. The figure below shows what can generally be done with data.

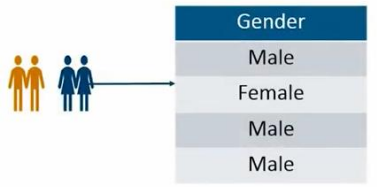


Data is subcategorized as depicted below:



Quantitative data deals with characteristics and descriptors that cannot be easily measured, but can be observed subjectively.

- Nominal data are data with no inherent order or ranking such as gender or race.



- Ordinal data are data with an ordered series.

Customer ID	Rating
001	Good
002	Average
003	Average
004	Bad

Qualitative data deals with numbers and things that can be measured objectively.

- Discrete data are also known as categorical data; it can hold finite number of possible values e.g. number of student in a class.
- Continuous data are data that can hold infinite number of possible values e.g. a person's weight.

It is worthy to mention here types of variable; which are:

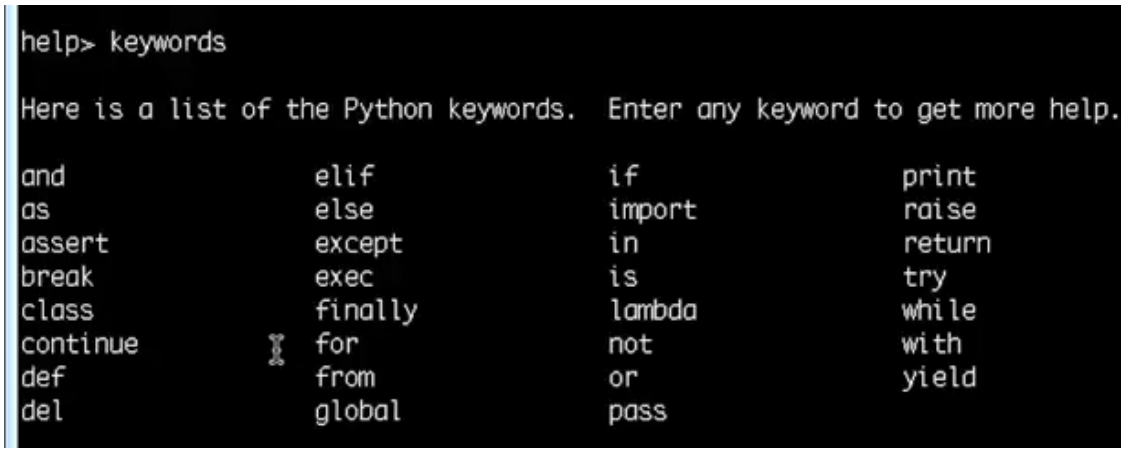
- Discrete variable also known as categorical variable; it can hold values of different categories. For instance, your email can hold value for inbox message or spam message.
- Continuous variable are variables that stores infinite number of values e.g. a vehicle speed.

Hence, variable is anything used to store a value and the kind of data associate with such variable determines if such variable is discrete or continuous.

Variables can either be dependent or independent. Dependent variable are variables whose value depends on any other independent variable.

1.3. Important Attributes

Presented below as some keywords in python



1.3.1. Comments

Passing comments in python can be done using the # or *'''comment'''*. Where there is only one line of comment the # is used. That is any text to the right of # is not executed by python. The *'''comment'''* is applicable where multiply line of comment is to be passed in python.

Example:

 # This code analysis a dataset

Or

'''

This

Code

Analysis

a

dataset

'''

```
# Comment

'''
Bulk Comments
Multi-line Comment
'''
```

Presented below is a demo of the python interface

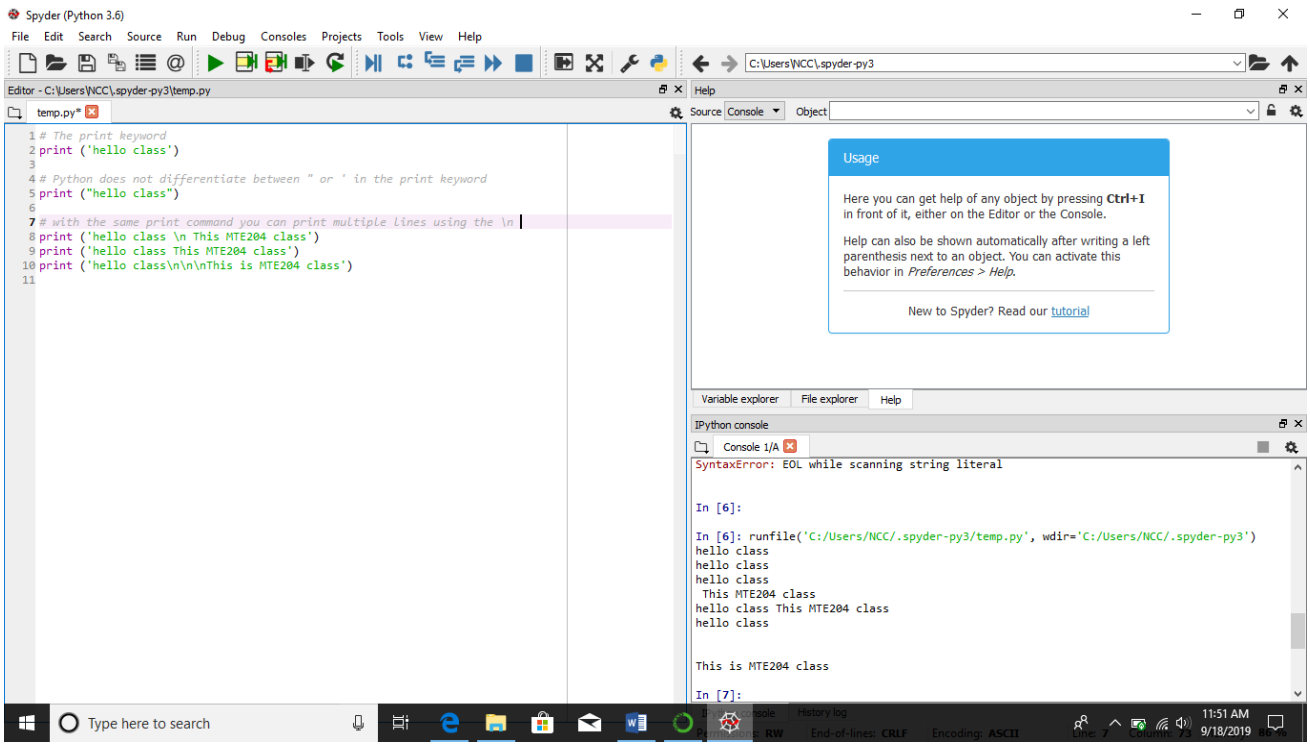


Figure 1 Character printing

We can see form the Fig. 2 above that the keyword *print* is used to print the character in the string (“”) or (‘’) in the brace.

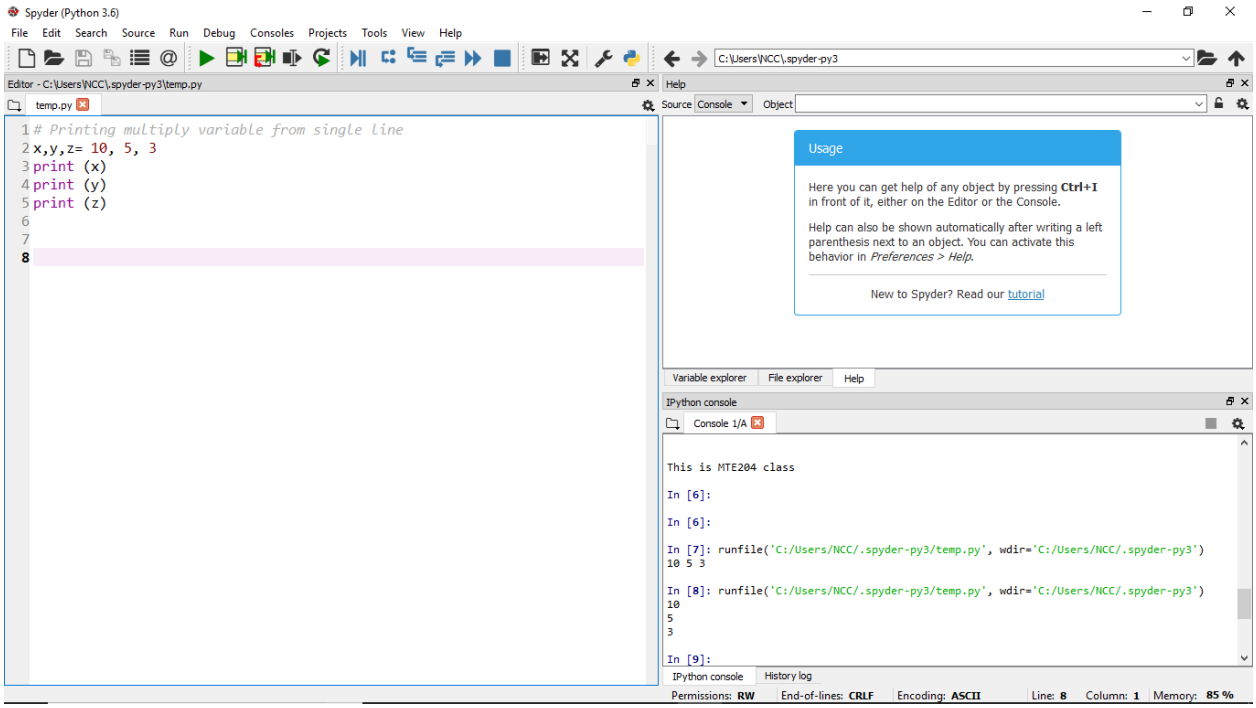


Figure 2 Printing multiply variable form a single line input

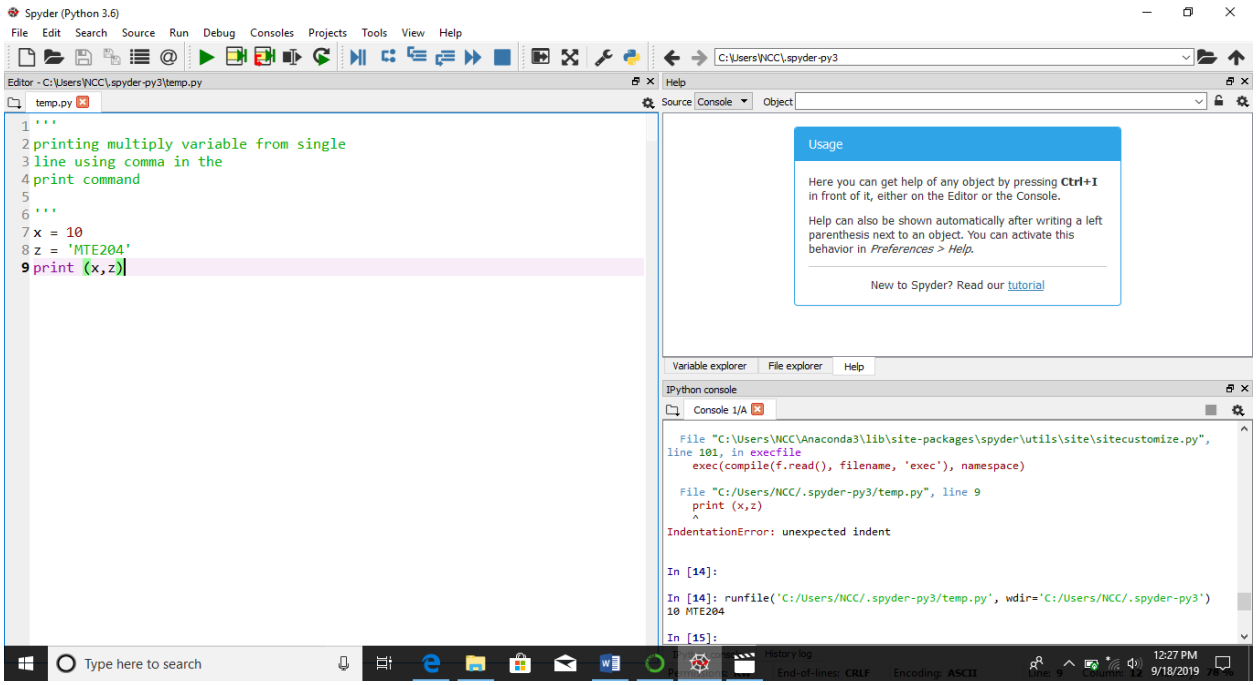


Figure 3 Printing multiply variable with comma in print command brace.

1.3.2. Identifier

This is a name used to identify a variable, function, class, module or other object. For instance, in Fig 3 X, Y and Z are identifiers. An identifier can start with A to Z or a to z or an underscore (_) followed by zero or more letters, underscore and digits (0 to

9). Note that python is a case sensitive programming language and does not allow any special character within identifiers such as %, $ etc.

Identifier Naming Convention

1. class name start with an uppercase letter. All other identifier starts with lowercase letter.
2. Starting an identifier with one leading underscore means that identifier is private
3. Starting an identifier with two leading underscores means that identifier is strongly private
4. Ending an identifier with two trailing underscores means that identifier is language-defined special name

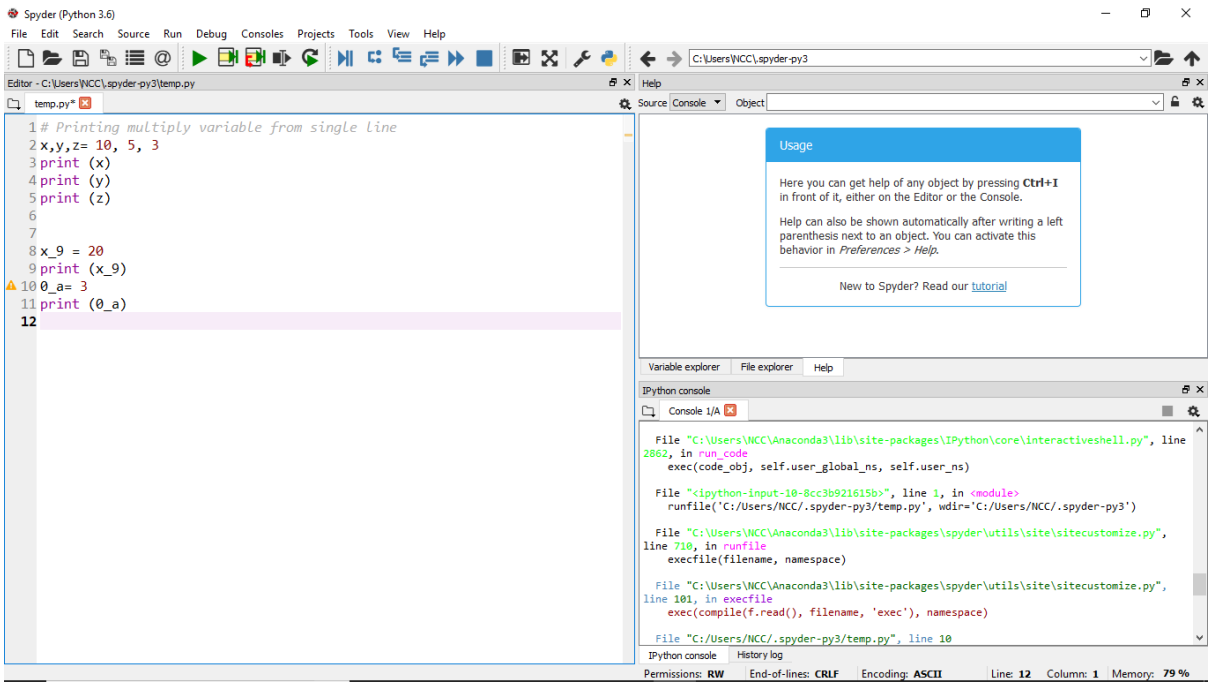


Figure 4 identifier

1.3.3. Standard Data Types

Data type is a way of defining what kind of data and entry is; it can be numbers, integer, float, Boolean etc. The data is the entry on the right side of the equality sign and to the left is the identifier. Data can mutable or immutable data type. The figure below presents more details:

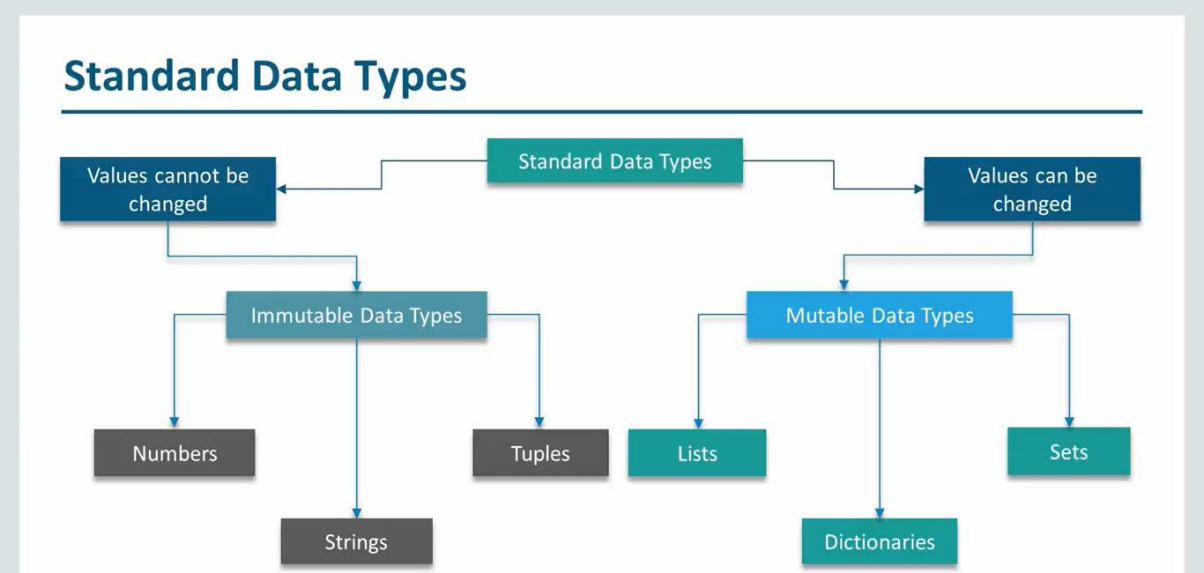
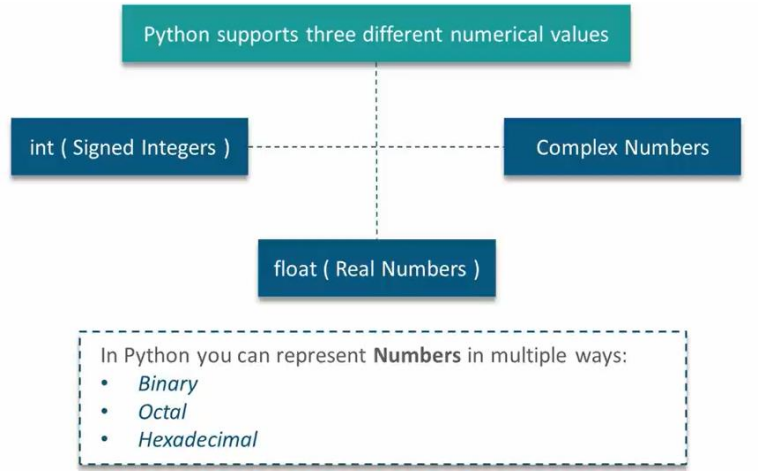
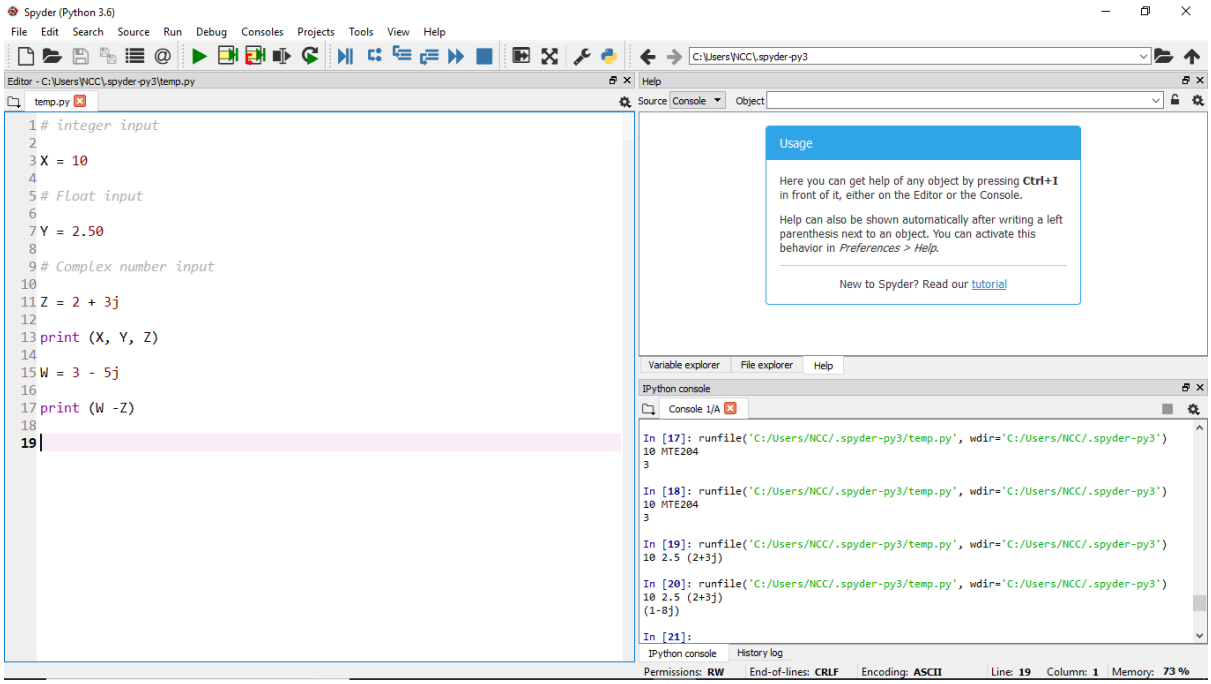


Figure 5 Standard Data Type

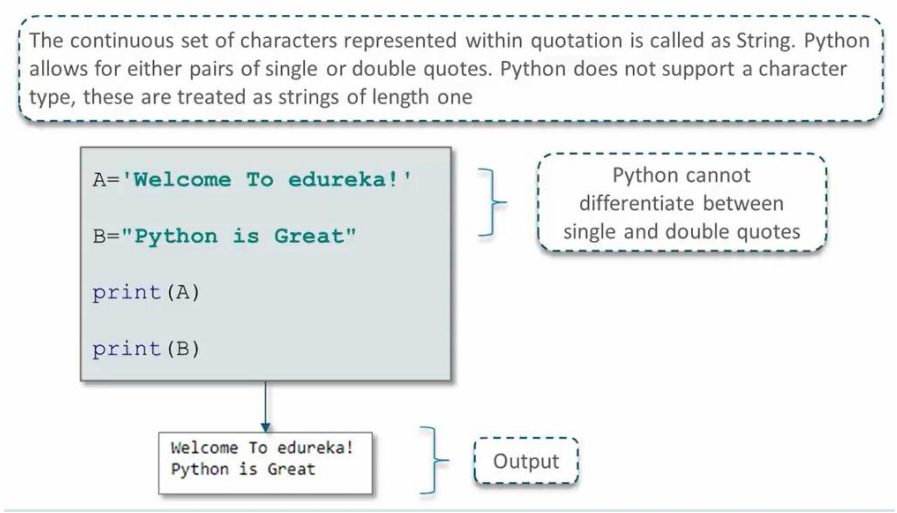
Immutable Data Type –

1. Numeric Data Type




Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Users\NCC\.spyder-py3\temp.py
temp.py
1 # integer input
2
3 X = 10
4
5 # Float input
6
7 Y = 2.50
8
9 # Complex number input
10
11 Z = 2 + 3j
12
13 print (X, Y, Z)
14
15 W = 3 - 5j
16
17 print (W -Z)
18
19
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer File explorer Help
IPython console
Console 1/A
In [17]: runfile('C:/Users/NCC/.spyder-py3/temp.py', wdir='C:/Users/NCC/.spyder-py3')
10 MTE204
3
In [18]: runfile('C:/Users/NCC/.spyder-py3/temp.py', wdir='C:/Users/NCC/.spyder-py3')
10 MTE204
3
In [19]: runfile('C:/Users/NCC/.spyder-py3/temp.py', wdir='C:/Users/NCC/.spyder-py3')
10 2.5 (2+3j)
In [20]: runfile('C:/Users/NCC/.spyder-py3/temp.py', wdir='C:/Users/NCC/.spyder-py3')
10 2.5 (2+3j)
(1-8j)
In [21]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 19 Column: 1 Memory: 73 %

2. Sting


The continuous set of characters represented within quotation is called as String. Python allows for either pairs of single or double quotes. Python does not support a character type, these are treated as strings of length one
A='Welcome To edureka!'
B="Python is Great"
print(A)
print(B)
Python cannot differentiate between single and double quotes
Welcome To edureka!
Python is Great
Output

3. Tuples

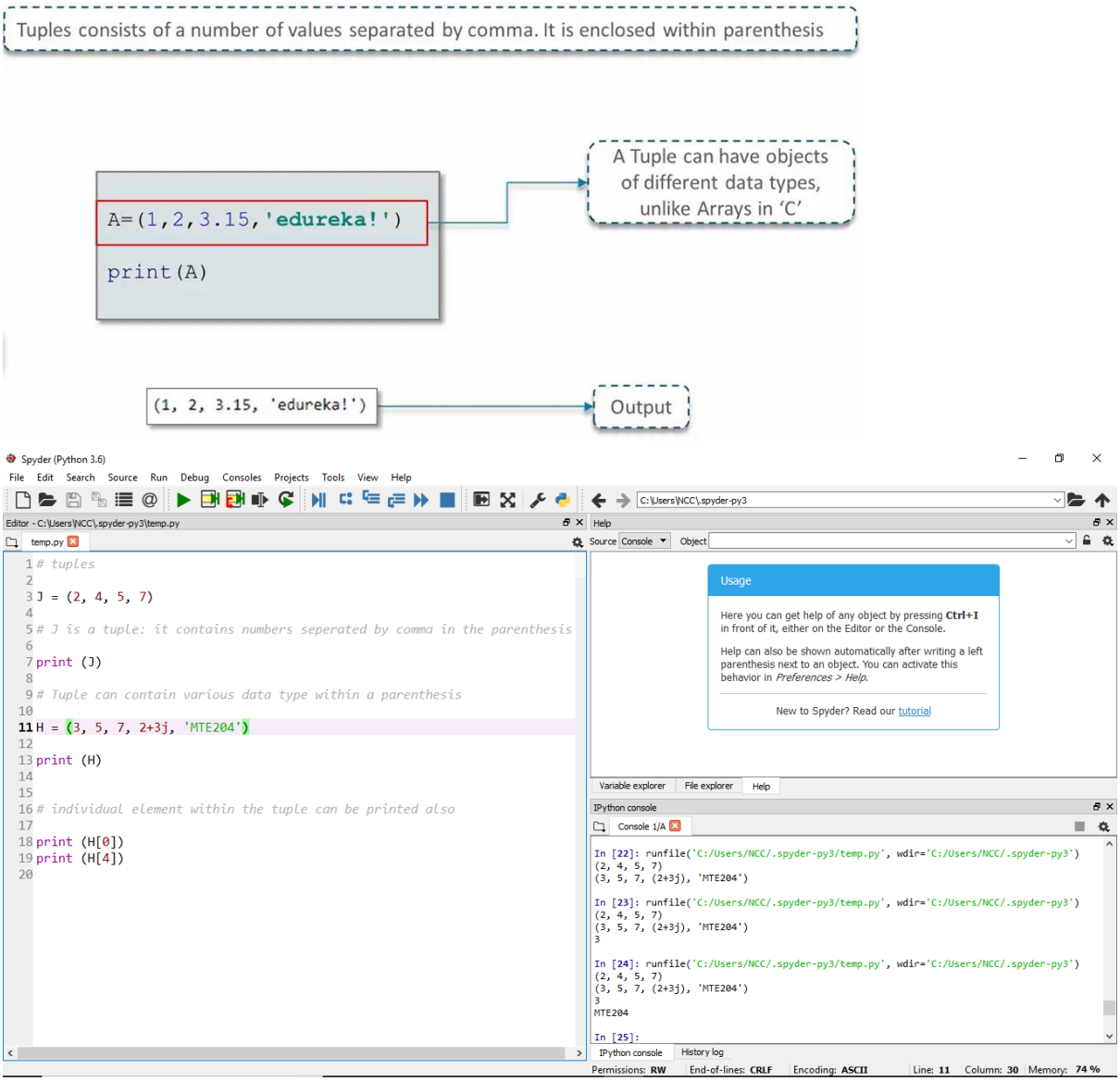
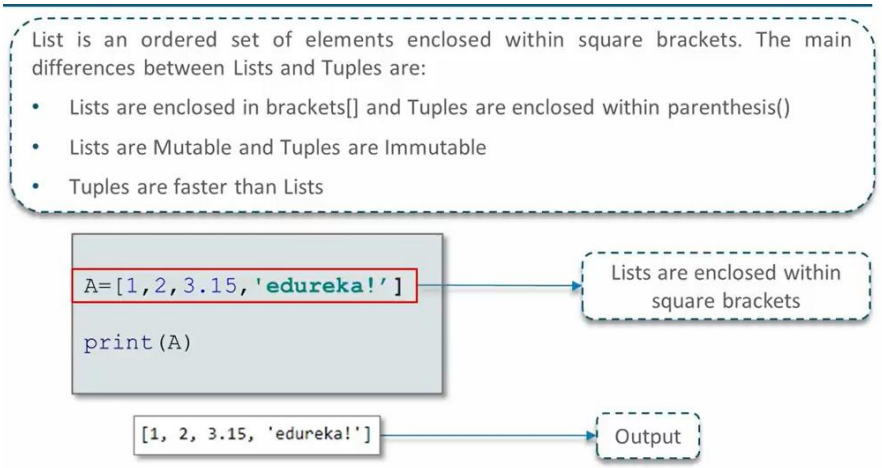
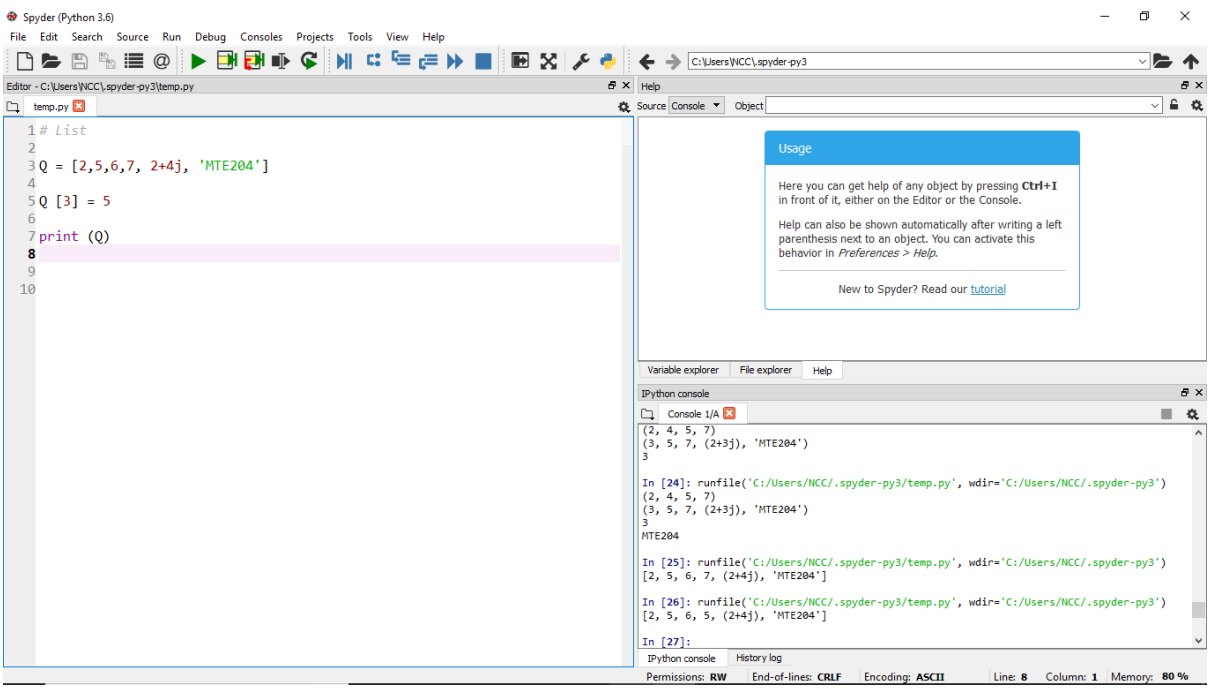


Figure 6 Tuples

Mutable Data Type

1. List





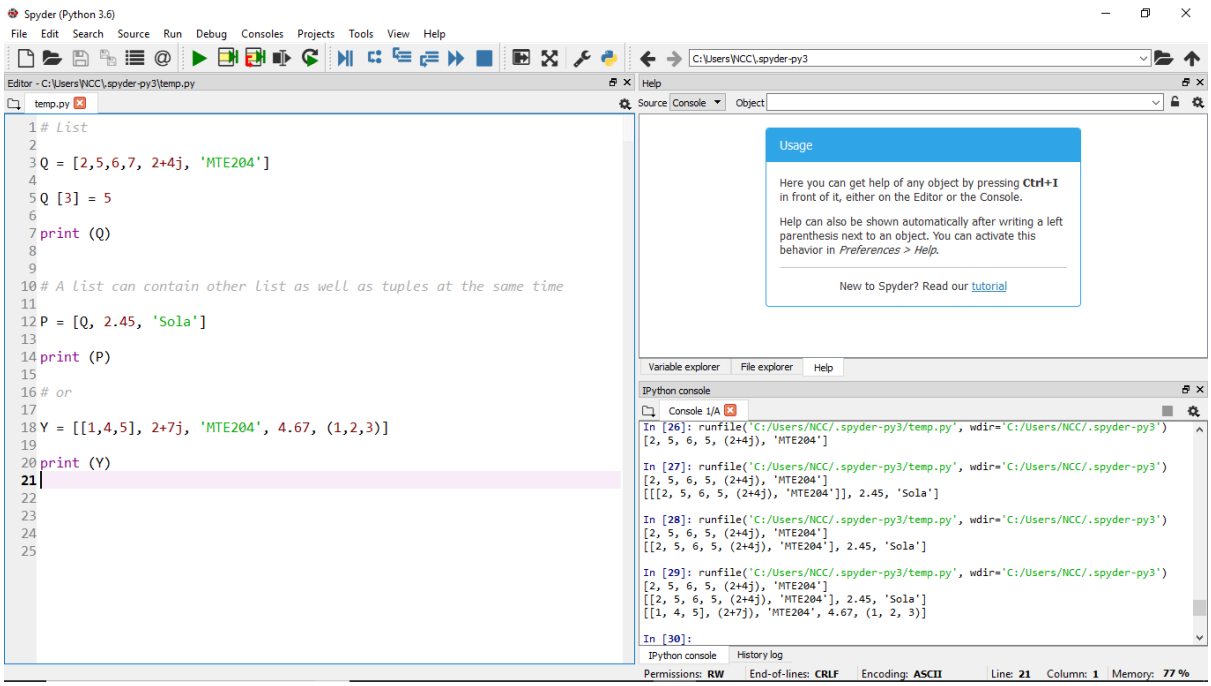
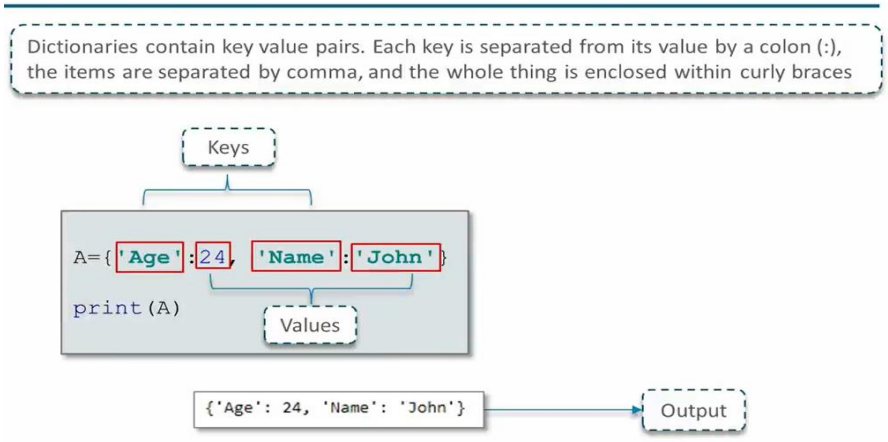


Figure 7 List data type

2. Dictionaries



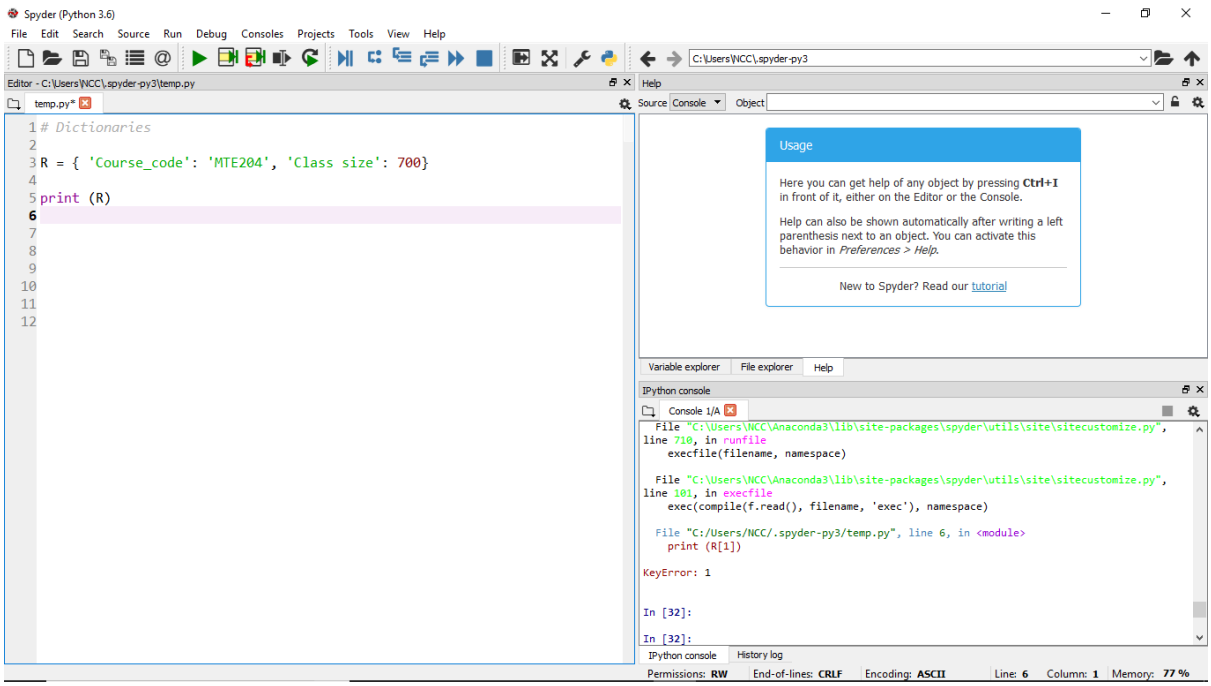
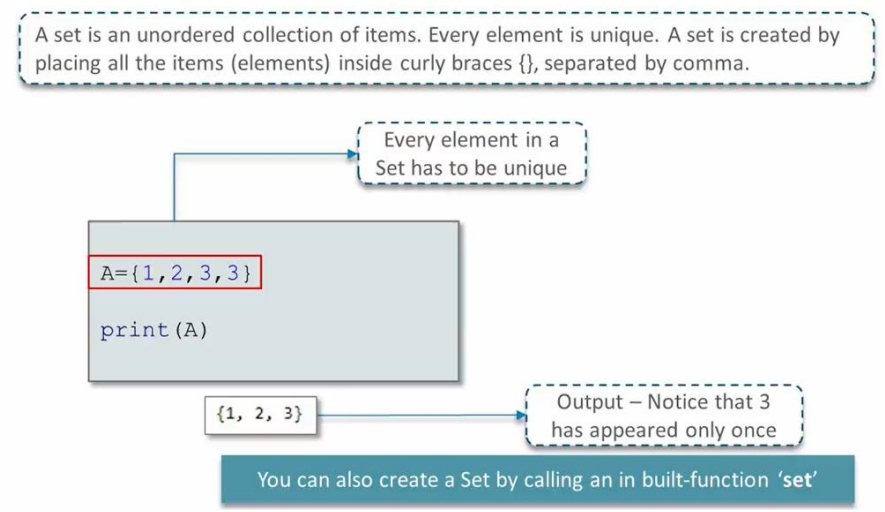


Figure 8 Dictionaries

You can have dictionary within a dictionary or any other combination of data types.

3. Sets



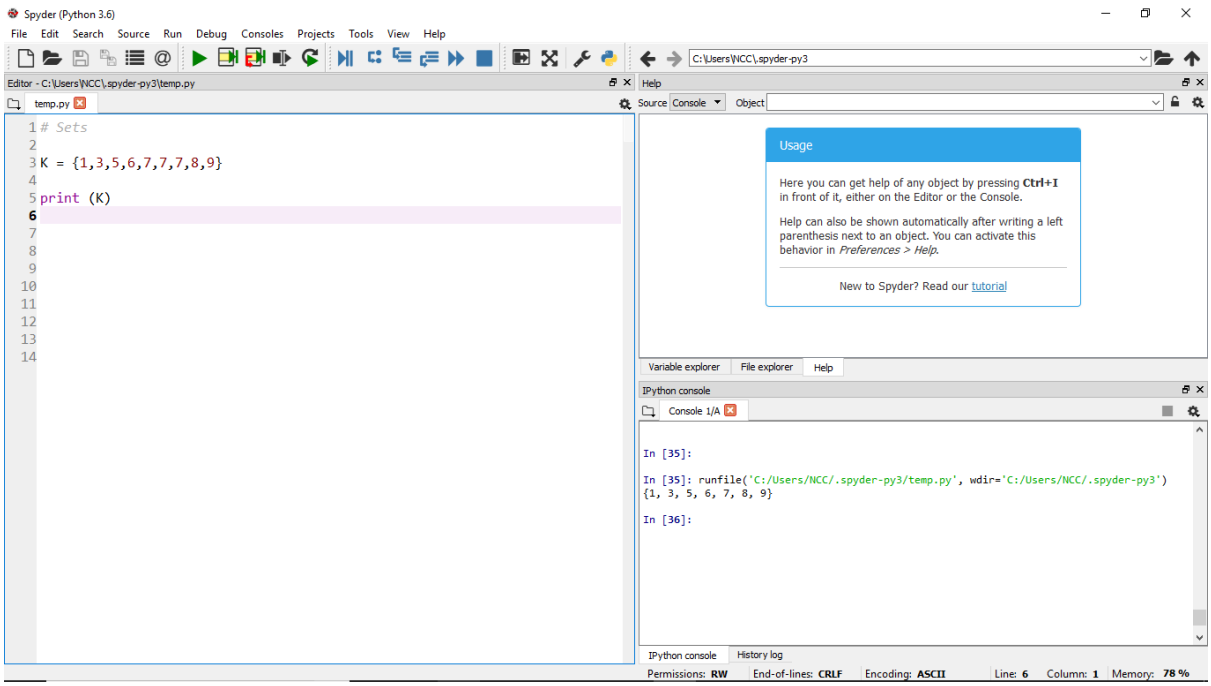
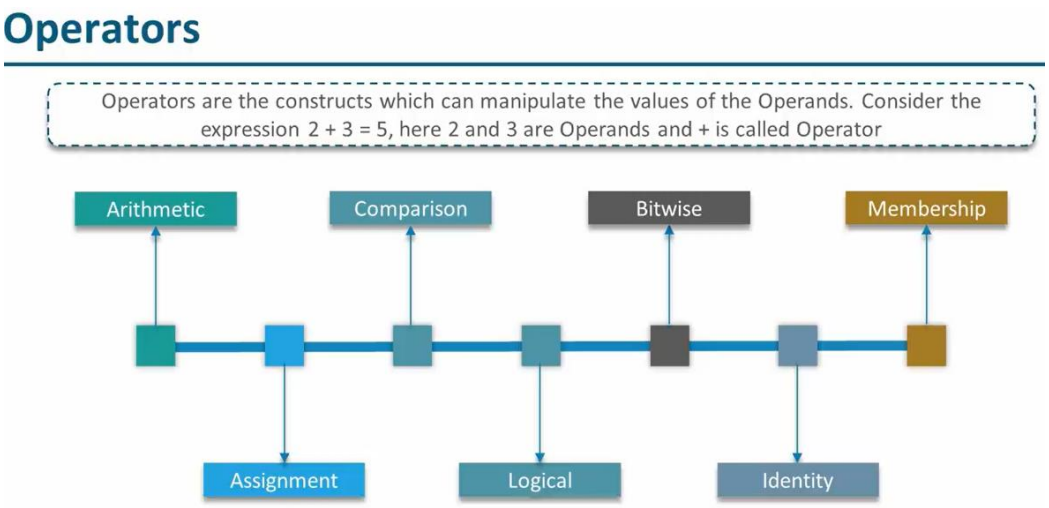


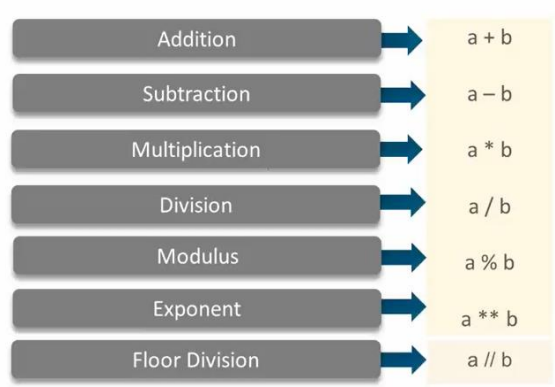
Figure 9 Sets

1.3.4. Operators

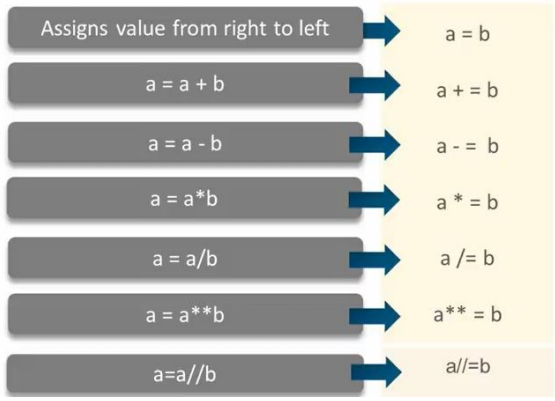
Operators are important in the execution of operation in python. Fig 11 below presents operators in python.



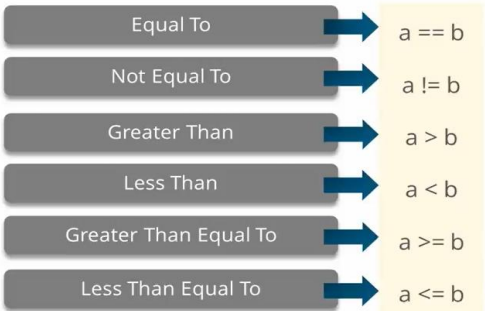
1. Arithmetic Operators

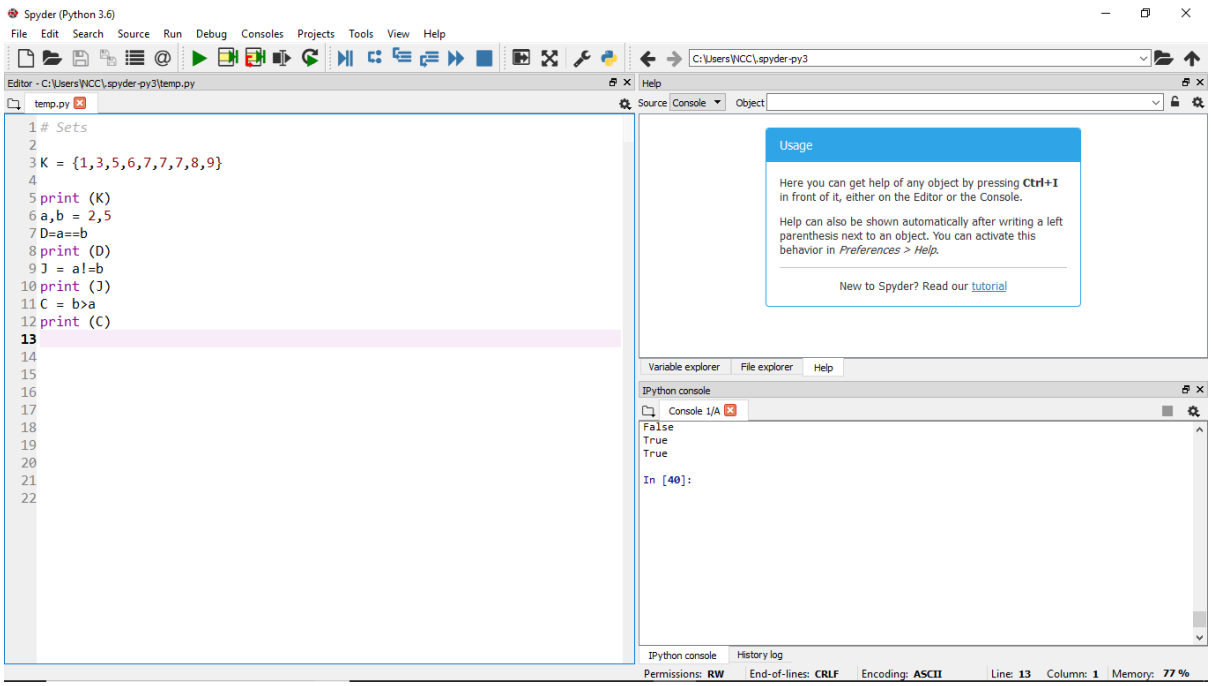


2. Assignment Operator

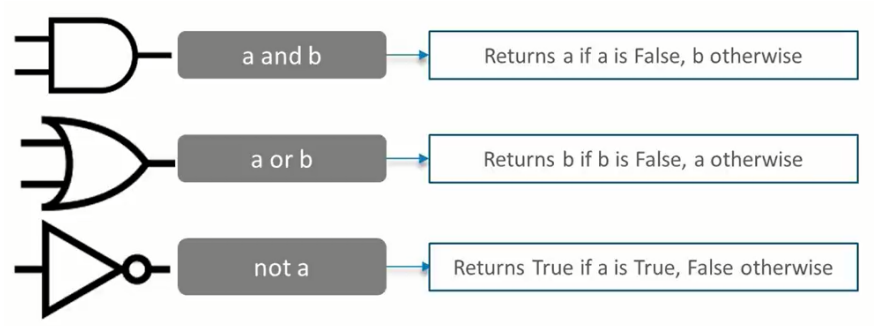


3. Comparison Operator

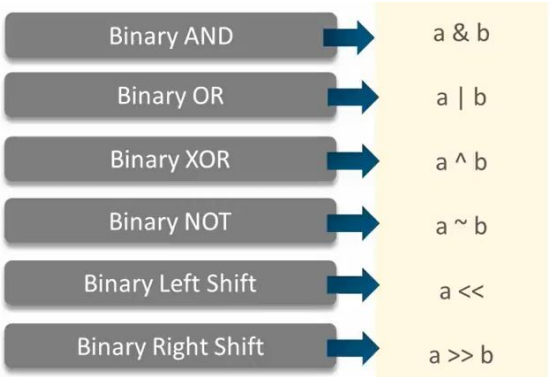




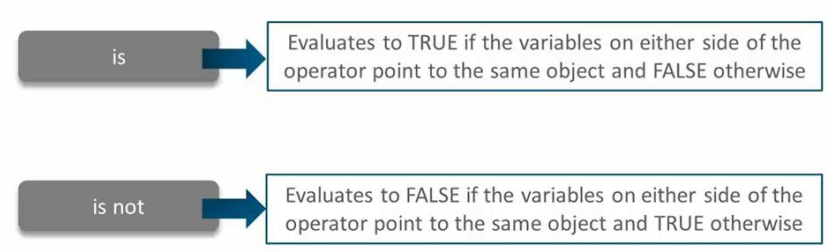
4. Logic Operators



5. Bitwise Operator



6. Identity Operator



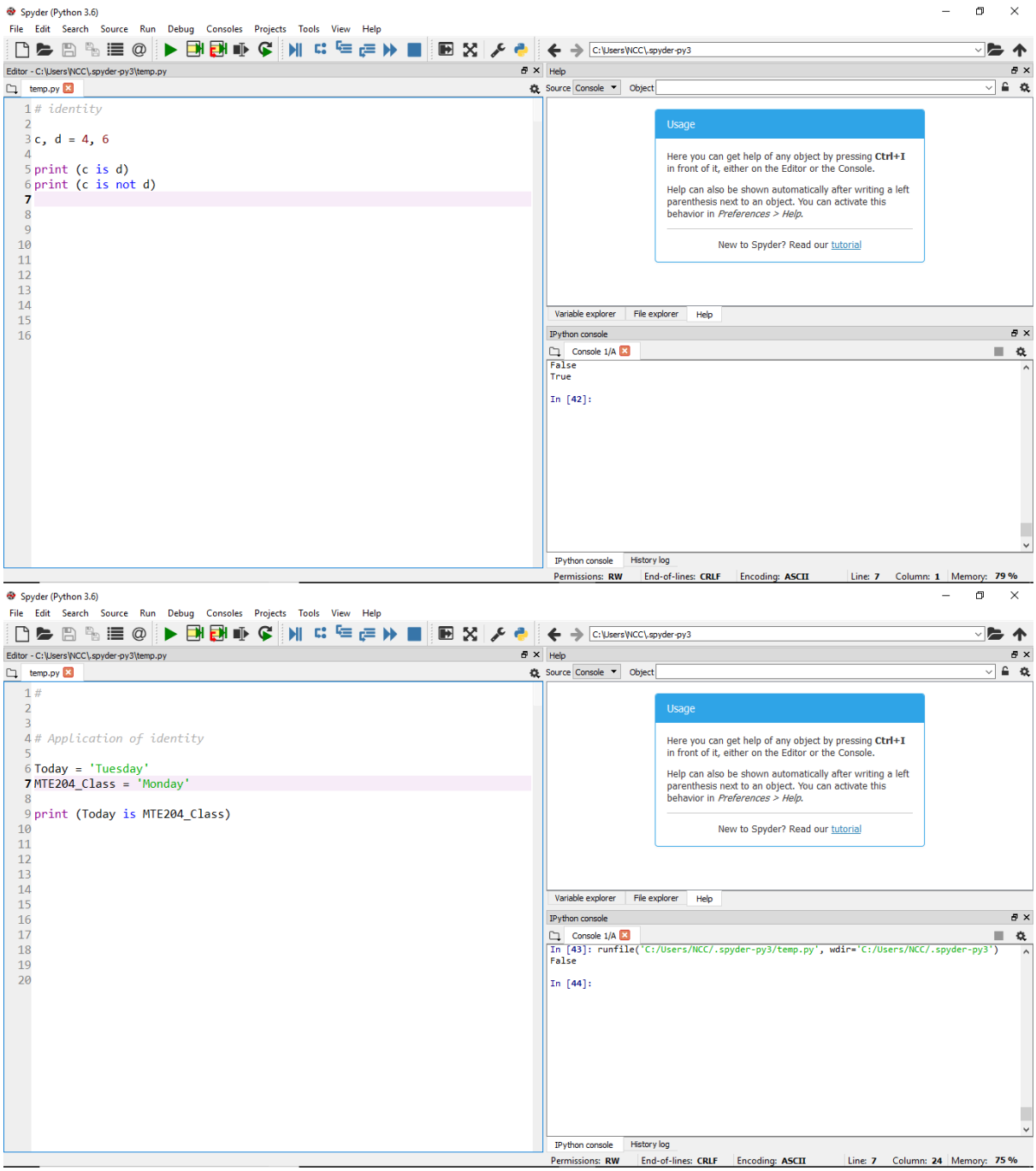
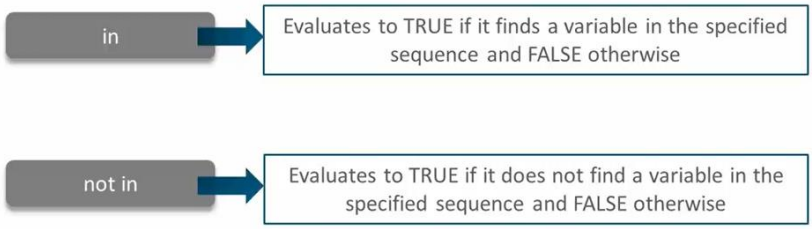


Figure 10 Identity operator

7. Membership Operator: works on list, dictionary and tuples to check if an element exist within any of those.



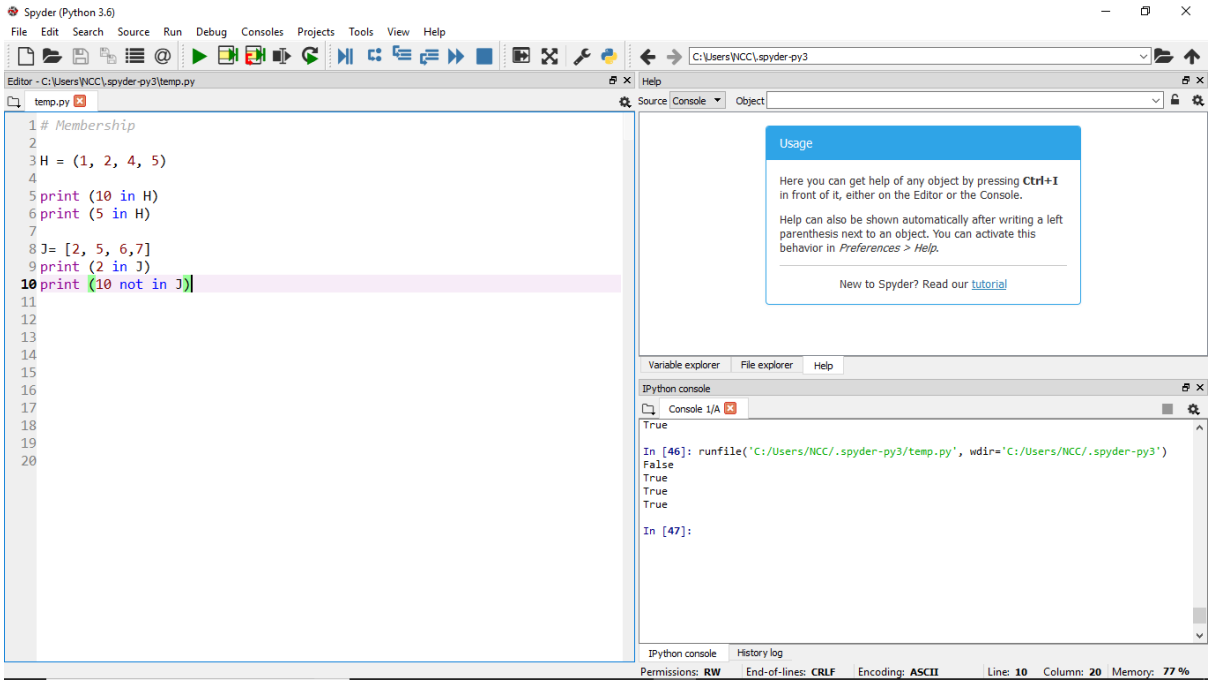


Figure 11 Membership

2.0. Data Analysis with Python

This is the process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information's, suggesting conclusions and supporting decision making.

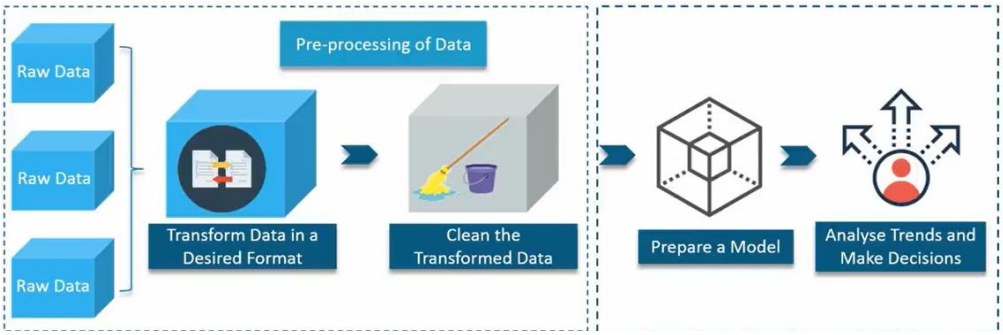


Figure 12 Data life cycle

Python provides various methods for data analysis, manipulation (NumPy and Pandas libraries) and visualization (Matplotlib library) (see Fig 14).

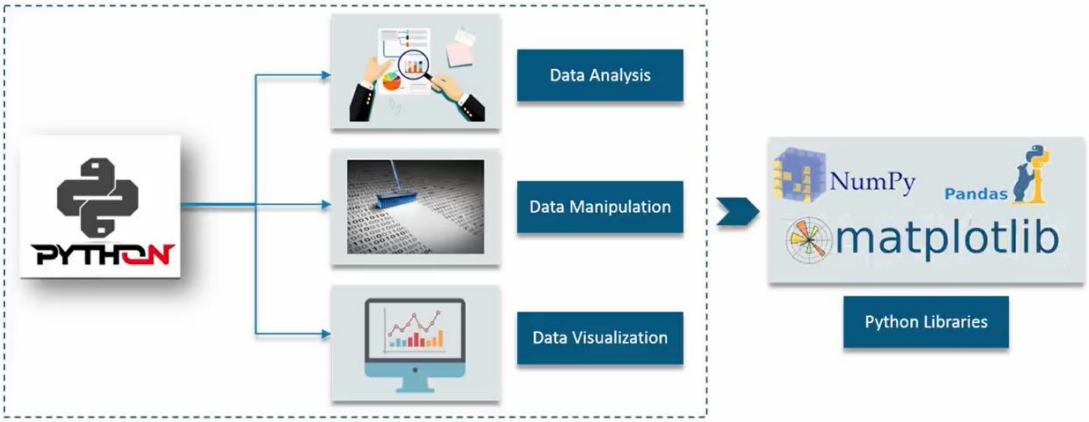


Figure 13 Data Analysis in Python

2.1. Introduction to NumPy Library

NumPy is a package for scientific computing in Python. NumPy features is presented in Fig. 15 and operations in NumPy is presented Fig. 16.

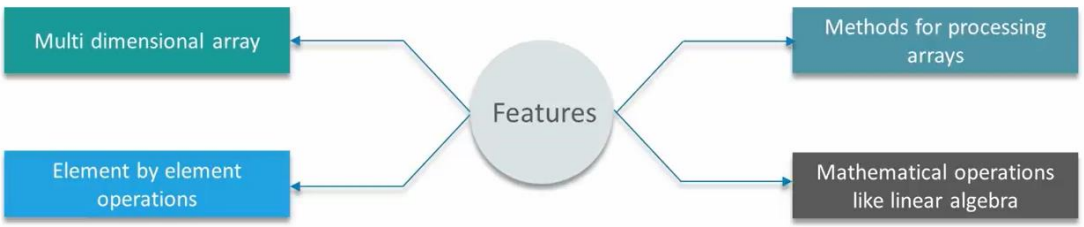


Figure 14 NumPy Features

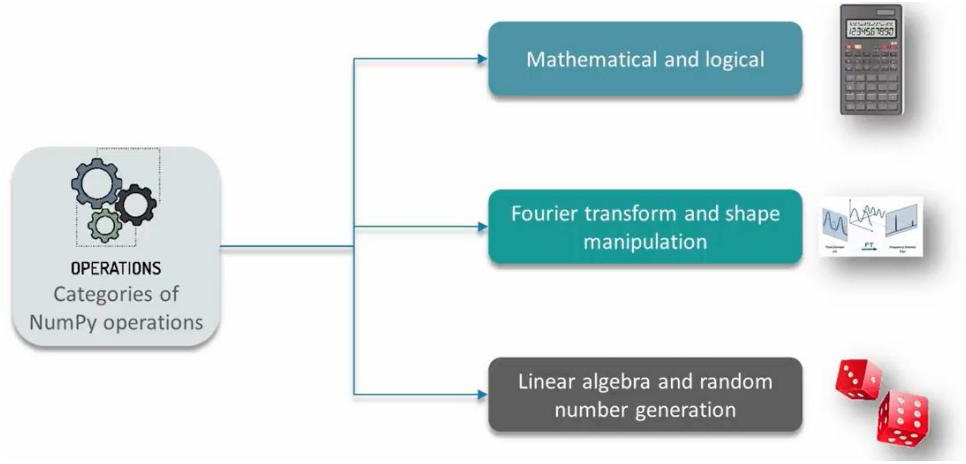
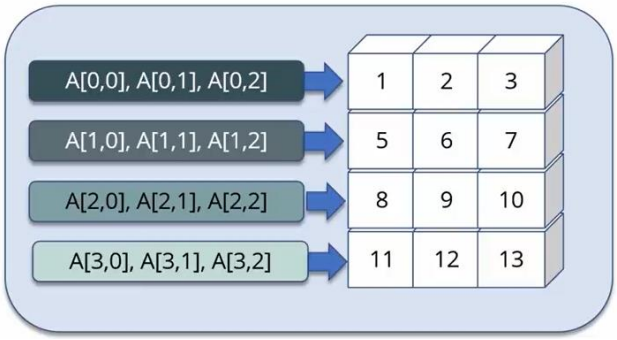
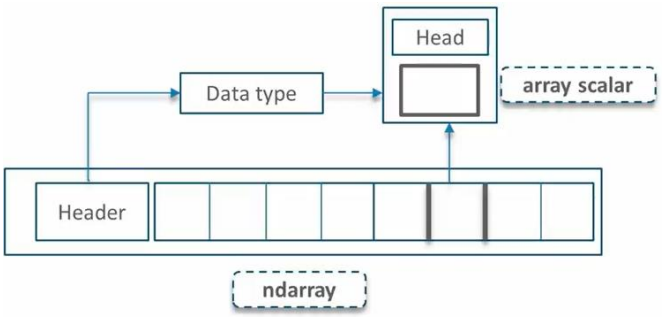


Figure 15 Operations in NumPy

NumPy has an array called ndarray; is is a multidimensional array object of two parts-the actual data, some metadata which describes the stored data. They are indexed just like sequences are in Python, starting from 0



Each element in ndarray is an object of data-type object called *dtype*. An item extracted from ndarray, is represented by a Python object of an array scalar type (Please note that this done internally by Python).



2.1.1. Creating a NumPy Array

The NumPy is a library in Python therefore, the first step is to import the NumPy library. See Fig 17.

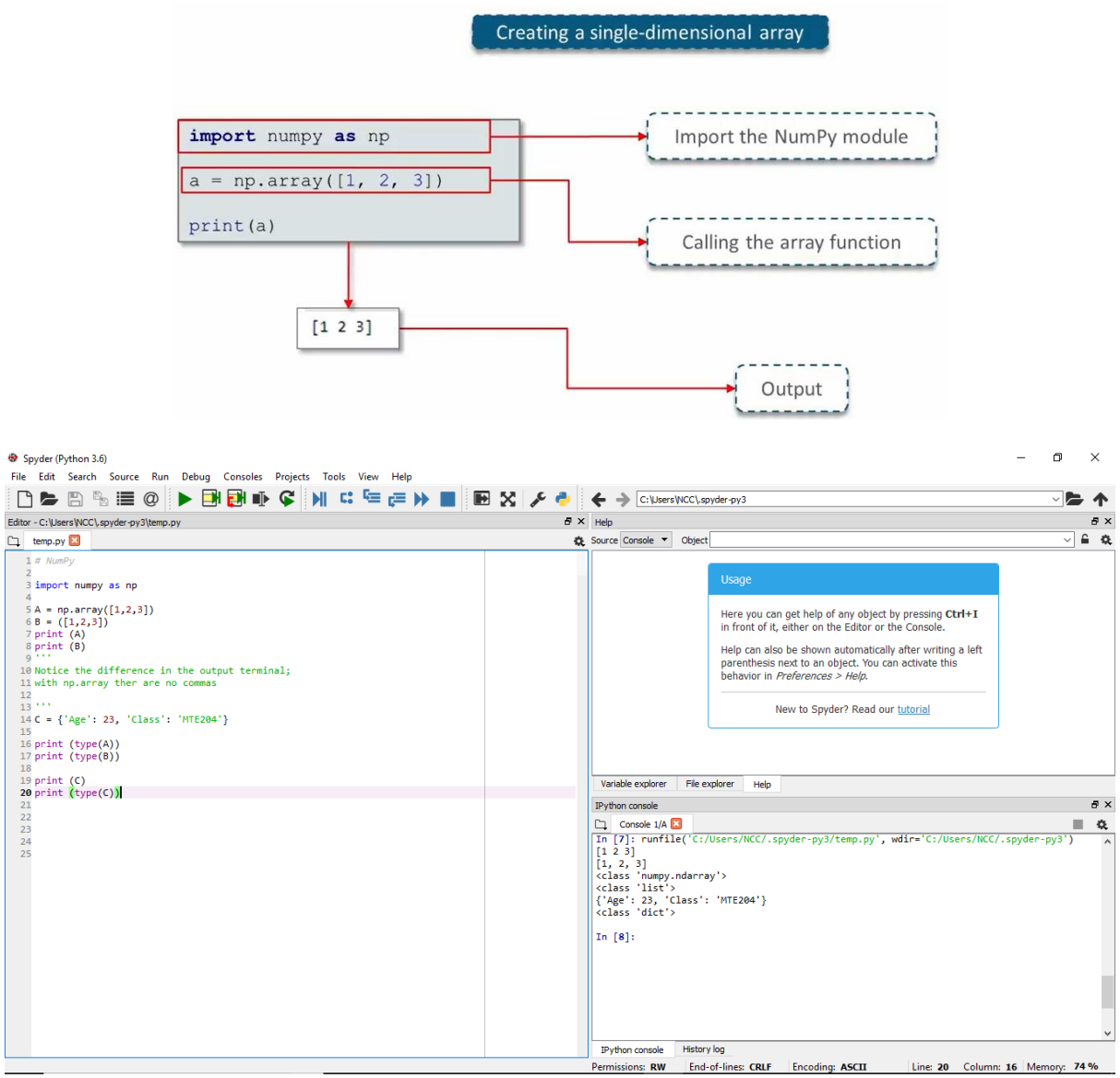


Figure 16 creating NumPy Array

The linspace function in Python can allow us to create a vector by calling the linspece function in NumPy and specifying the initial, final and the step. See Fig. 18.

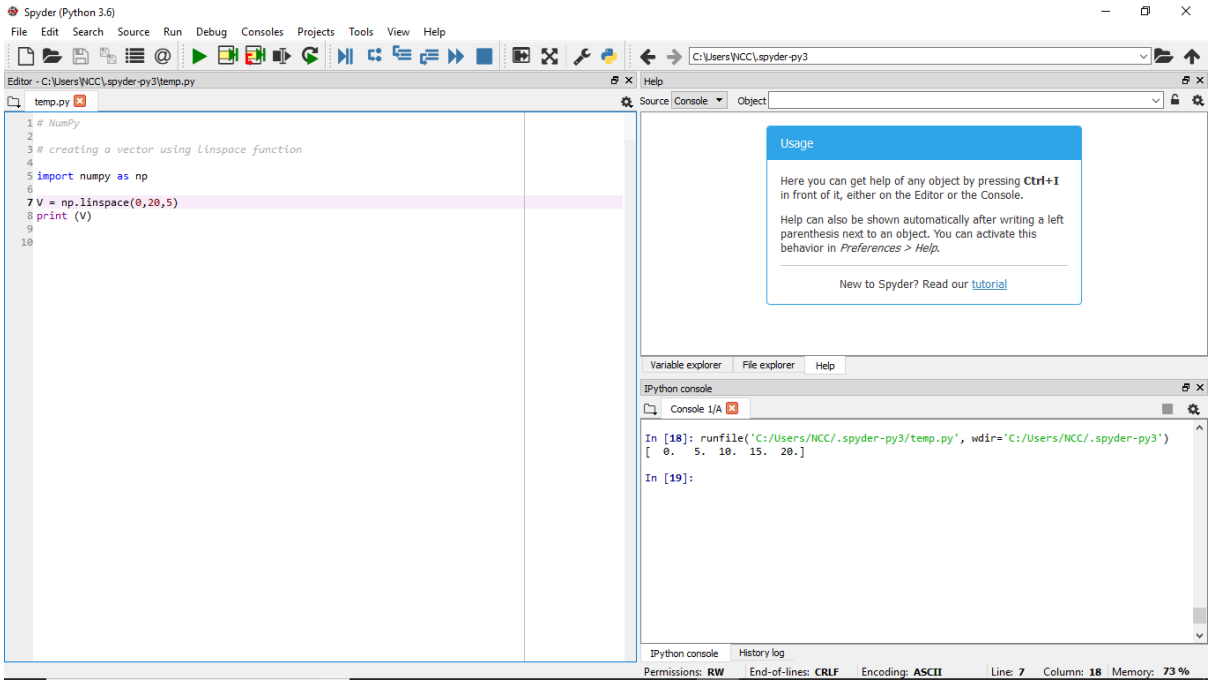


Figure 17 Linspece function

2.1.2. Creating a Multidimensional NumPy Array

Import the NumPy library as np (as in Fig 17) and pass the array code as seen in Fig. 19.

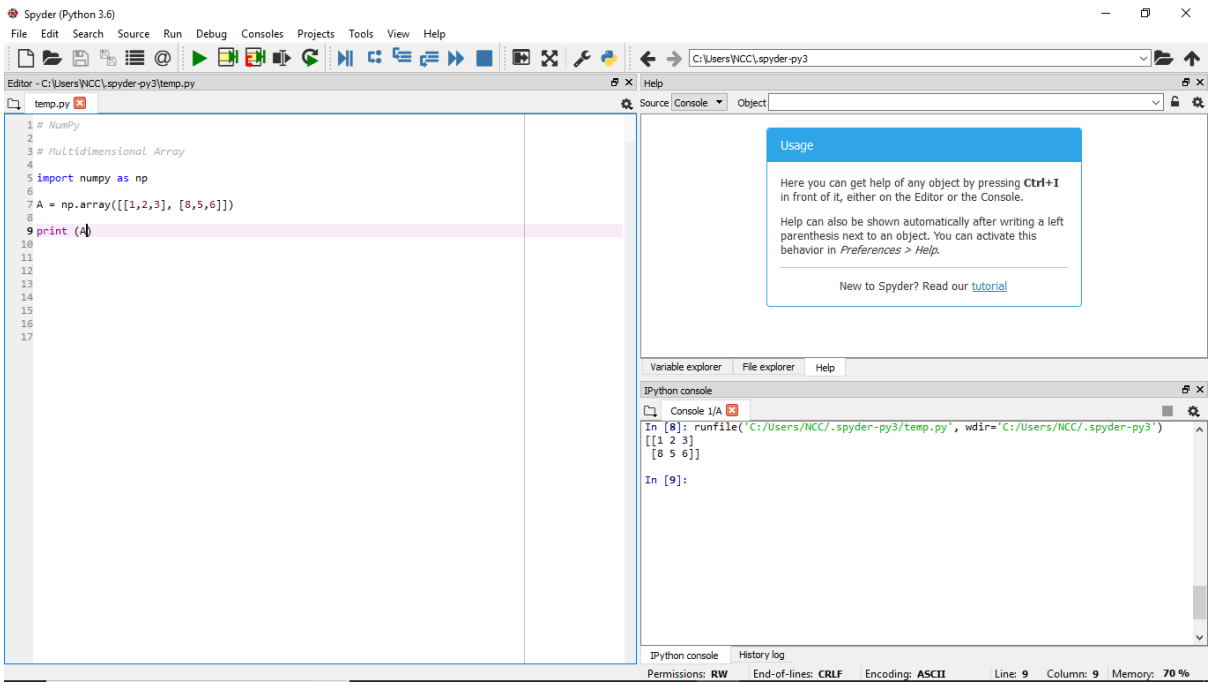


Figure 18 Multidimensional NumPy Array

The arrange function can also be used to create a multidimensional array within a specified range. See Fig. 20.

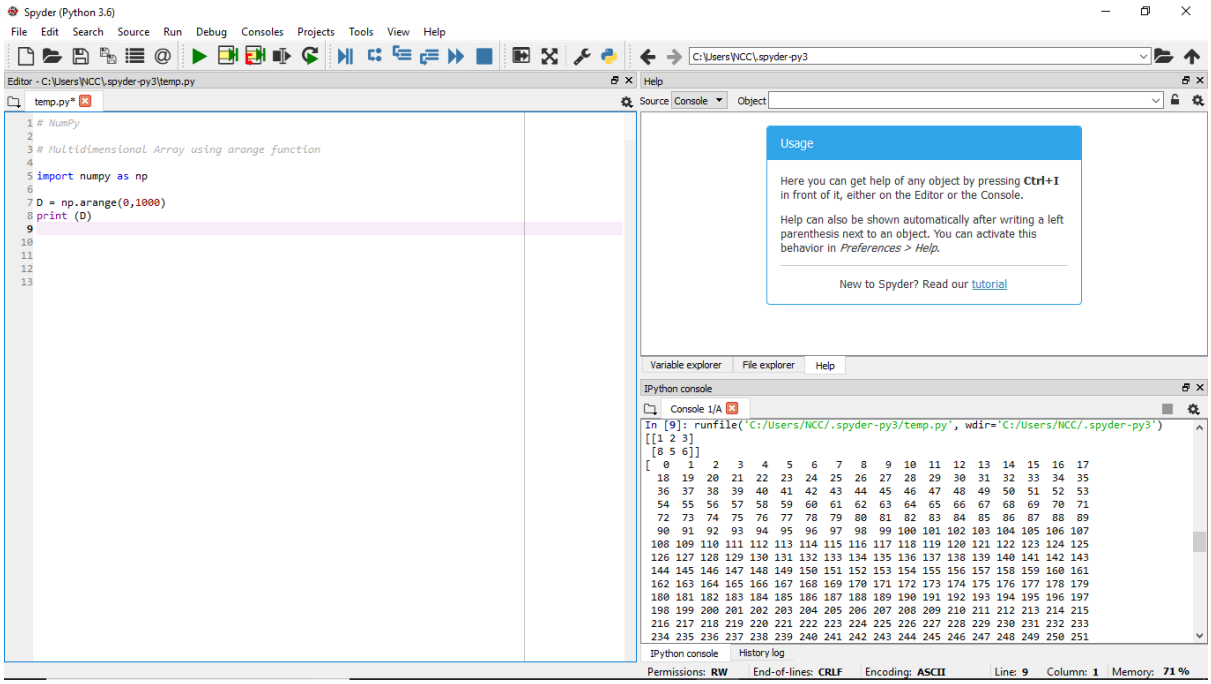


Figure 19 Arange function

We can also create an array of zeros by using the zeros function and specifying the number of rows and column. See Fig. 21

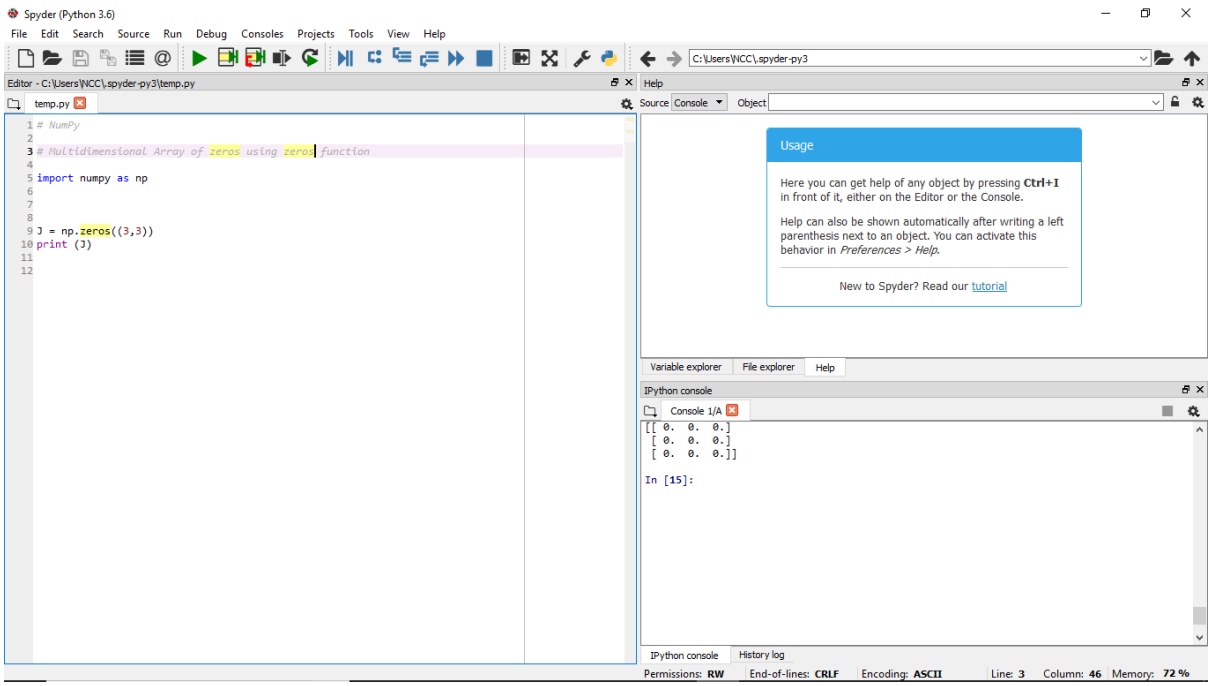


Figure 20 Array of zeros

2.1.3. Creating an Array from Existing Data

The numpy.asarray is used converting Python sequence into ndarrays.

- Syntax - *numpy.asarray(a, dtype = None, order = None)*
- The following piece of code converts a python list into an array

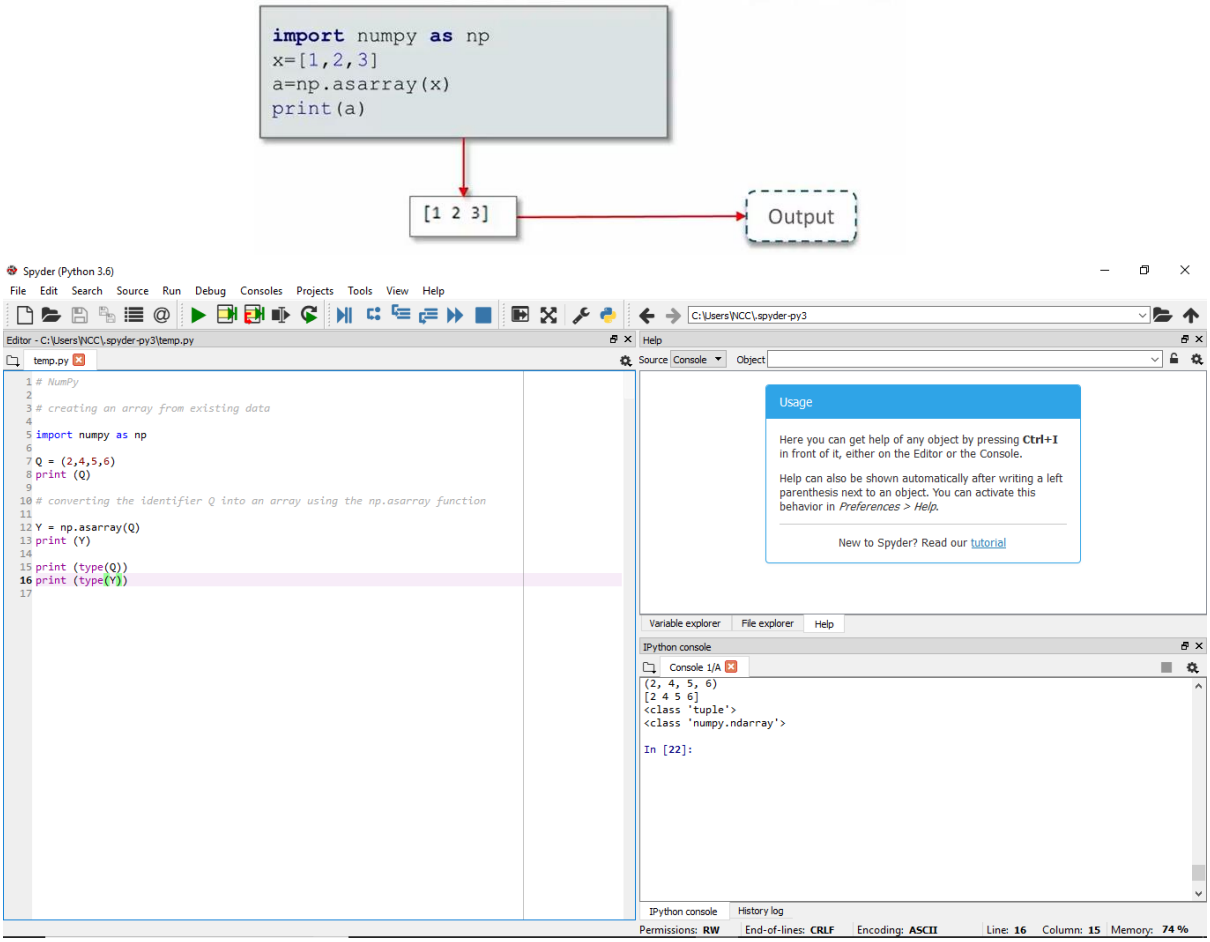
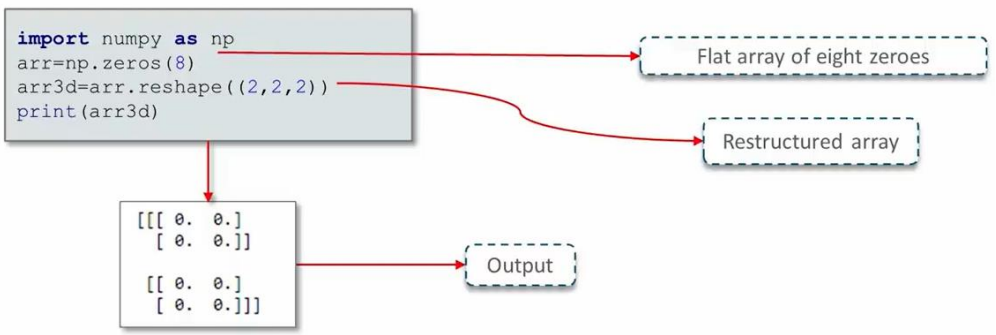


Figure 21 Creating an Array from Existing Data

2.1.4. Restructuring a NumPy Array

A linear array of any number of elements can be restructured as desired. Here will make an instance, converting a linear array of 8 elements into 2×2×2 3D array consider the case of transpose in matrix.



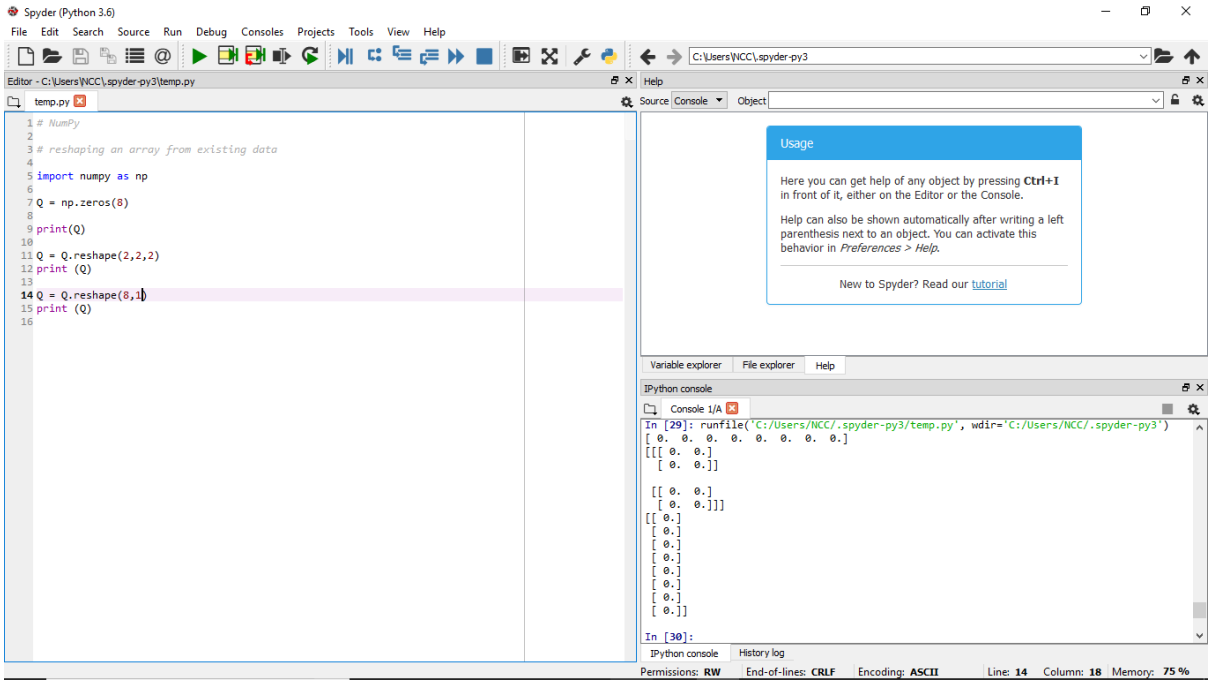


Figure 22 Restructuring a NumPy Array

The restructured array can be returned to its initial state by using the function ravel.

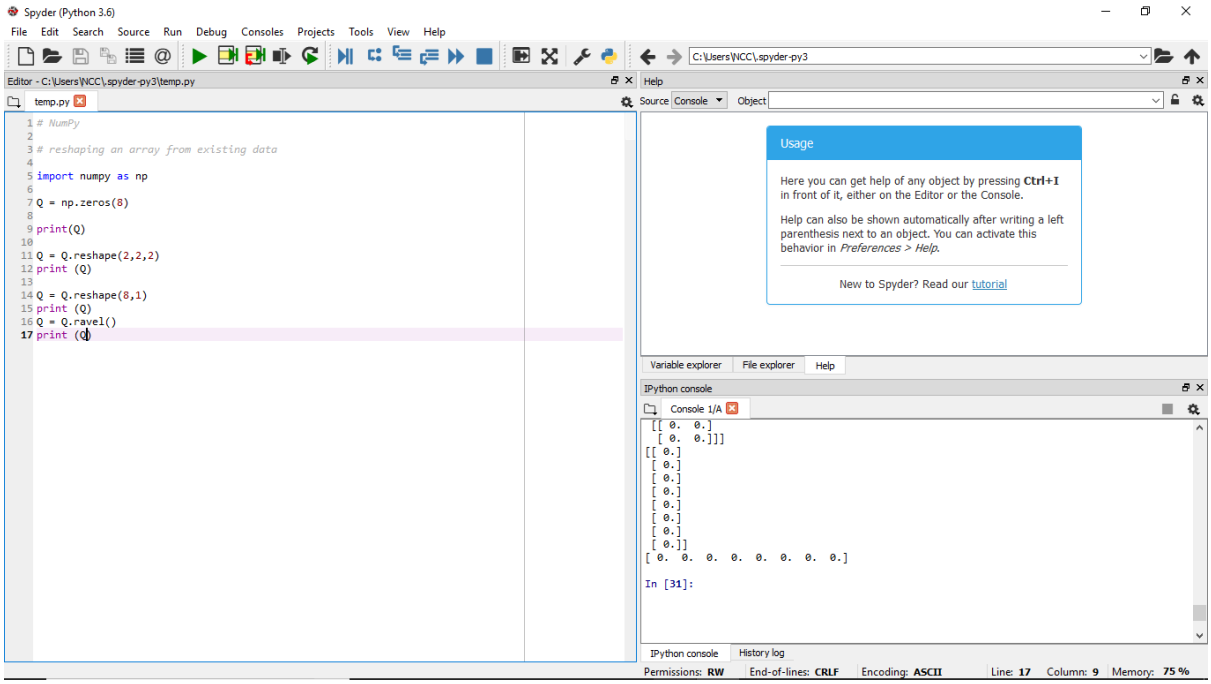


Figure 23 using the ravel function

2.1.5. Indexing a NumPy Array

Indexing in NumPy array is identical to Python's indexing scheme

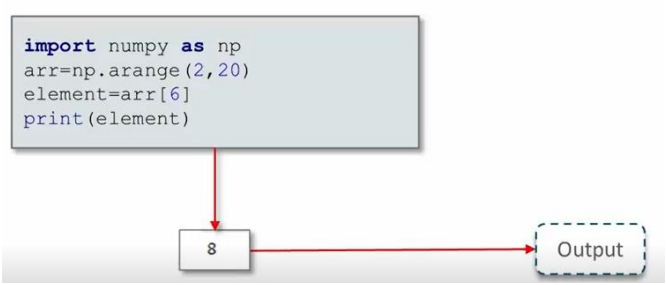


Figure 24 Indexing NumPy Array

Slicing a NumPy array, the slice object is constructed by providing the initial, final and the step parameter to slice ()

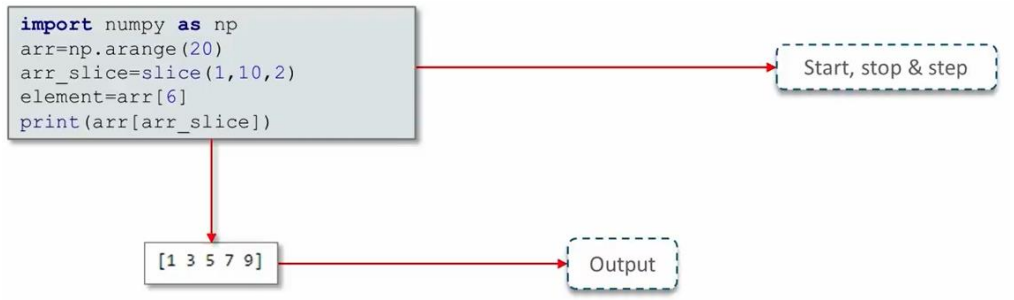


Figure 25 Slicing a NumPy Array

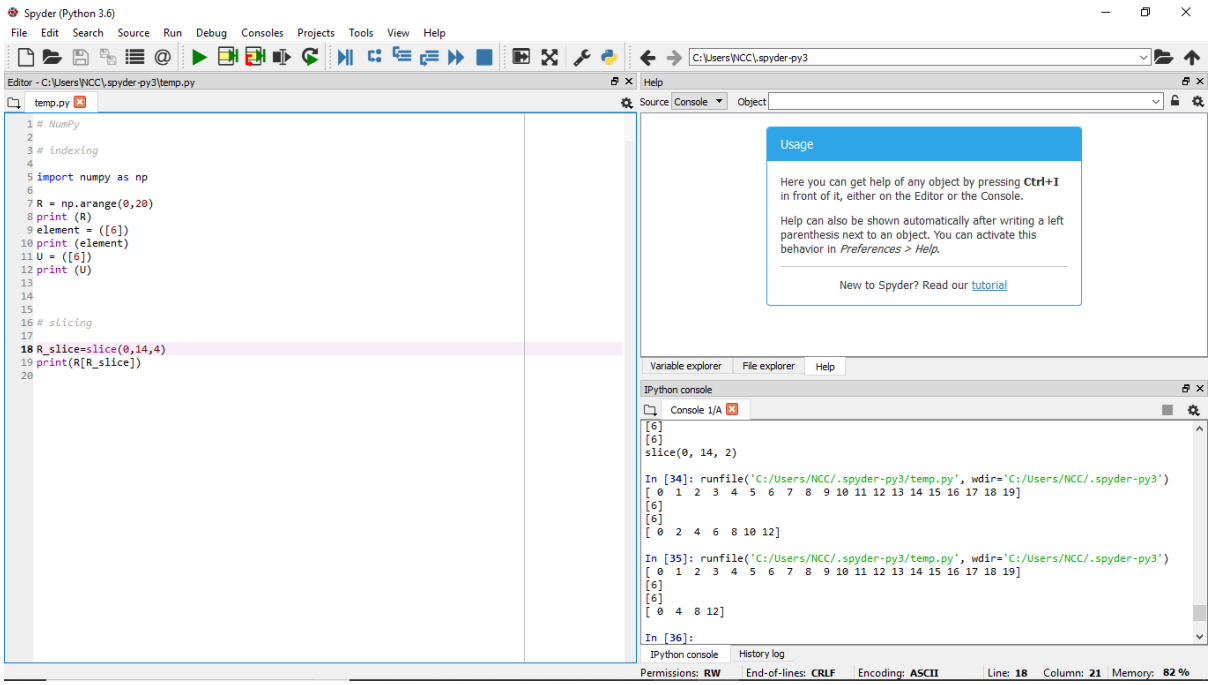


Figure 26 Indexing and Slicing of NumPy Array

Other slicing method possible in Python is presented in Fig.28

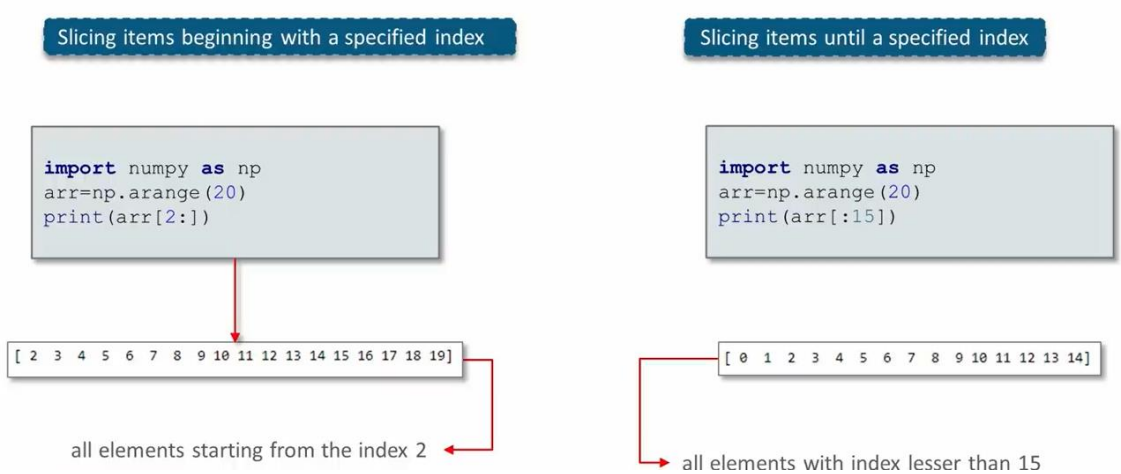


Figure 27 Other Slicing Methods

NumPy array attributes can be derived by following the step in Fig. 29

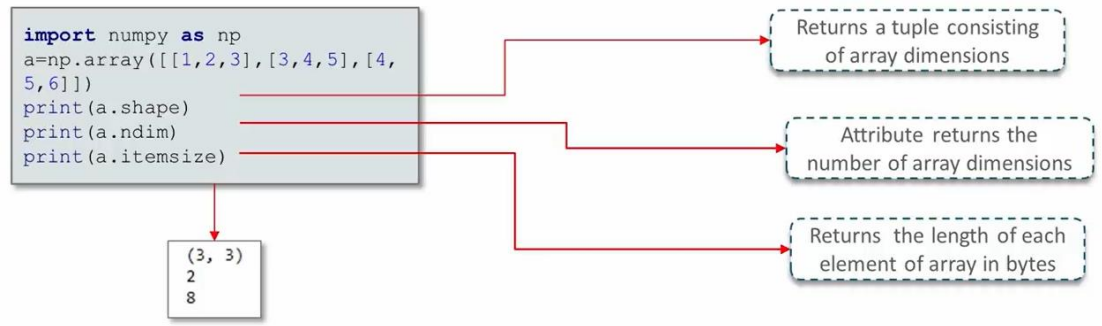
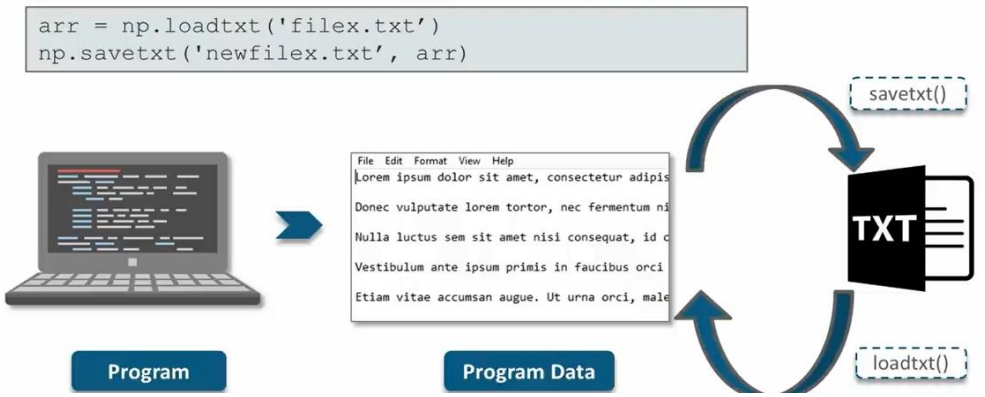


Figure 28 NumPy Array Attributes

2.1.6. Reading and writing from files

2.1.6.1.Reading and Writing from Text file

NumPy provides the option of importing data from files directly into ndarray using the loadtxt function. The savetxt function can be used to write data from an array into text file.



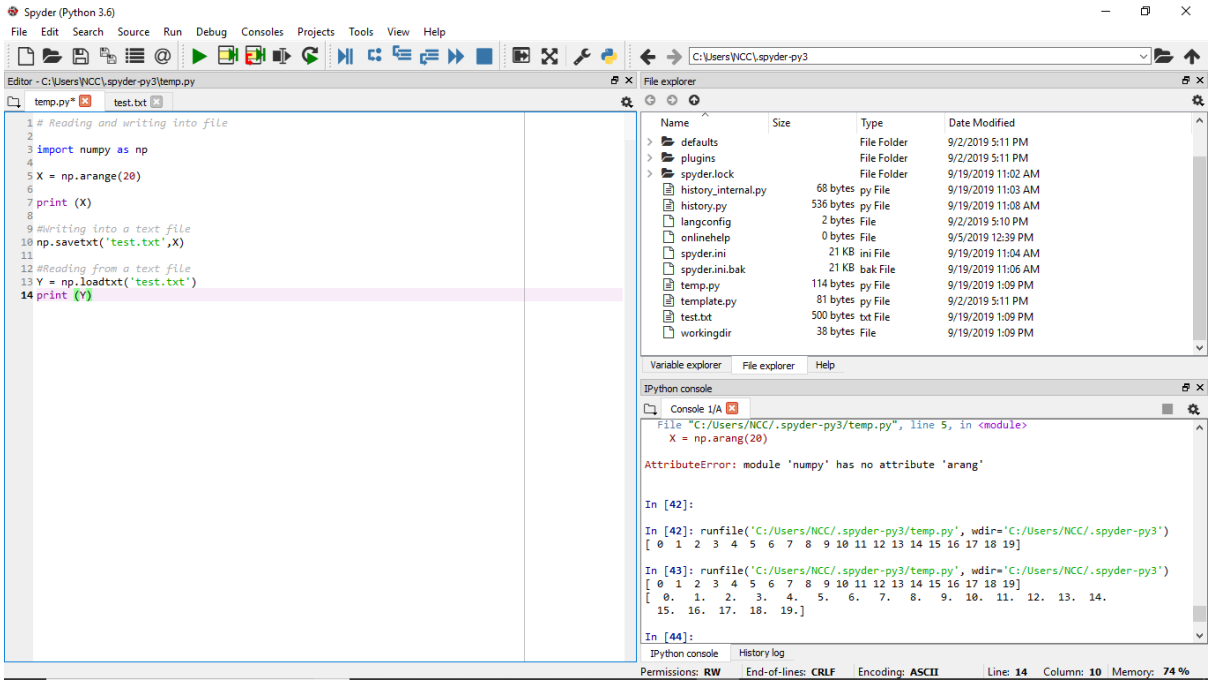
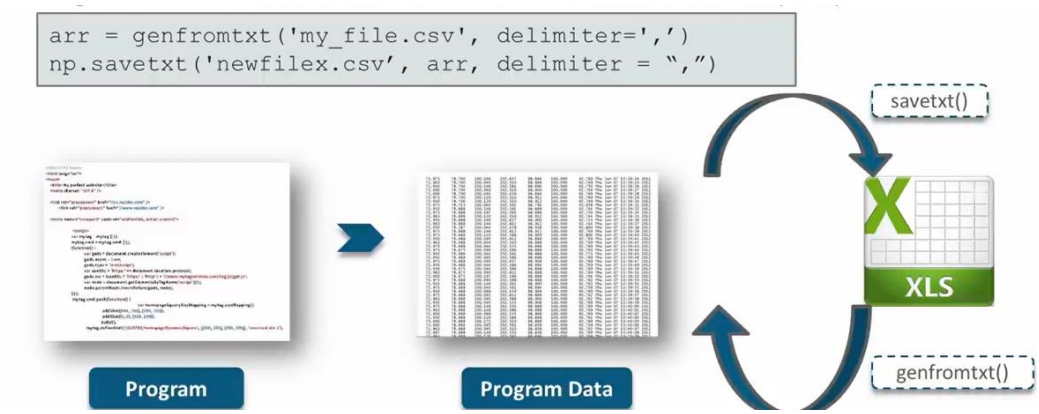


Figure 29 Writing and Reading from a Text file

2.1.6.2.Reading and Writing from CSV file

NumPy arrays can be dumped into CSV using the savetxt and the comma delimiter and the CSV file can be read into NumPy array using the genfromtxt function.



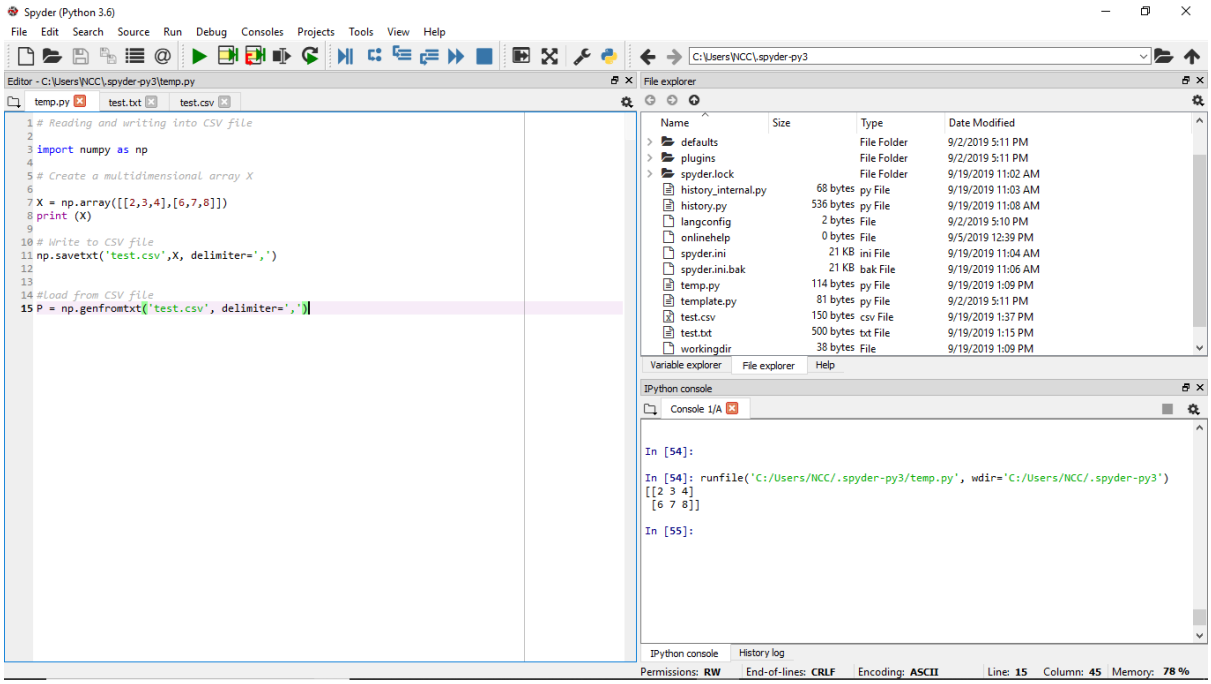


Figure 30 Writing and Reading from CSV file

2.2. Introduction to Pandas Library

Pandas is an open-source Python library which provides efficient, easy-to-use data structure and data analysis tools. Pandas is built on NumPy and the name Pandas is derived from "Panel Data" = an Econometrics form multidimensional data. The Pandas library is well suited for several kind of data like:

1. Tabular data with heterogeneously-typed columns
2. Ordered and unordered time series data.
3. Arbitrary matrix data with row and column labels
4. Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into Pandas data structure

2.2.1. Data Structure in Pandas

Pandas provides three data structures; Series, DataFrames, Panels all to which are built on NUmPy array. Note all data structure in Pandas are value-mutable.

Data Structure	Dimension	Description
Series	1	Labeled, homogenous array of immutable size
DataFrames	2	Labeled, heterogeneous typed, size-mutable tabular data structure
Panels	3	Labeled size-mutable array

2.2.1.1. Series

Series is a single dimensional array structure that stores homogenous data i.e., data of single type. All element in the Series are value-mutable but size-immutable.

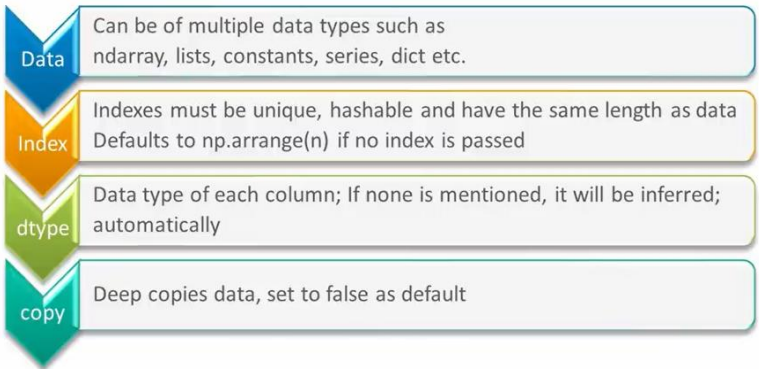


Fig. 31 presents the creation of a Series using the Pandas library.

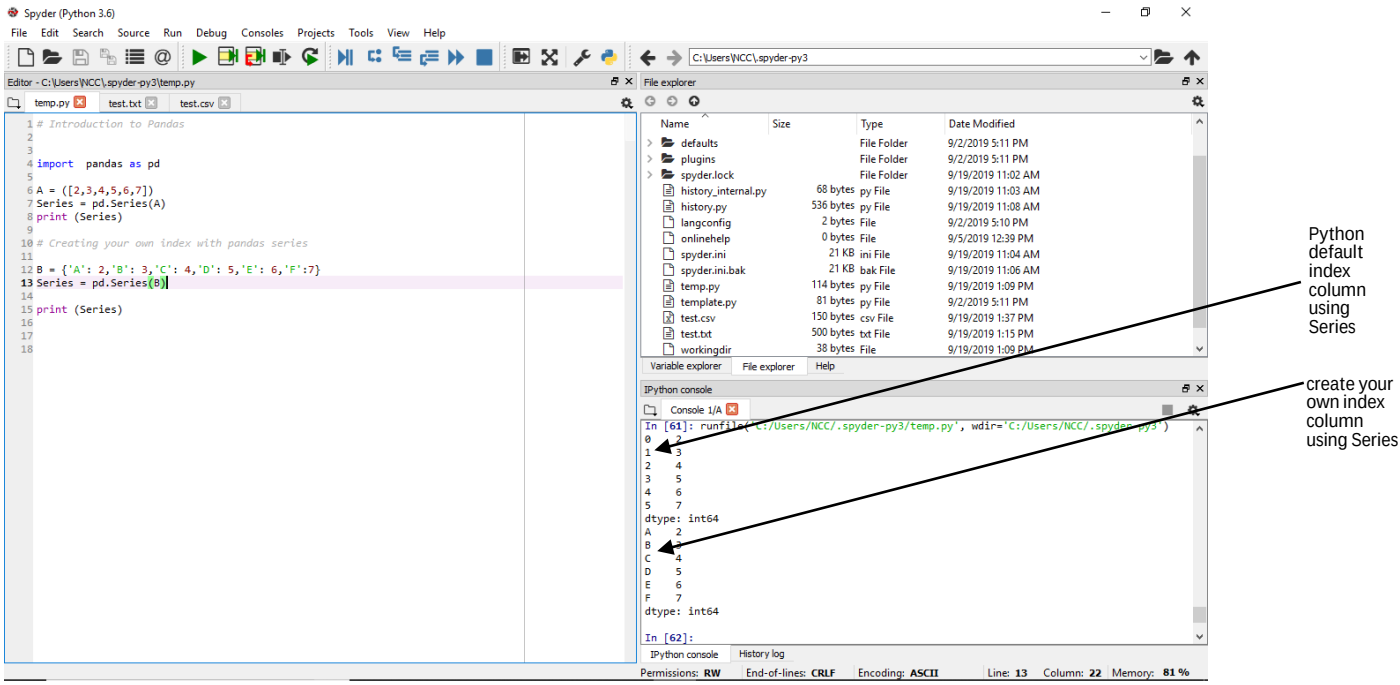


Figure 31 Creating Series Using Pandas Library in Python

We can see from the output console in Fig. 31 that Python automatically added the default index for the data. However, we can set the index by using the dictionary data structure to input the Series. See Fig 31.

Elements within a Series can be accessed using the slicing function as earlier stated in Fig. 27 and 28 however, an example is provided in Fig. 32.

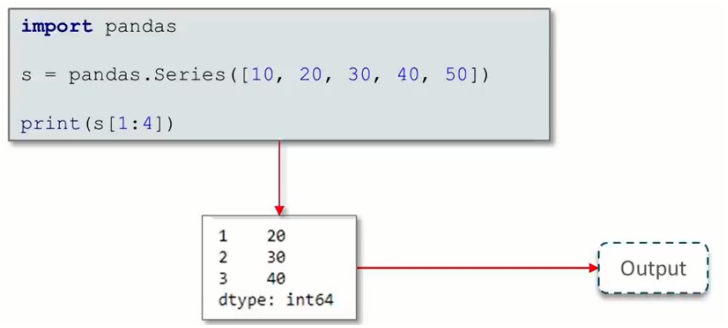


Figure 32 Accessing Data in Series Using Slicing

2.2.1.2.DataFrames

DataFrames is a 2D data structure in which data is aligned in tabular fashion consisting of rows and columns.

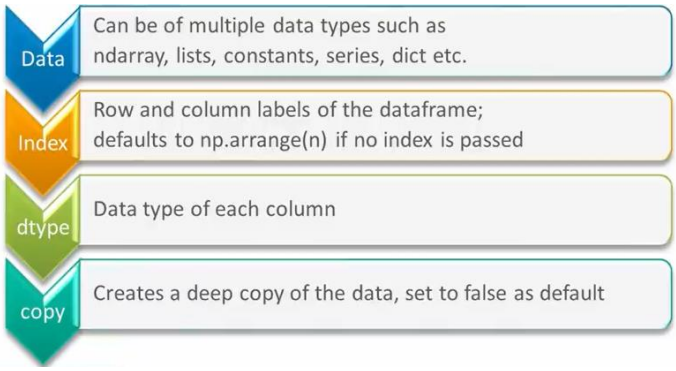


Fig. 33 shows how DataFrame is created. We will notice that Python automatically printed the default index as in the case of Series and a default column name of 0. We can input our own column name by using a dictionary to input the data.

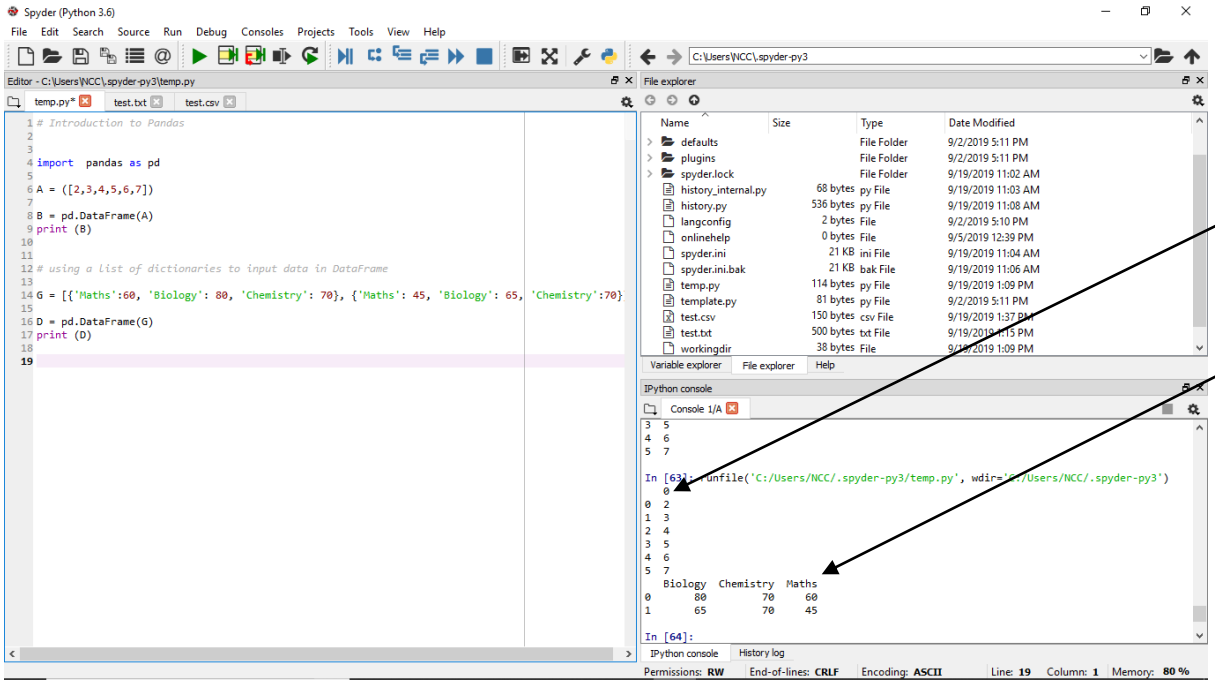


Figure 33 Creating DataFrame

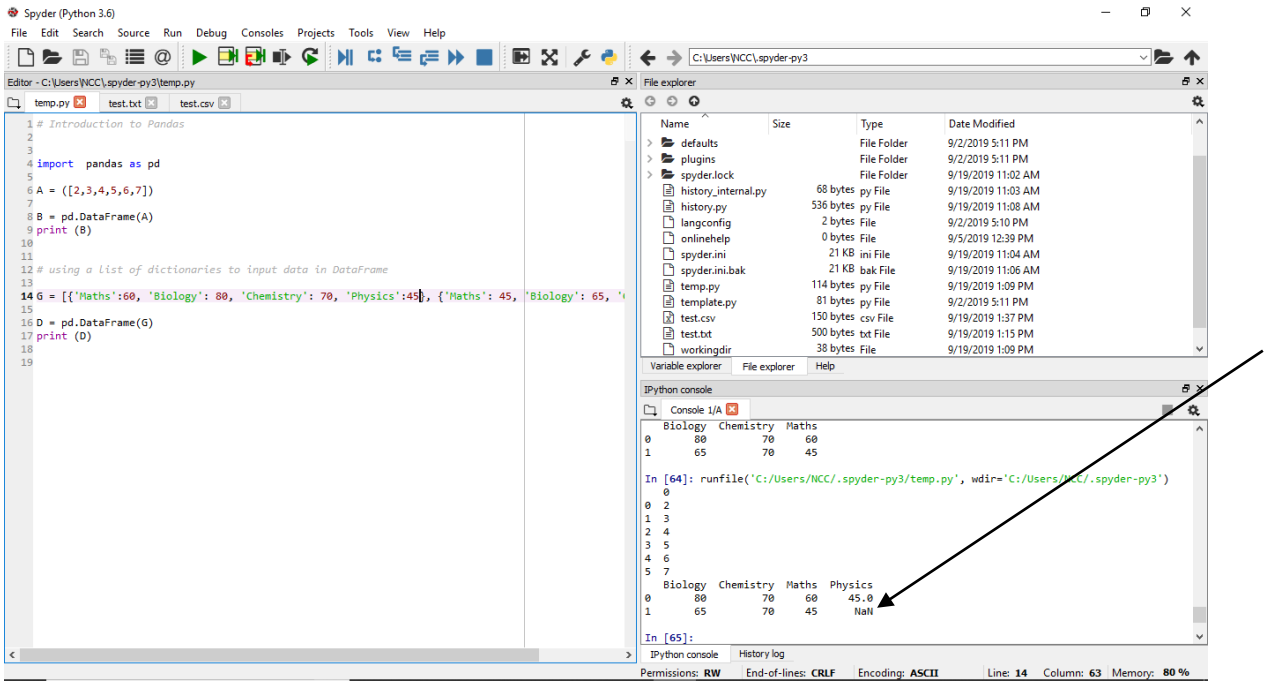


Figure 34 The NaN case

We can also specify our own index see Fig. 35

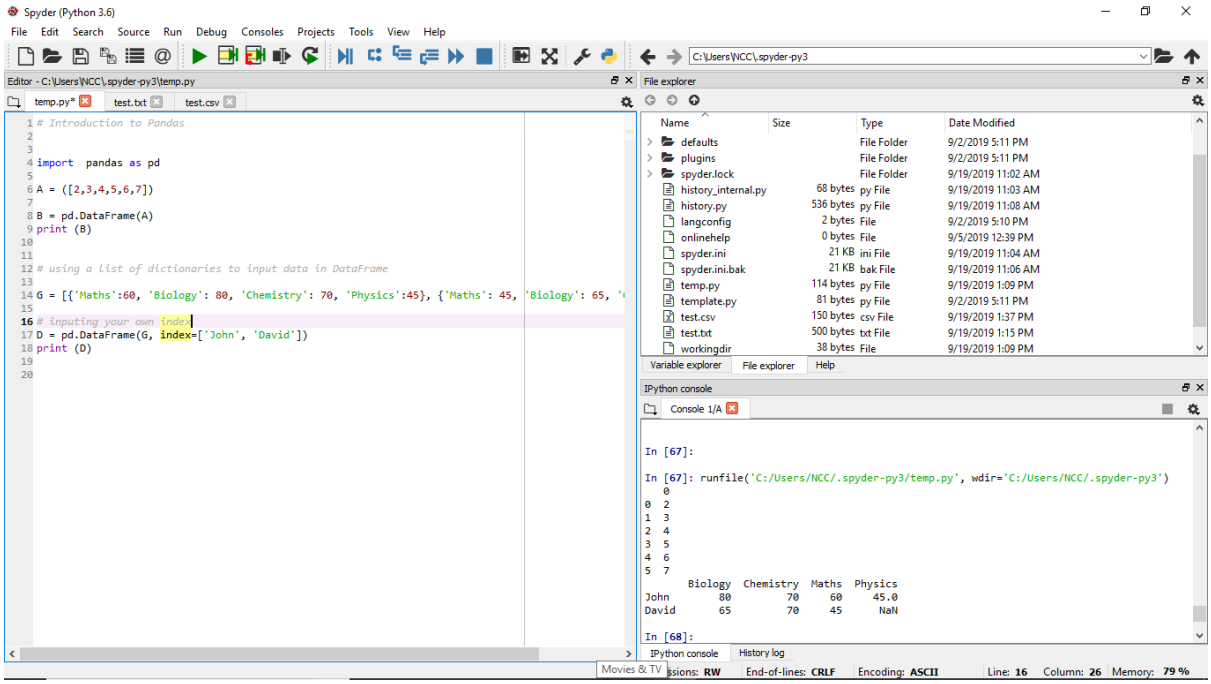


Figure 35 Using your own Index in Data Frame

Series can be converted to DataFrame see Fig. 36

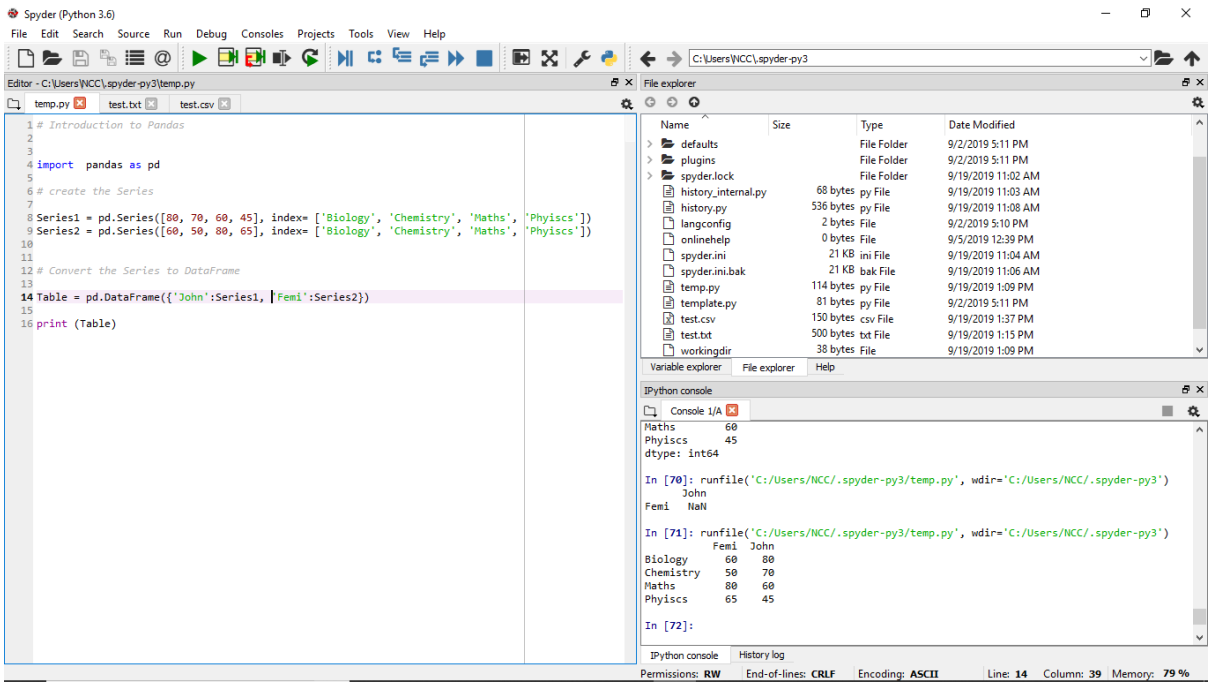


Figure 36 Converting Series to DataFrame

Column addition and deletion is possible with DataFrame

1. To add column to a DataFrame the data must be passed as Series

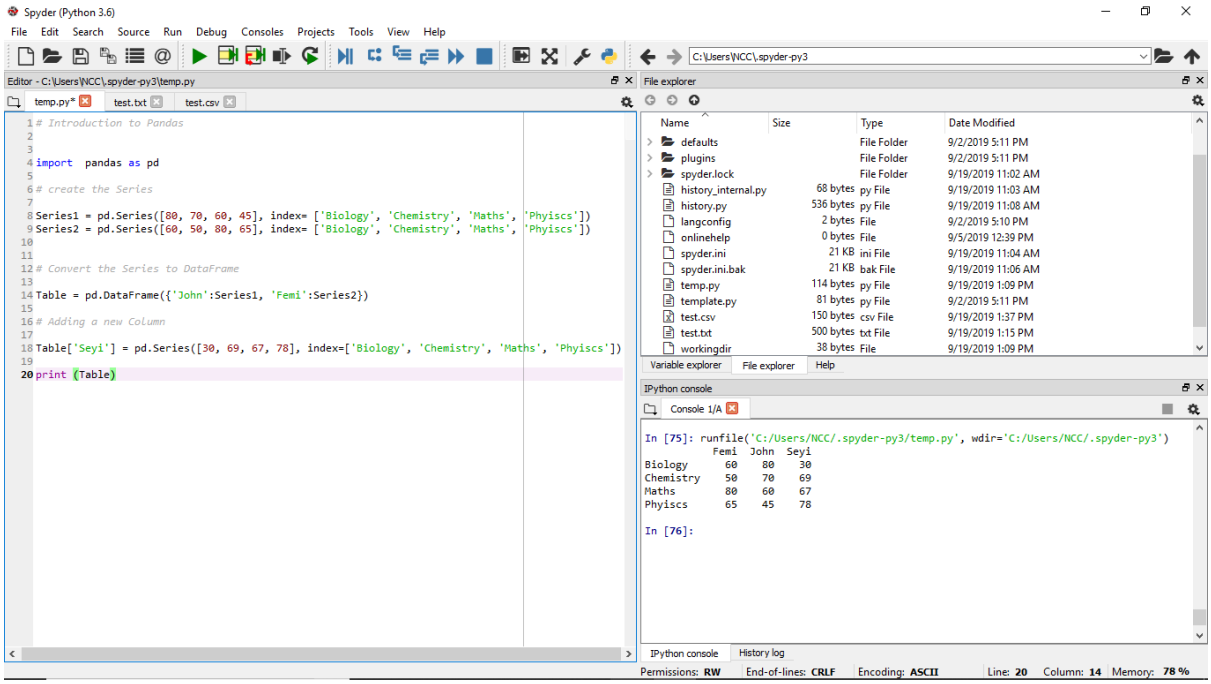


Figure 37 Adding a Column to a DataFrame

2. Deleting a Column from a DataFrame is done by using the del function

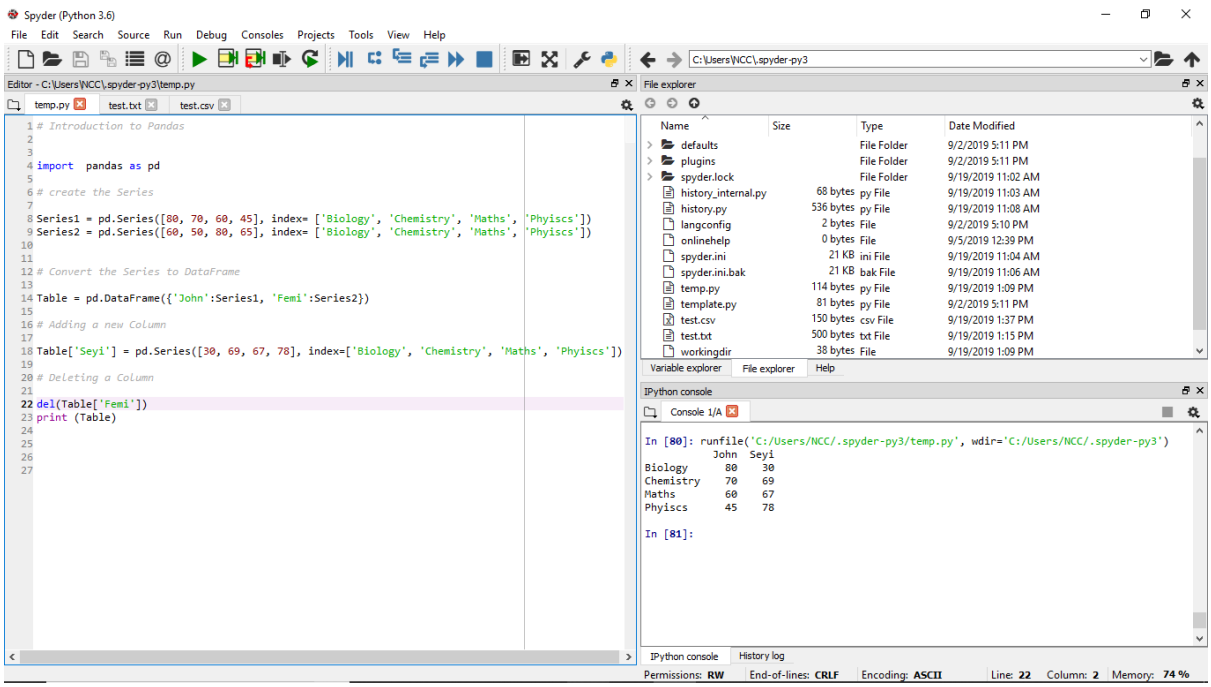


Figure 38 Deleting a Column from a DataFrame

3. We can also use the pop function to view on the column of interest. For instance, let view the result of Femi only

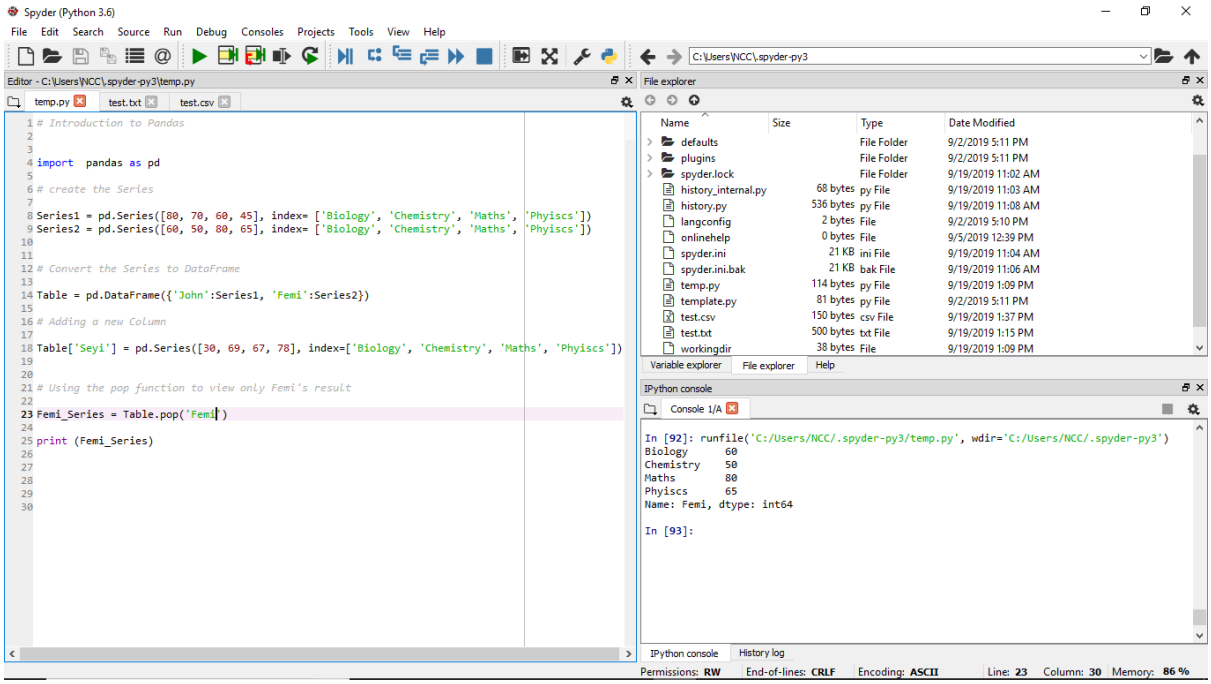


Figure 39 Using the pop function

Row addition and deletion is possible with DataFrame

1. Rows can be selected in DataFramenby passing the row label to the loc[] function. Alternatively, we can pass in the row index into the iloc[] function.

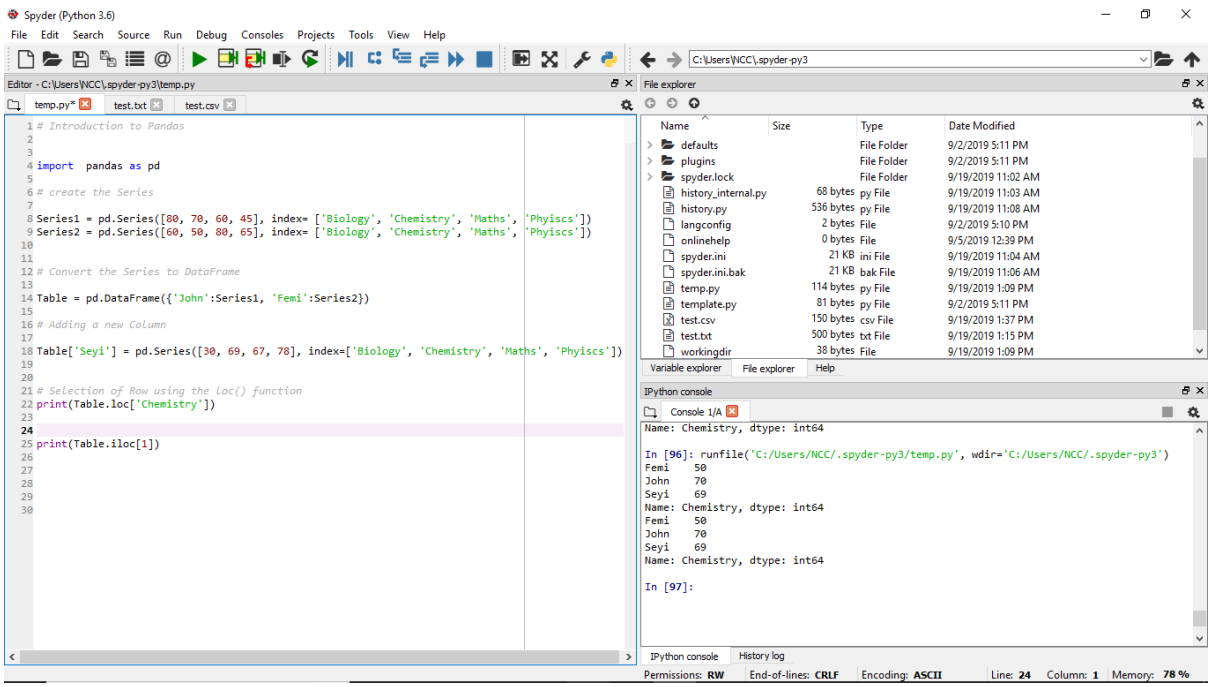
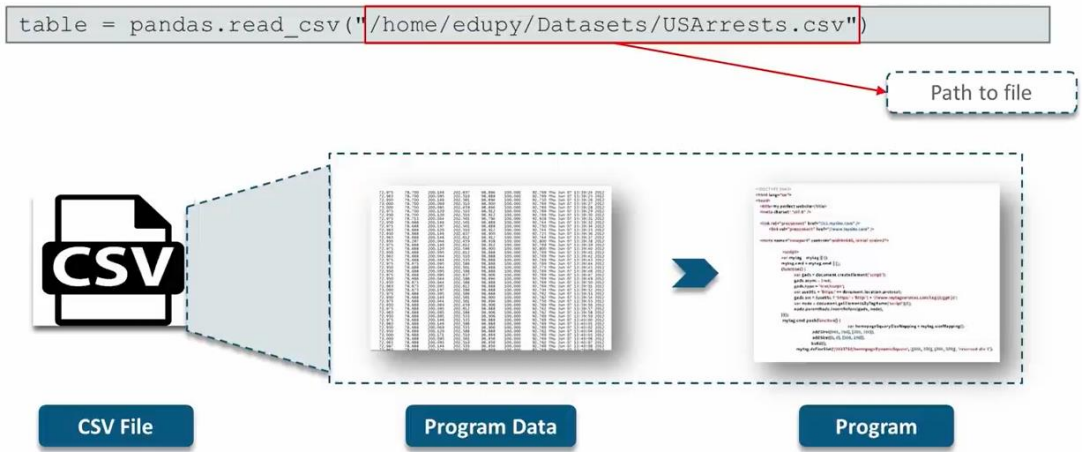


Figure 40 Using the loc function for row selection

2.2.2. Importing and Exporting data using Pandas

Data can be loaded into DataFrame from input data stored in CSV format using the read_csv() function



1. Create a CSV file in Python
2. Then read the file

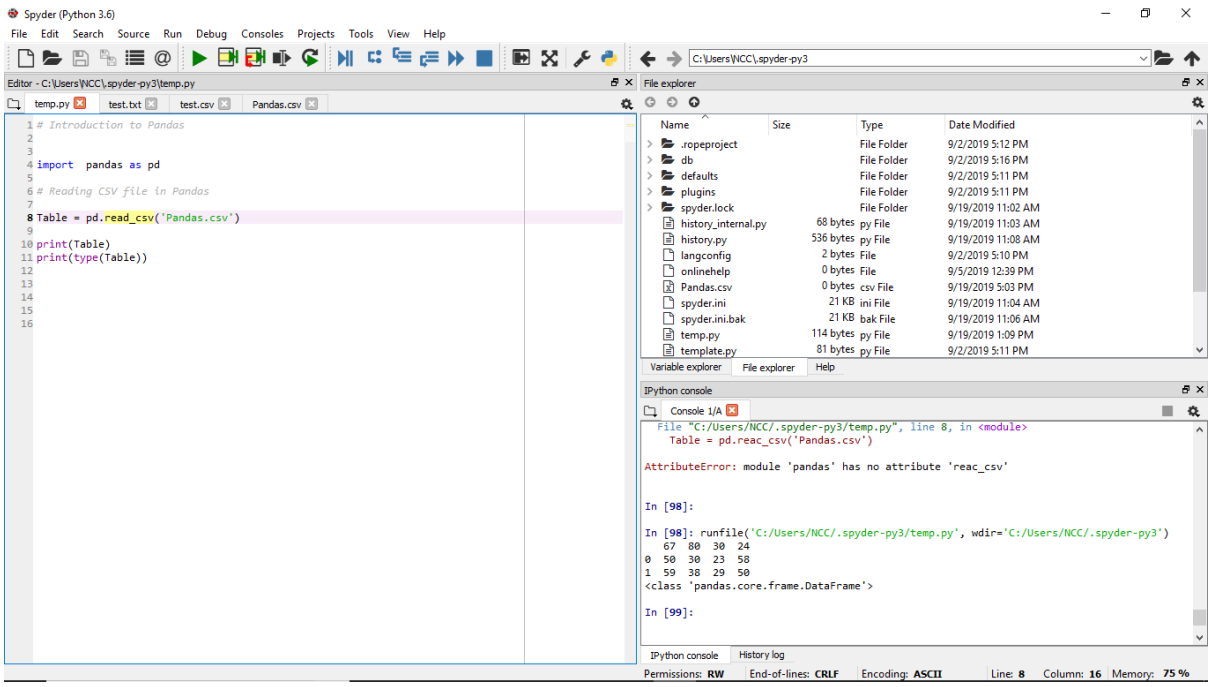
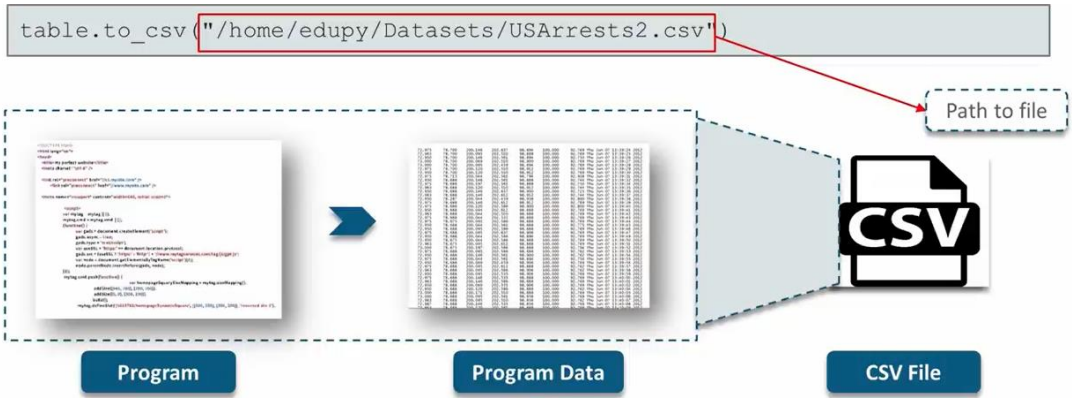


Figure 41 Reading CSV file to Pandas Library

Data present in a given DataFrame can be written to CSV file using the to_csv() function. If the specified path does not exist, a file of the same name is automatically created.



Similarly, we can write to and read from excel file using Pandas

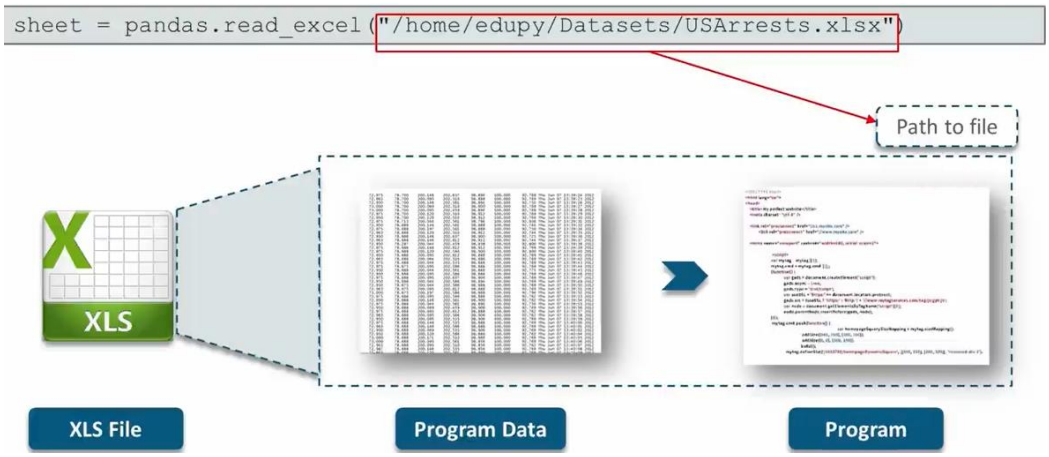


Figure 42 Reading from Excel file

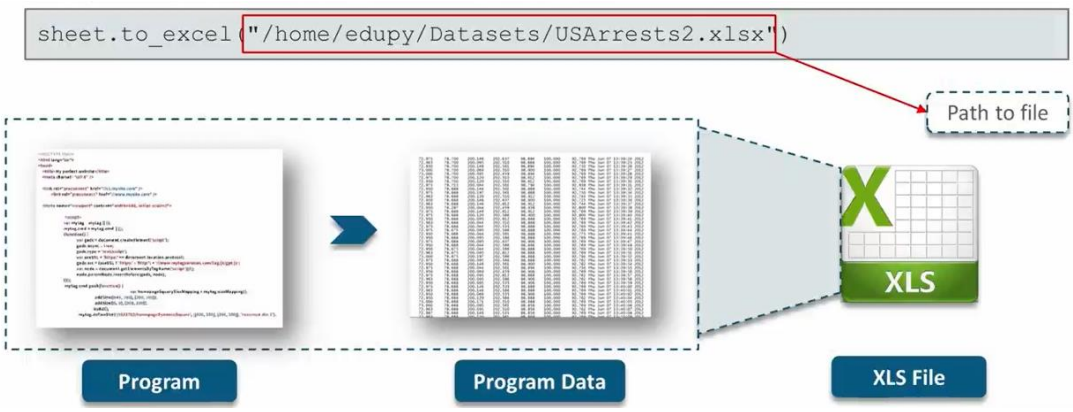


Figure 43 Writing to Excel file

2.3. Introduction to Data Visualization and Matplotlib

Matplotlib is a Python library specifically designed for development of graphs, charts etc., in order to provide interactive data visualization.

2.3.1. Plotting in Matplotlib

Plotting ins Python is done by importing the matplotlib.pyplot library as plt and pass in the argument.

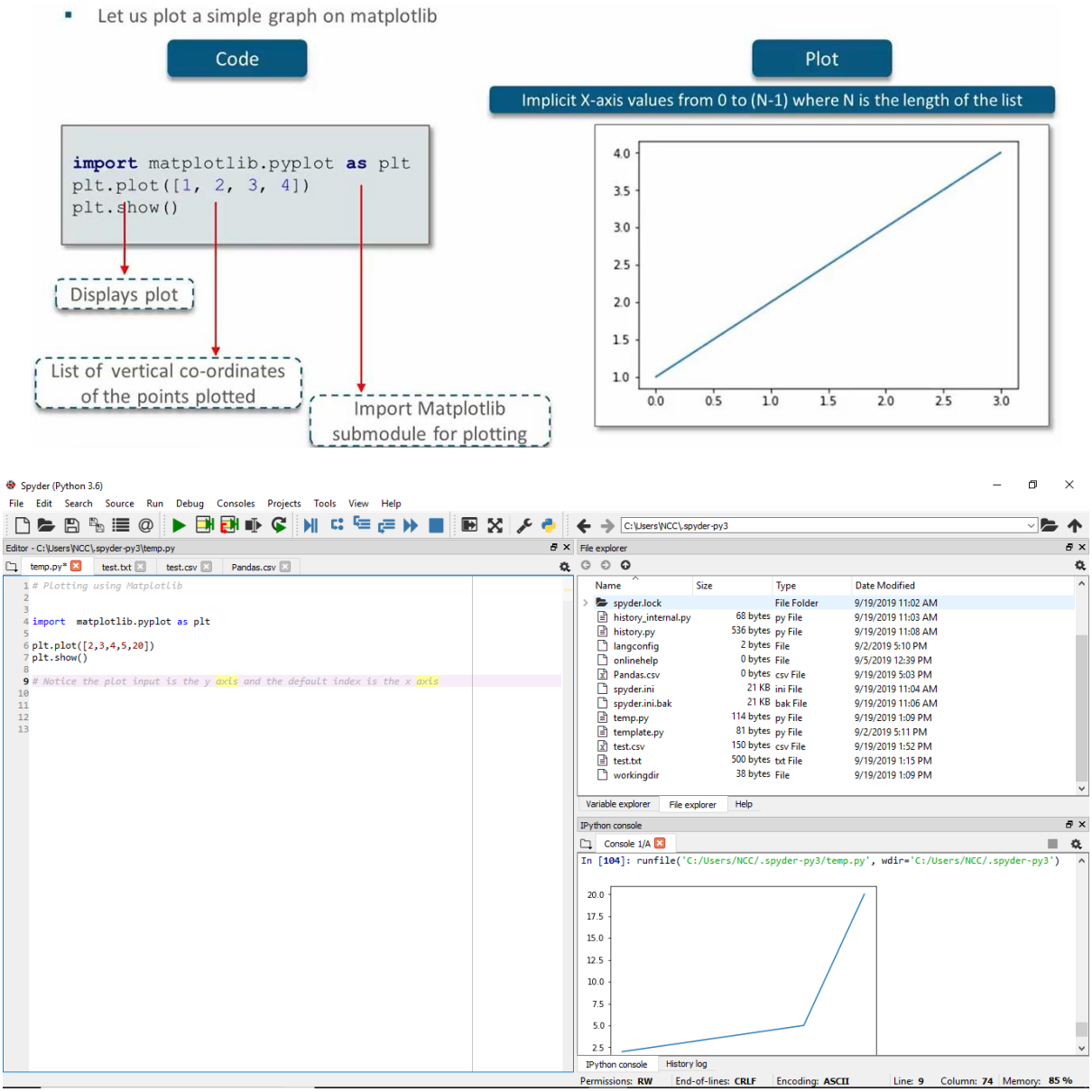


Figure 44 Using Matplotlib Library in Python

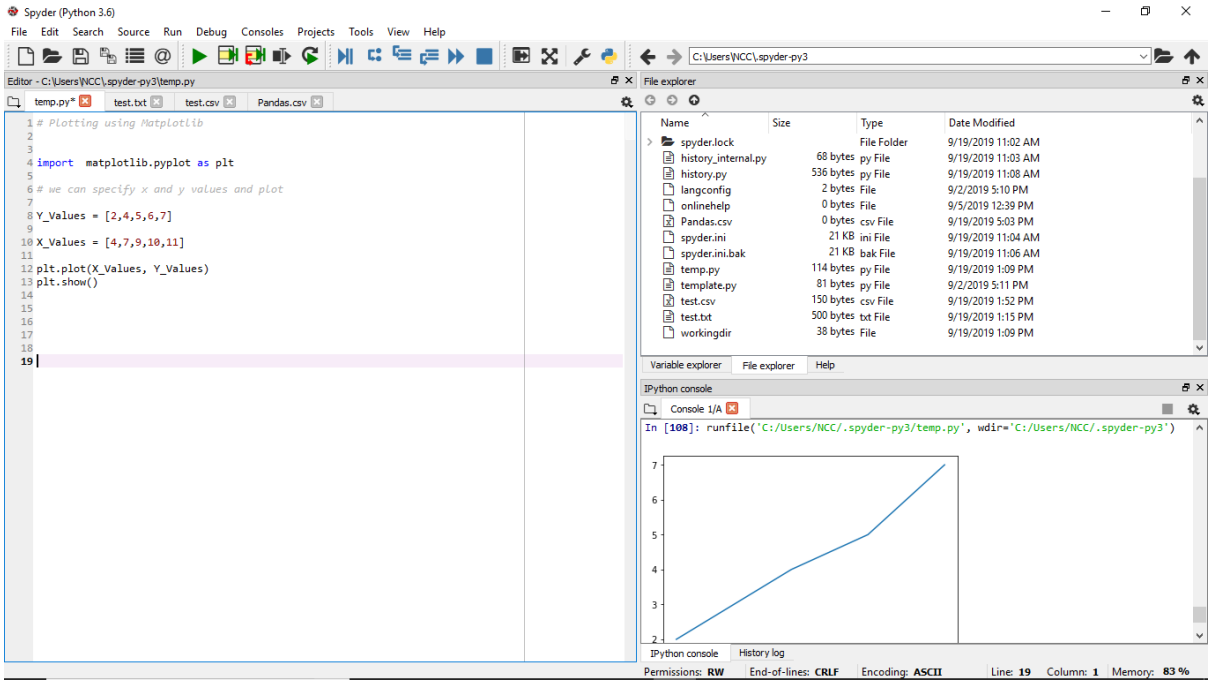
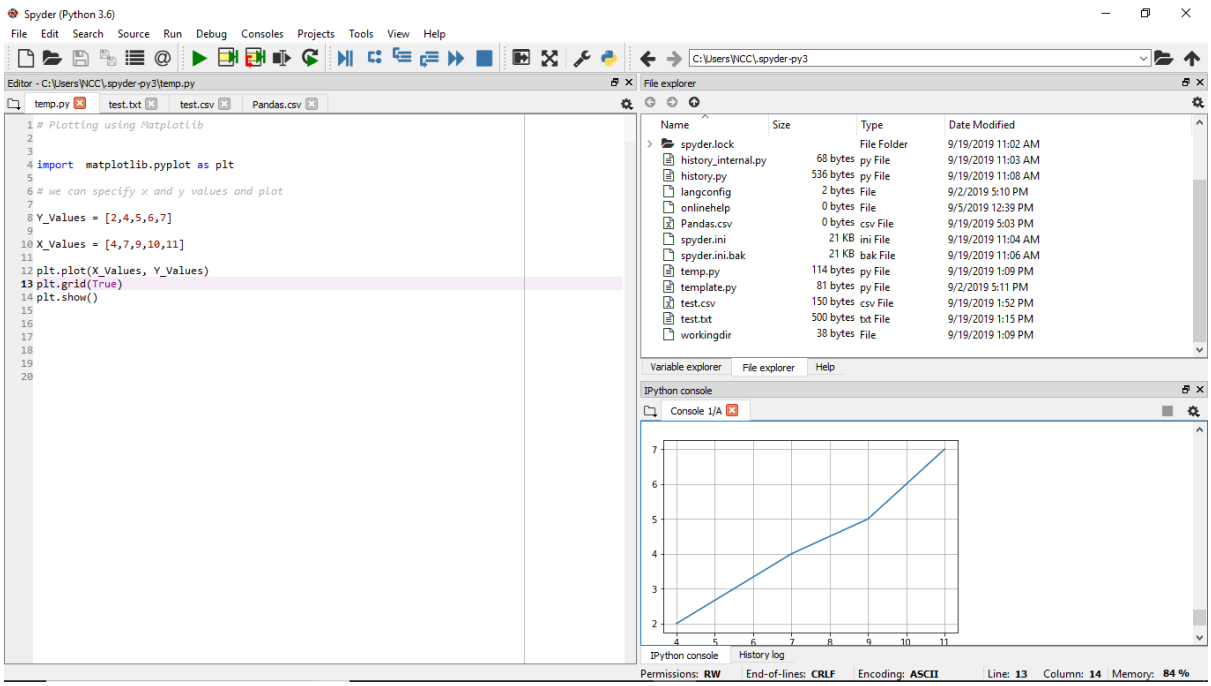


Figure 45 Specifying the x and v values in Matplotlib

We can add the grid by using the *plt.grid(True).*

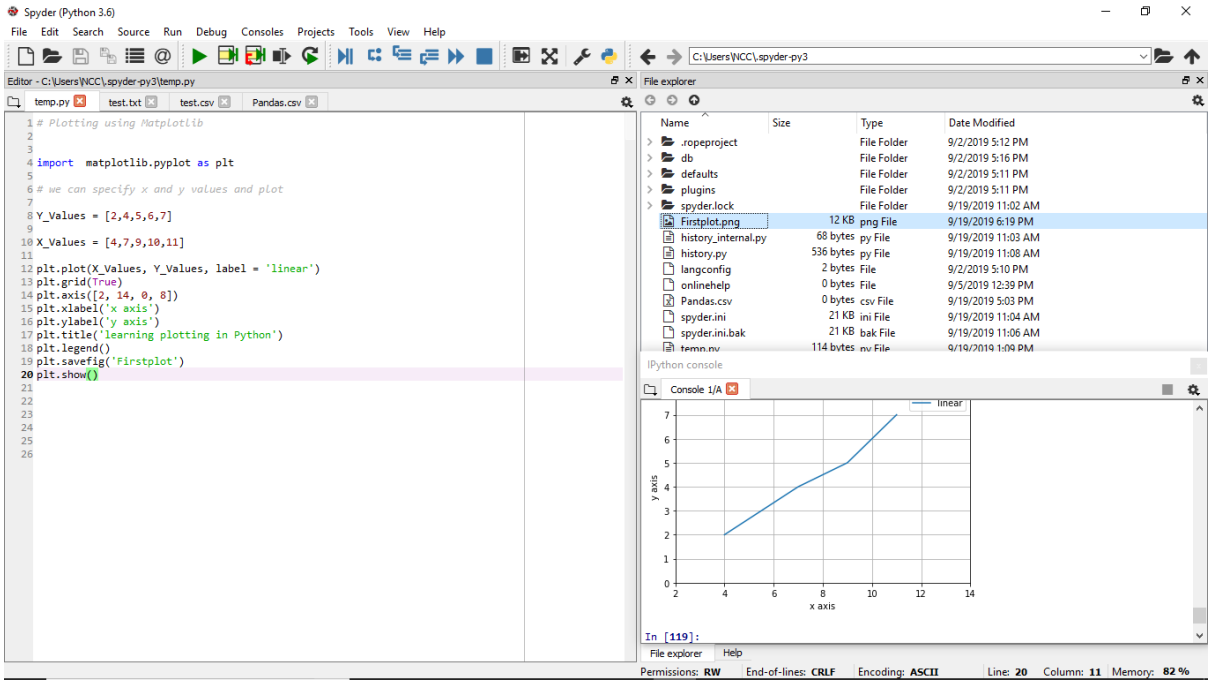


Setting of axis is done by using *plt.axis([xmin, xmax, ymin, ymax])*

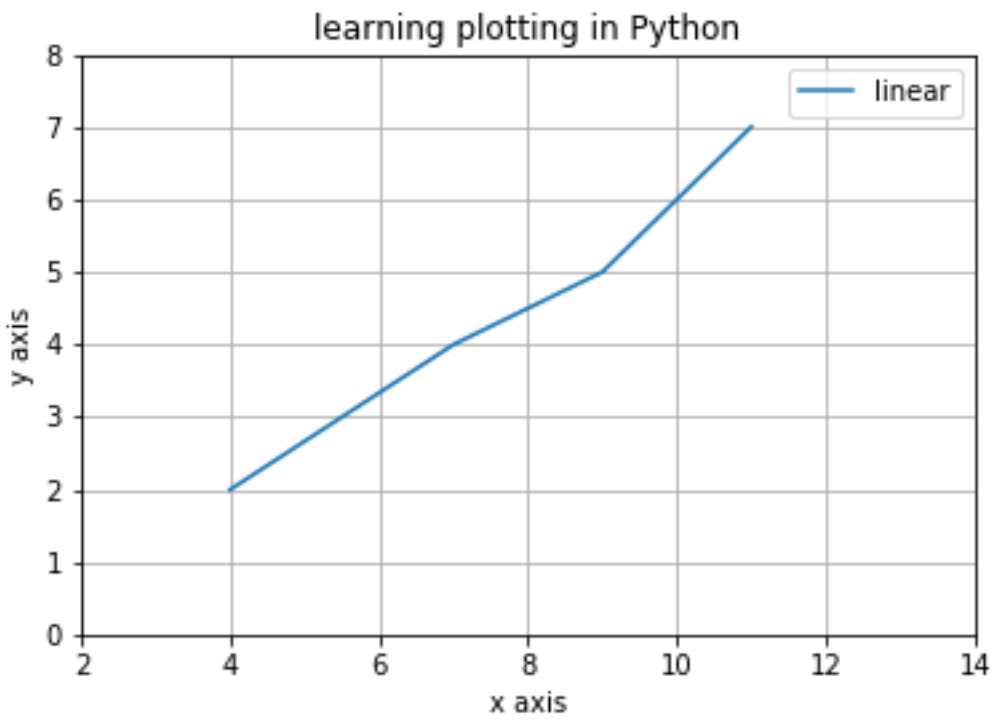
Labels can be added to x and y axis using *plt.xlabel('X axis')*and *plt.ylable('Y axis')*

Title can be added using *plt.title('learning plotting in Python')*

Legend can be set using the legend function *plt.legend()*



We save the plot by using the *plt.savefig('file name')*

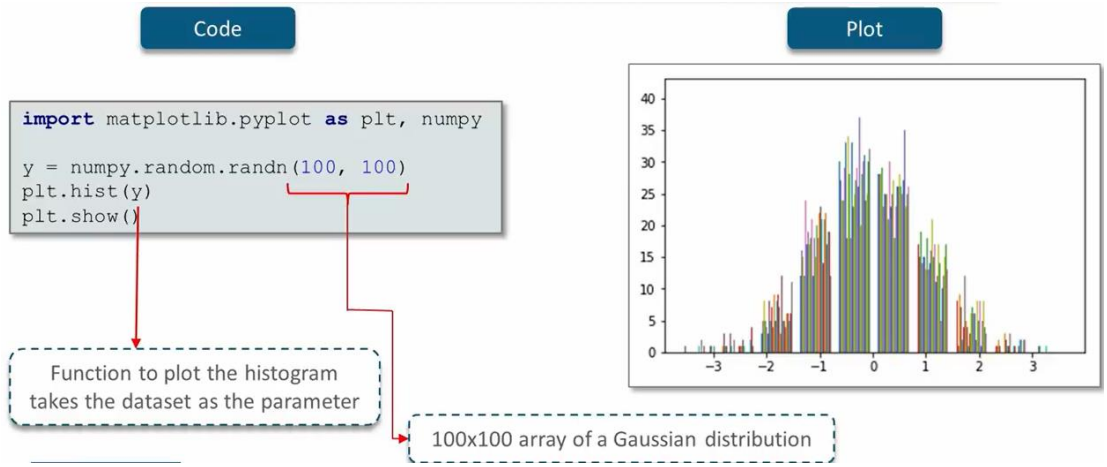


2.3.2. Plot Types

There are several plot formats for visualizing information in Python such as Histogram, Scatter Plot, Bar Graph and Pie Chart. We will show here how to demand any and other plot format in Python.

1. Histogram

Histogram describes the information of a variable over a range of frequencies or values.



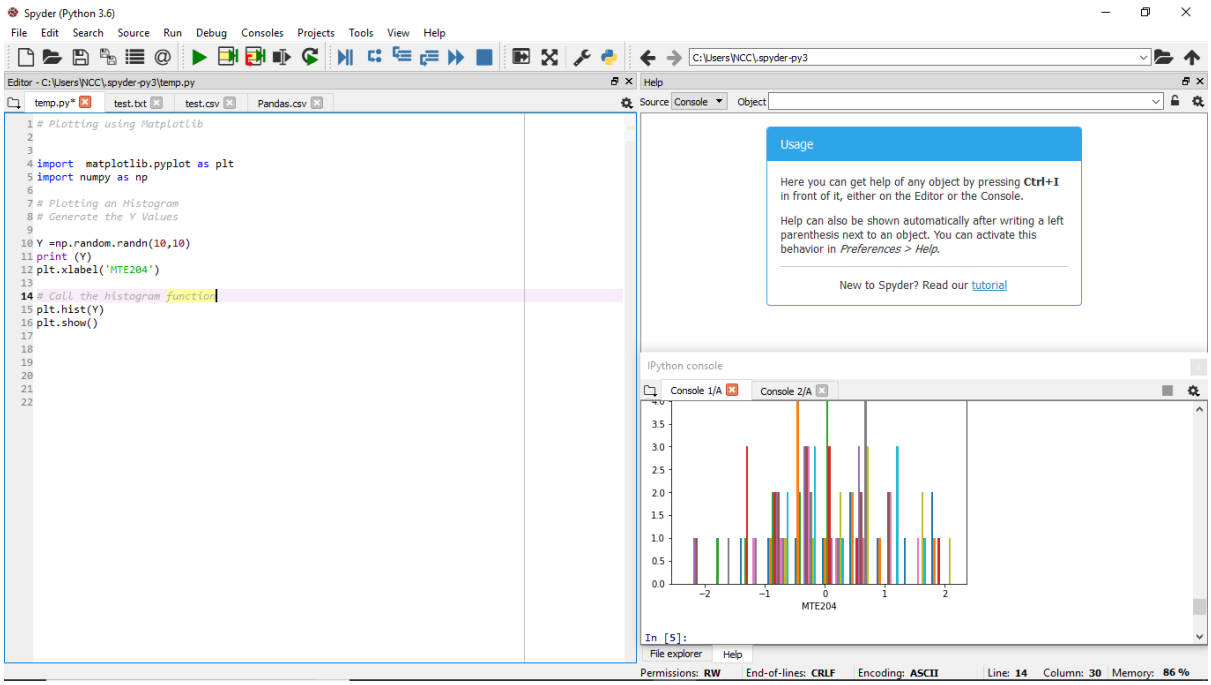


Figure 46 Plotting an Histogram in Python

2. Bar Graph

We create two arrays; the first array is the midpoint of the face of every bar i.e., where the midpoint of the bar graph should be. The second array is the height of the successive bar graph.

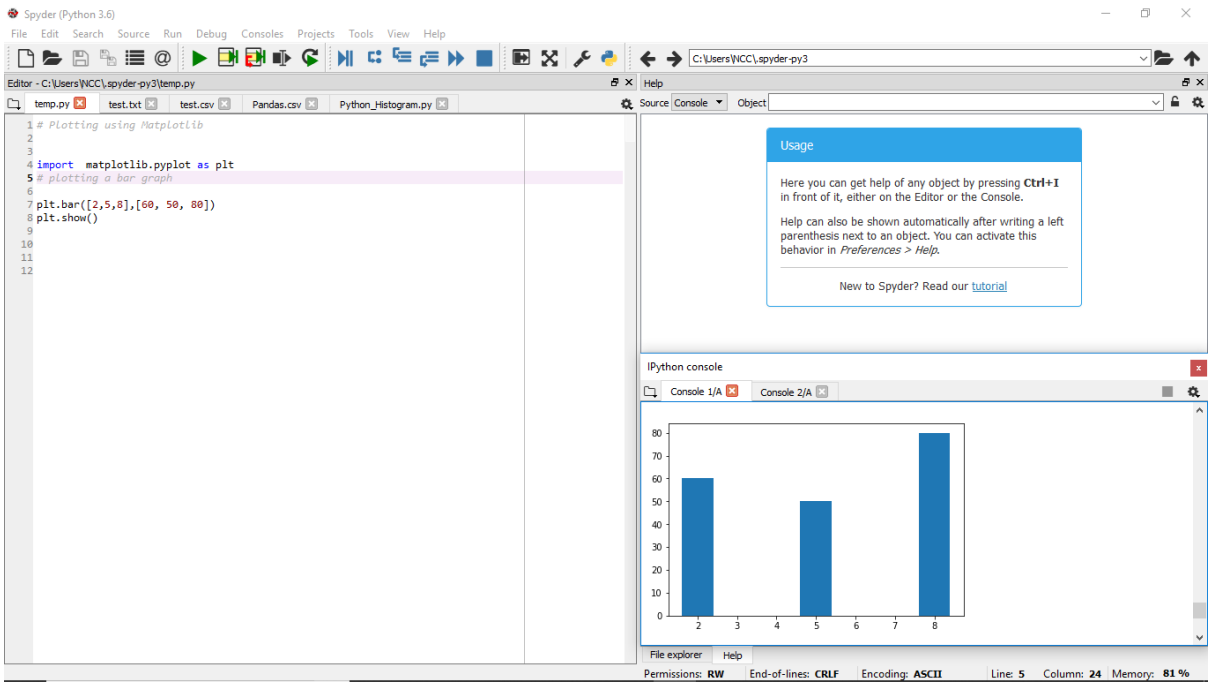
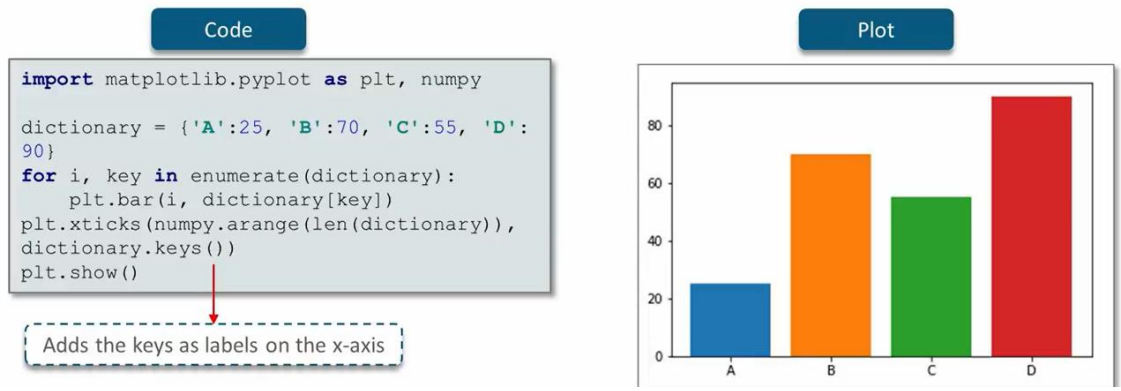


Figure 47 Bar graph in Python

We Can Plot a Dictionary Using Bar Chart



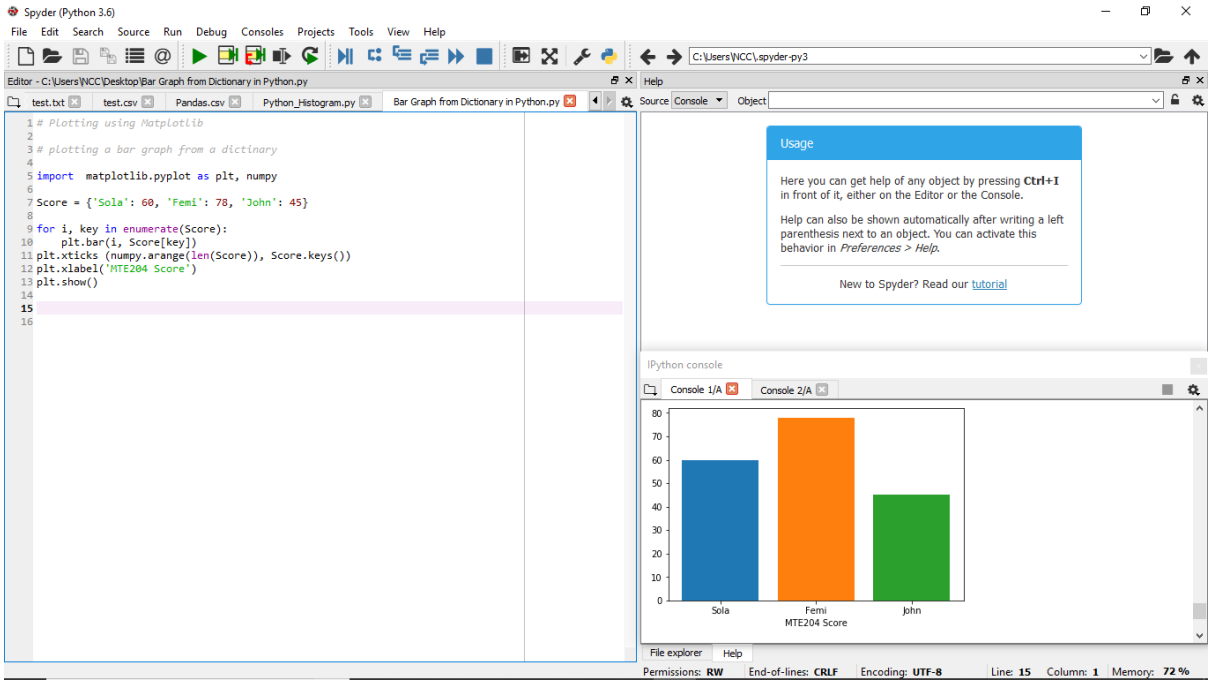


Figure 48 Plotting Bar graph from Dictionary

3. Pie Chart

Pie chart is used to compare multiple parts against the whole. For instance, let use a pie chart to visualize how a student spend her income.

Expenses	Amount
Food	5,000
Transport	1,500
Credit card	2,000
Accessory	500

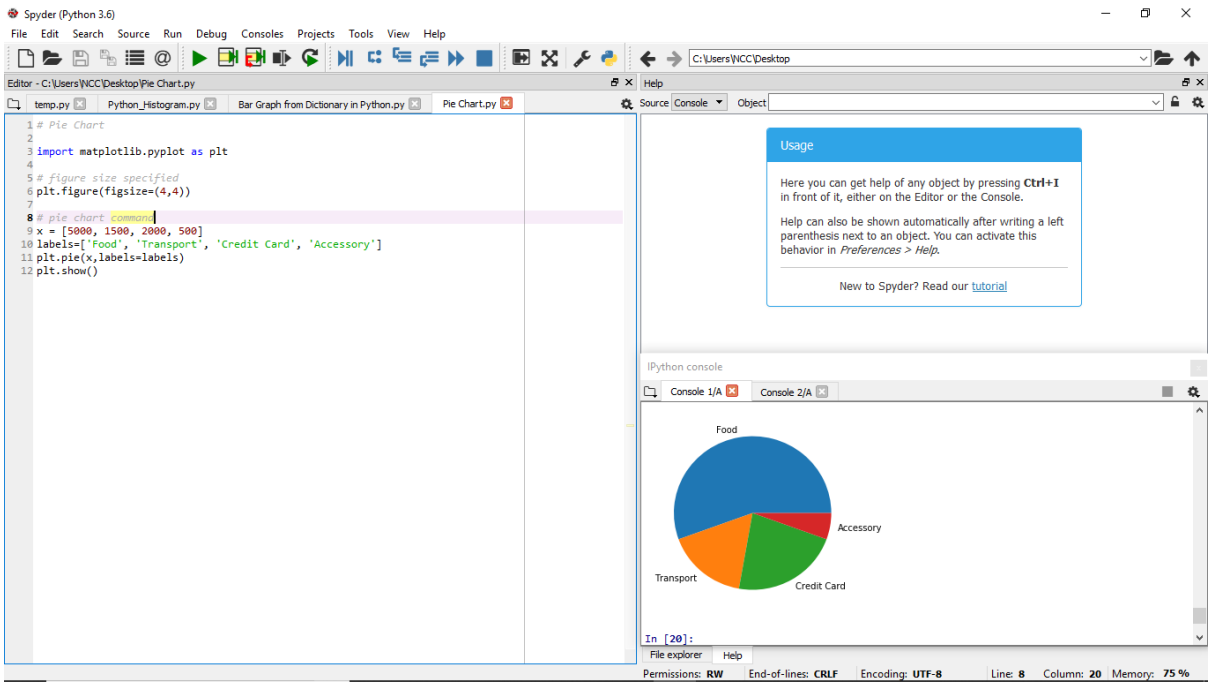
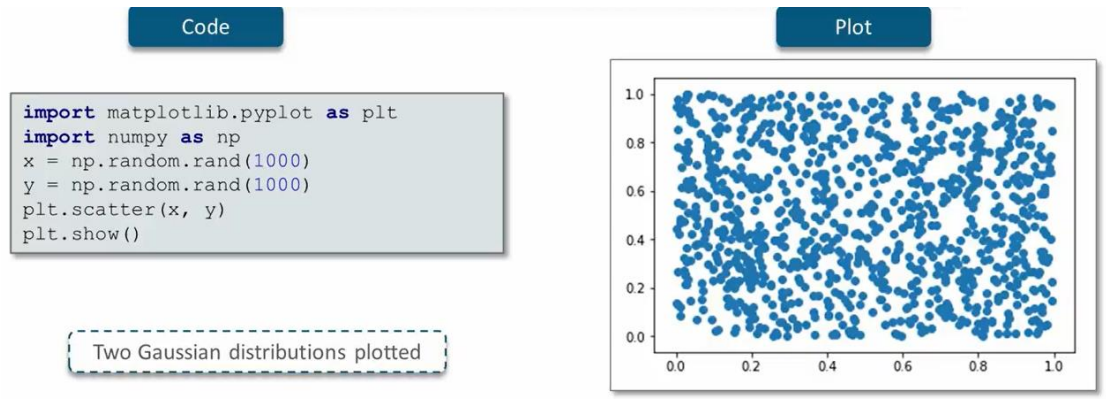


Figure 49 Pie chart in Python

4. Scatter plot

Scatter plots displays the values for two sets of data, visualized as a collection of points.



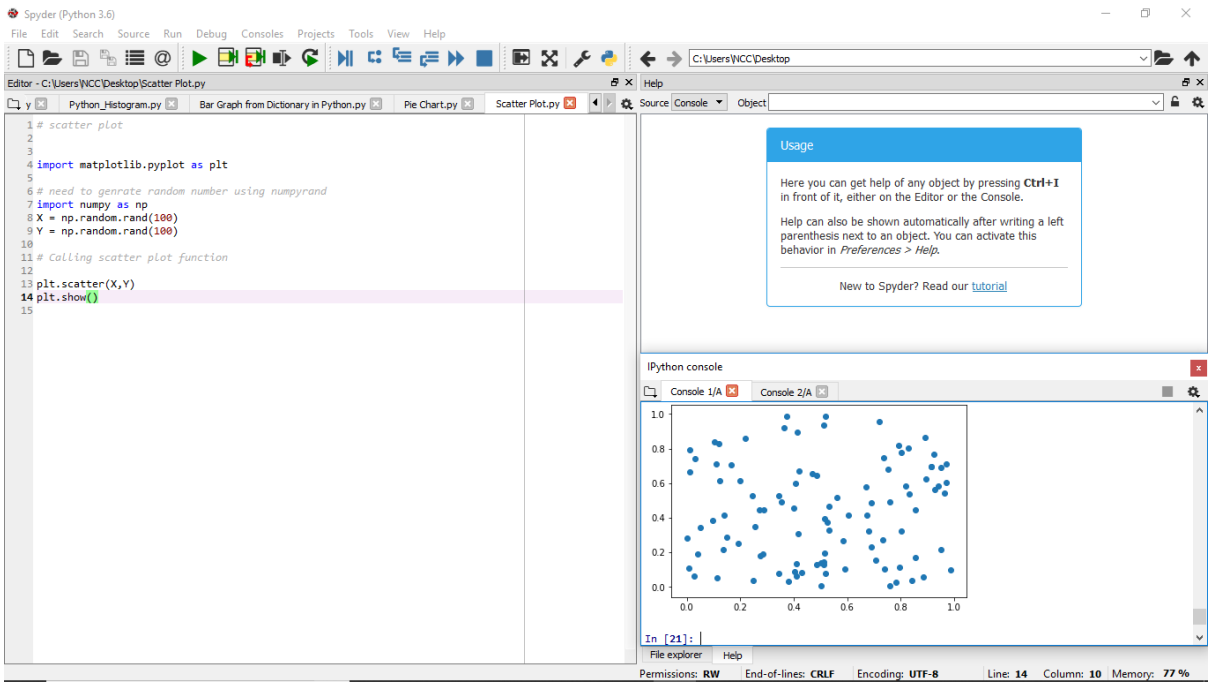


Figure 50 Scatter plot in Python