COMPUTER AIDED ENGINEERING II

(Course Code: MTE204, Units: 1)

for

Bachelor of Engineering

in

Mechatronics Engineering

Department of Mechatronics Engineering

Federal University Oye-Ekiti

Ekiti State

Lecture Note Prepared by: Engr. Dr. Oluwaseun. O. Martins

DISCLAIMER

This document does not claim any originality and cannot be used as a substitute for prescribed textbooks. The information presented here is merely a collection by the author for his teaching assignments. Various sources as mentioned in this document as well as freely available material from the internet were consulted for preparing this document. The ownership of the information lies with the respective authors or institutions.

MTE204

Introduction to Simulink

1.0. What Is Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamic systems (systems whose state varies with time). It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Simulink is a companion program to MATLAB and it contains two windows:

1. Library window: this contains several library and each library contains blocks
2. Workspace window: this is where the Simulink model is constructed

1.1. Tool for Interactive Simulation

Models can easily be constructed from scratch, or an existing model can be taken and addition made to it. Simulations are interactive, so one can change parameters in real time and immediately see the outcome.

1.2. Tool for Model-Based Design

Simulink turns the computer into a lab for modeling and analyzing systems. The workspace window in Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. Here one can draw the models just as you would with pencil and paper. Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors. One can also customize and create his own blocks. After drawing a model, one can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in the MATLAB Command Window. Using scopes and other display blocks, you can see the simulation results while the simulation is running. The simulation results can be put in the MATLAB workspace for post-processing and visualization. MATLAB and Simulink are integrated, so one can simulate, analyze, and revise your models in either environment at any point.

Simulink can be used to explore the behavior of a wide range of real-world dynamic systems, including electrical circuits, shock absorbers, braking systems, and many other electrical, mechanical, and thermodynamic systems.

2.0. Building a Model

The model below integrates a sine wave and displays the result along with the sine wave. The block diagram of the model looks like this.

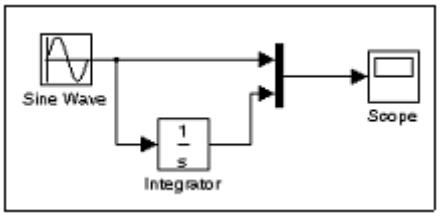
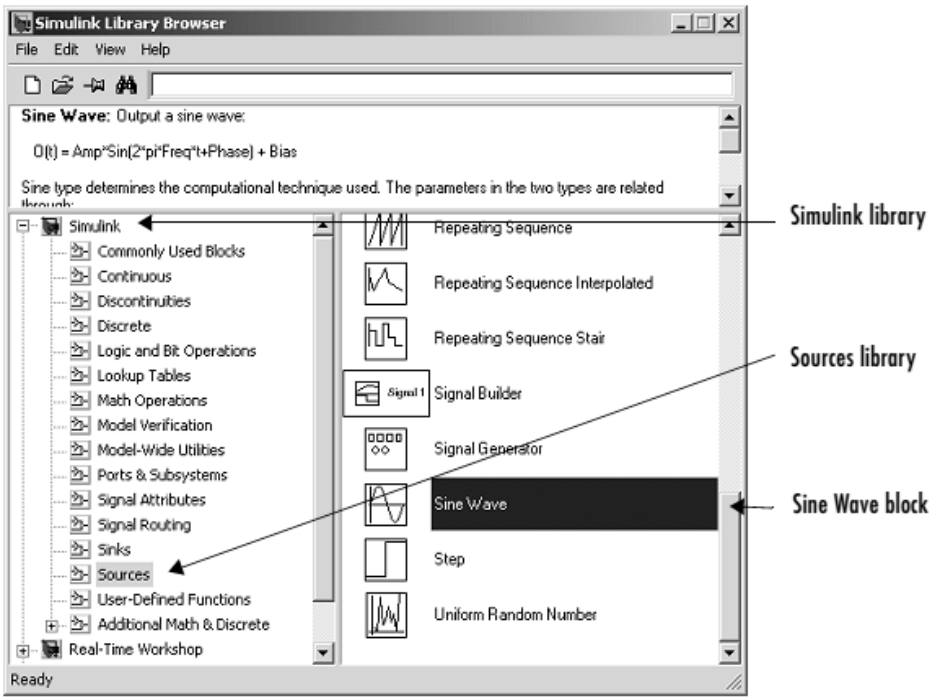


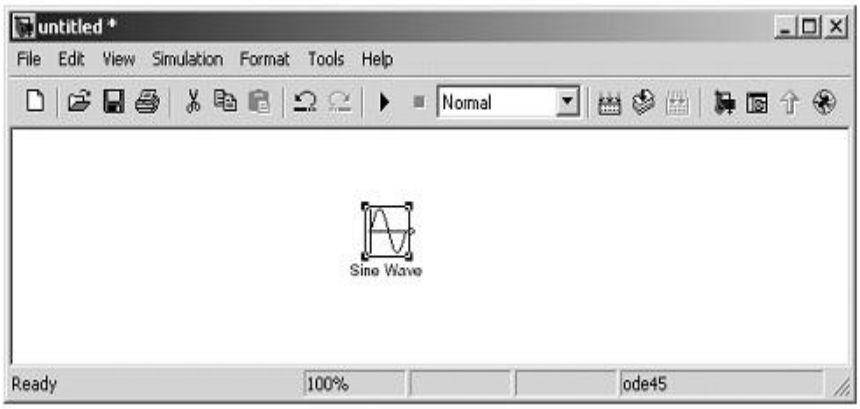
Fig. 1

2.1.Model creation steps

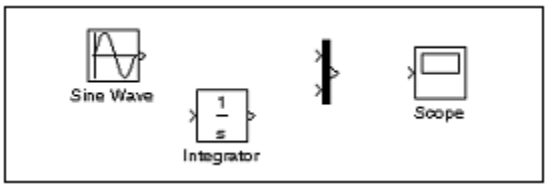
1. Enter Simulink in the MATLAB Command Window or click the Simulink library on the MATLAB toolbar, the Simulink Library Browser appears.
2. Click the New Model button on the Library Browser's toolbar, Simulink opens a new model window.
3. Create the model by copying blocks into the model from the following Simulink block libraries:
   - Sources library (the Sine Wave block)
   - Sinks library (the Scope block)
   - Continuous library (the Integrator block)
   - Signal Routing library (the Mux block)



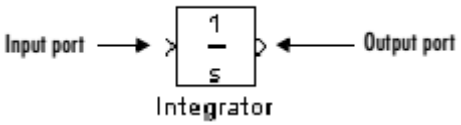
Now drag a copy of the Sine Wave block from the browser and drop it in the model window.



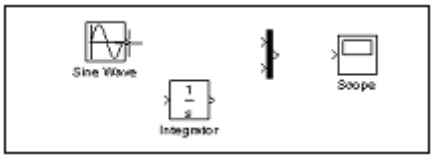
Repeat same for the other blocks.



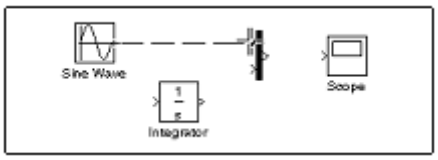
Examine the blocks above; observe the angle bracket on the right of the Sine Wave block and two on the left of the Mux block. *The > symbol pointing out of a block is an output port; if the symbol points to a block, it is an input port.* Signal travels out of an output port and into an input port of another block through a connecting line. When the blocks are connected, the port symbols disappear.



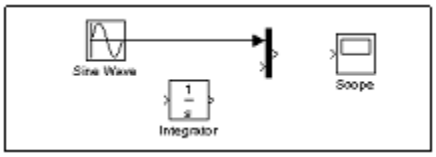
4. To connect the blocks, connect the Sine Wave block to the top input port of the Mux block. Position the mouse pointer over the output port right side of the Sine Wave block. Notice that the cursor shape changes to *crosshairs*.



Hold down the mouse button and move the cursor to the top input port of the Mux block.

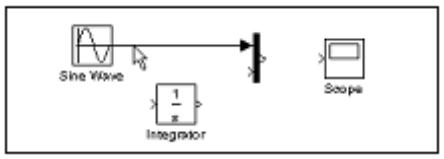


Now release the mouse button. The blocks are connected.

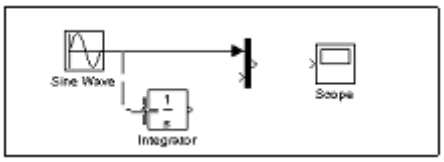


However, to create the branch line; such as that connects the Sine Wave output to the Integrator block in Fig.1 above;

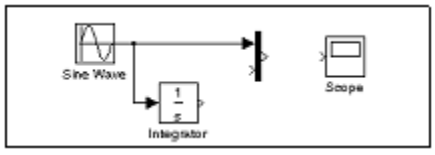
i. First, position the pointer on the line between the Sine Wave and the Mux block.



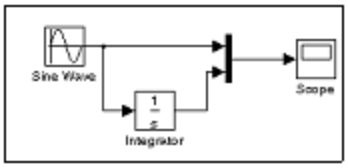
ii. Press and hold down the Ctrl key (or click the right mouse button). Press the mouse button, and then drag the pointer to the Integrator block's input port or over the Integrator blocks itself.



iii. Release the mouse button. Simulink draws a line between the starting point and the Integrator block's input port.



Finish making block connections. When you're done, your model should look something like this:



Note; the *branch line* carries the same signal that passes from the Sine Wave block to the Mux block.

2.2. Setting up to run the simulation

Now we set up Simulink to run the simulation for 10 seconds. First, open the Configuration Parameters dialog box by choosing Configuration Parameters from the Simulation menu. On the dialog box that appears, notice that the Stop time is set to 10.0 (its default value).

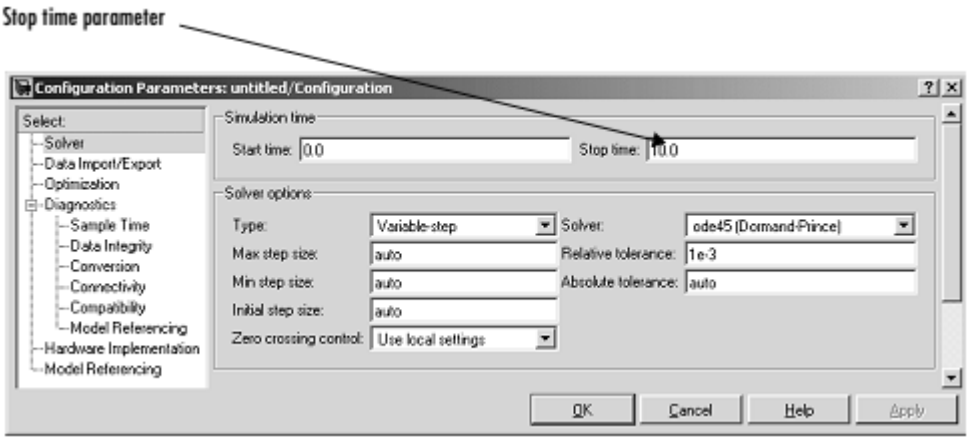


Fig. 2

Close the Configuration Parameters dialog box by clicking the OK button. Simulink applies the parameters and closes the dialog box. Now double-click the Scope block to open its display window. Finally, choose Start from the Simulation menu and watch the simulation output on the Scope.

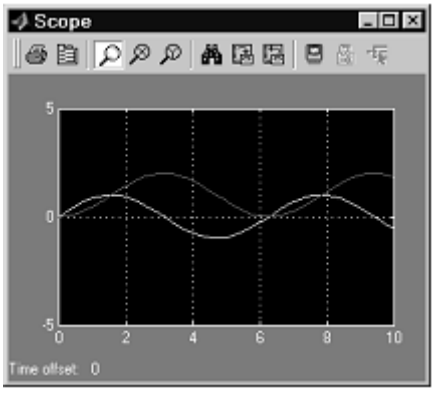


Fig. 3

The simulation stops when it reaches the stop time specified in the Configuration Parameters dialog box or when you choose Stop from the Simulation menu or click the Stop button on the model window's toolbar (Windows only).

To save this model, choose Save from the File menu and enter a filename and location. That file contains the description of the model.
To terminate Simulink and MATLAB, choose Exit MATLAB

2.2.1. Solving Differential Equations

Simulink can be used to solve evaluate various types of dynamic systems. Consider the first order differential equation:

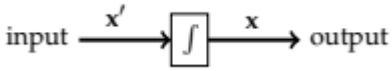
$$\frac{dx}{dt} = 2\sin 3t - 4x.$$

With initial condition $x(0) = 0$.

The system can be viewed as $x' = 2\sin 3t - 4x$, then fed into an integrator we have the output $x(t)$.

$$x(t) = \int x'(t)\,dt.$$

This can generally be depicted by



The schematics of the model of solving the system problem is presented in Figure 1

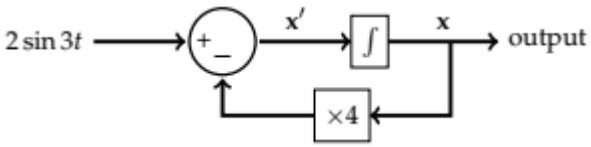


Figure 1 Model schematics 1

In Simulink the blocks required to model this system is as follows:

- Integrator block from the Continuous group;
- Sum block from the Math Operations group,
- Gain block from the Math Operations group,
- Sine Wave block from the Math Operations group; and,
- Scope block from the Sink group.

Therefore, the Simulink model is presented in Figure 2 after inputting the parameters of the sine wave, sum and integrator blocks.

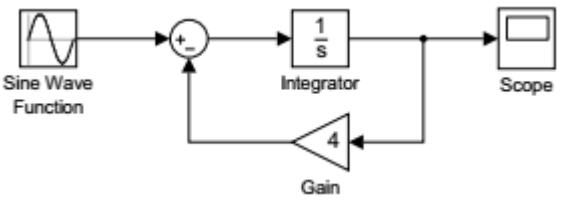


Figure 2 Simulink model 1

The simulation of the model gives the output of the system as presented in Figure 3. Obtained by double-clicking the Scope to see the solution. Figure 3 shows the Scope plot after using the auto-scale ( ) feature to rescale the scope view.

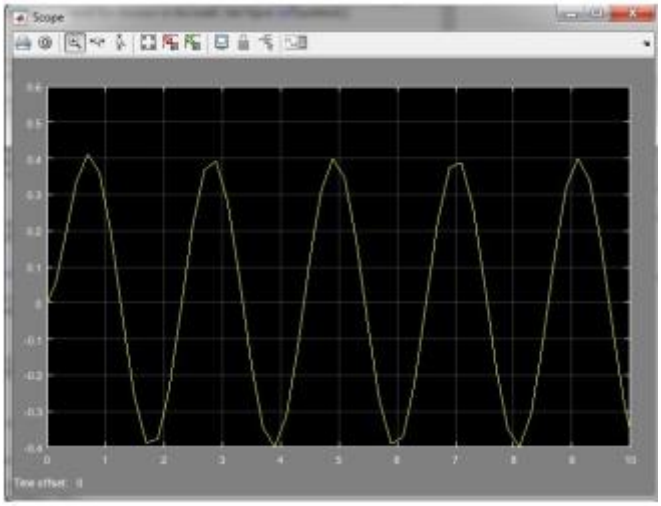


Figure 3 Oscilloscope output of model 1

However, we can make further changes to the system by checking the Configuration Parameters under the Simulation menu item. See Figure 4. In particular, changing the Refine Factor to 10 units can lead to smoother solutions.

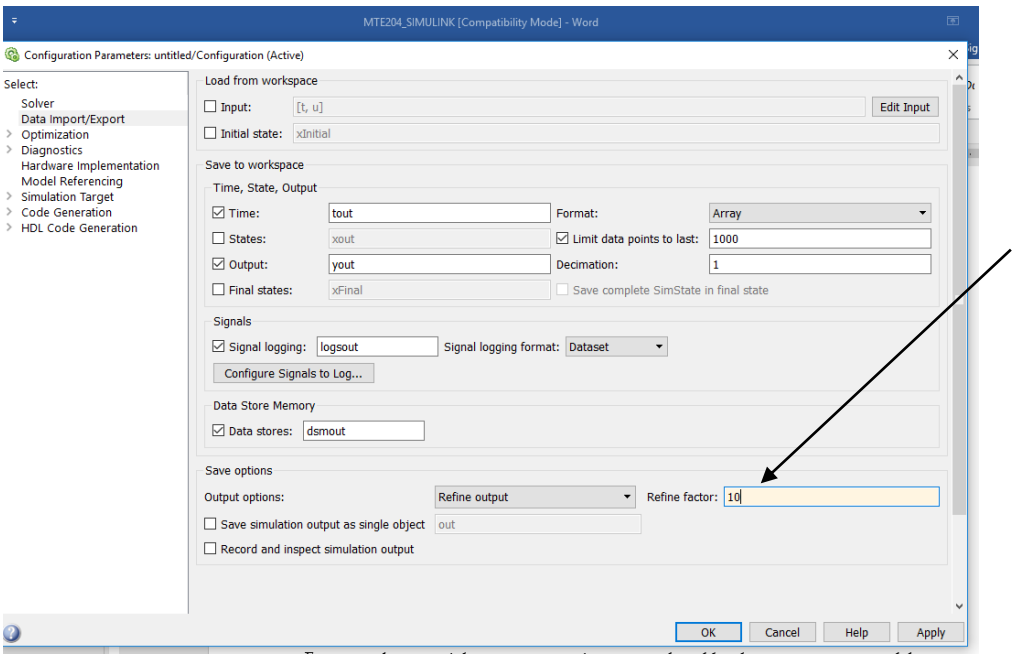


*Figure 4 System configuration parameter - Data Import/Export Parameters. Changing the **Refine Factor** to 10 units for smoother solution*

Hence, we obtain the scope output in Figure 5.

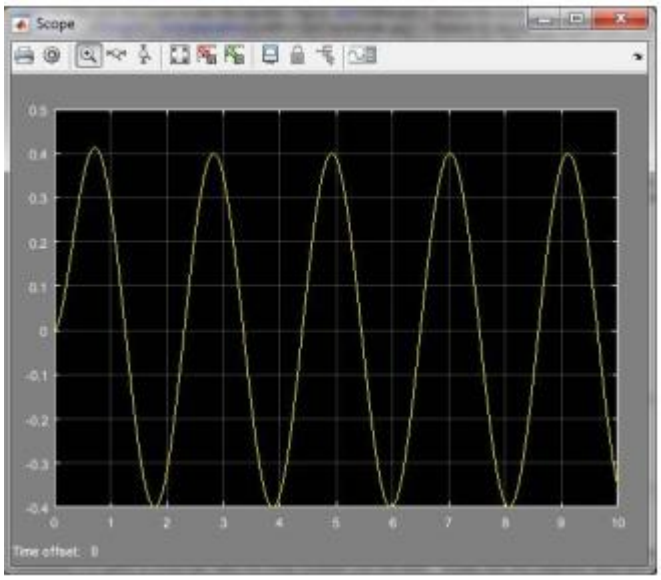


Figure 5 Scope plot of the solution, with **Refine Factor=** 10

Until now we have inputed the initial condition for the integrator internally. However, there are instances whereby we will like to do same externally. Double-click the Integrator block and change the initial condition source from internal to external Figure 6. This adds another input to the block Figure 7. Drag a Constant block from the Sources group into the model, connect it to the new input, and change the constant value to the desired initial value. This results in the simulation shown in Figure 8.

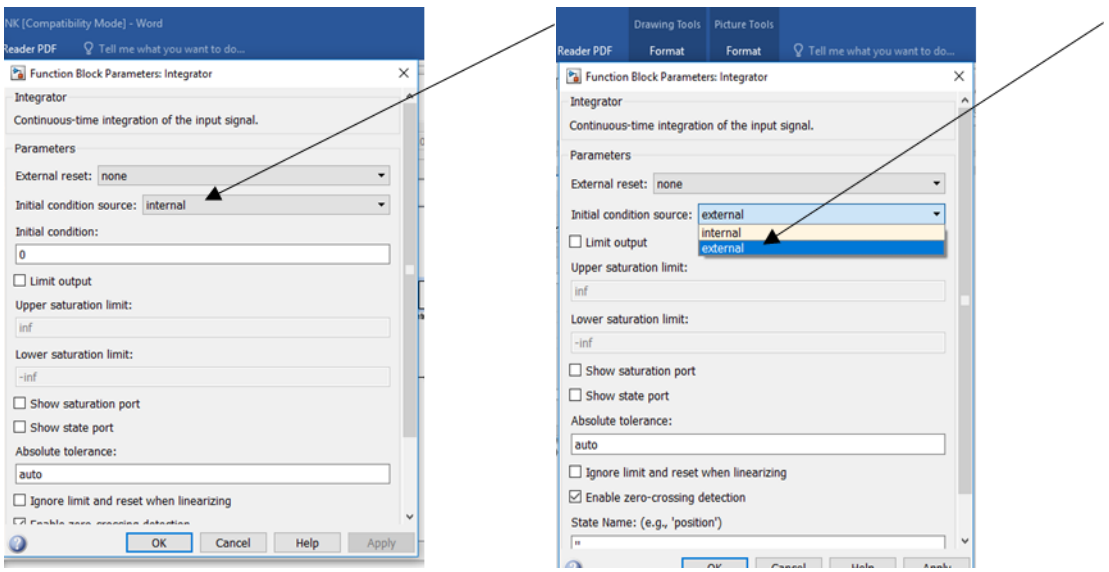


Figure 6 integrator block initial condition source

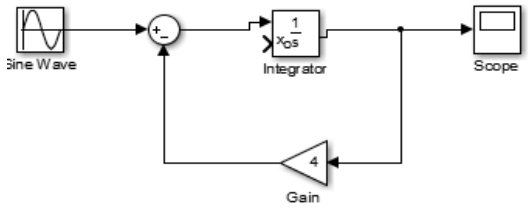


Figure 7 Additional input to integrator block

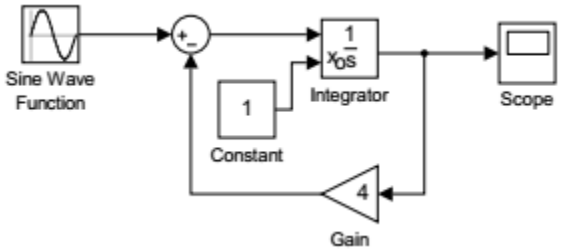


Figure 8 Constant block addition as external initial condition source for the integrator

2.2.2. Exponential Growth and Decay

The simplest differential equations are those governing growth and decay. As an example, we will discuss population models. Let P(t) be the population at time t. We seek an expression for the rate of change of the population, $\frac{dp}{dt}$. Assuming that there is no migration of population, the only way the population can change is by adding or subtracting individuals in the population. The equation would take the form

$$\frac{dp}{dt} = Rate\ In - Rate\ Out.$$

$$Rate\ In\ =\ bP\ and\ Rate\ Out\ =\ mP.$$

This gives the total rate of change of population as

$$\frac{dp}{dt} = bp - mp$$

$$where, k = b - m$$

$$Then,\quad \frac{dp}{dt} = kp$$

The equation above can be modelled easily in Simulink. Using Integrator, Constant, Gain, and a Scope block. Given the initial value, $P(0)\ =\ 8$ and $k\ =\ -0.8$

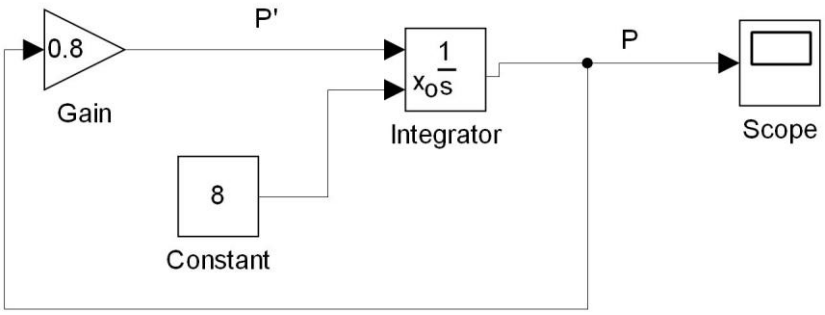


Figure 9 Simulink model for exponential growth and decay

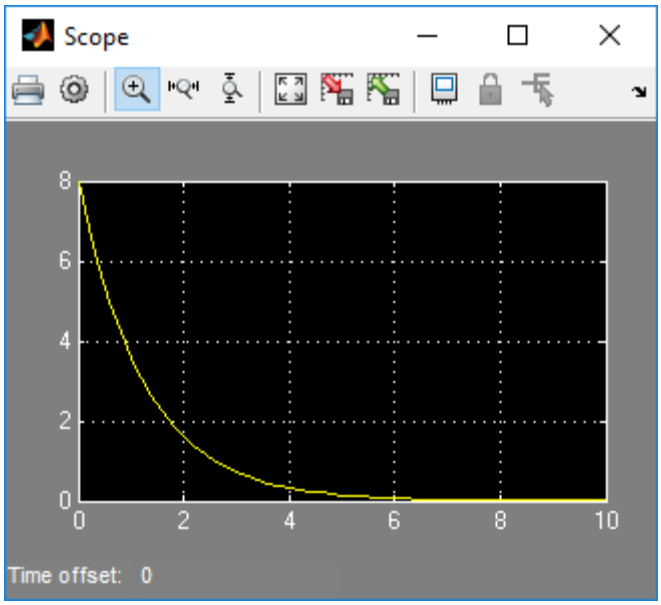


Figure 10 Solution for the exponential decay with $\boldsymbol{P}(0) = 8$ *and* $\boldsymbol{k} = -0.8$*. The simulation time was set at 10*

2.2.3. Newton's Law of Cooling

The law of cooling is attributed to Isaac Newton (1642-1727) who was probably the first to state results on how bodies cool. For instance A cup of hot tea, kept in a room will cool off and reach room temperature after a period of time. The main is idea is that a body at temperature $T(t)$ is initially at temperature $T(0) = T0$. It is placed in an environment at an ambient temperature of Ta. The goal is to find the temperature at a later time, $T(t)$.

Let assume that the rate of change of the temperature of the body is proportional to the temperature difference between the body and its surroundings. Thus, we have

$$\frac{dT}{dt} \propto \mathrm{T} - \mathrm{Ta}$$

The proportionality is removed by introducing a cooling constant,

$$\frac{dT}{dt} = -\mathrm{K}\,(\mathrm{T} - \mathrm{Ta})$$

Where, $K > 0$

Example 2.1. A cup of tea at 90^oC cools to 85^oC in ten minutes. If the room temperature is 22^oC, what is its temperature after 30 minutes?

Using the general solution with $T_0 = 90^o$C,

$$T(t) = 22 + (90 - 22)e^{-k} = 22 + 68e^{-kt},$$

we then find k using the given information, $T(10) = 85^o$C. We have

$$\begin{aligned} 85 &= T(10) \\ &= 22 + 68e^{-10k} \\ 63 &= 68e^{-10k} \\ e^{-10k} &= \frac{63}{68} \approx 0.926 \\ -10k &= \ln 0.926 \\ k &= -\frac{\ln 0.926}{10} \\ &\approx 0.00764\text{min}^{-1}. \end{aligned}$$

This gives the solution for this model as

$$T(t) = 22 + 68e^{-0.00764t}.$$

Now we can answer the question. What is $T(30)$?

$$T(30) = 22 + 68e^{-0.00764(30)} = 76^o\text{C}.$$

Example 2.2

Using Simulink; A cup of tea at 60°C is placed in a room with a temperature of 20°C. Find the temperature of a cup of tea at time 60s and 80s. given k = 0.1 s^{-1}

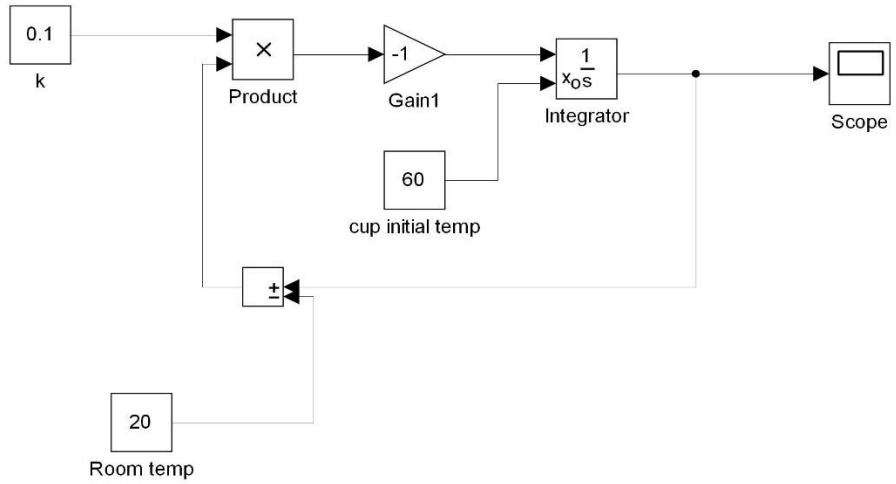


Figure 11 Simulink model Newton's law of cooling

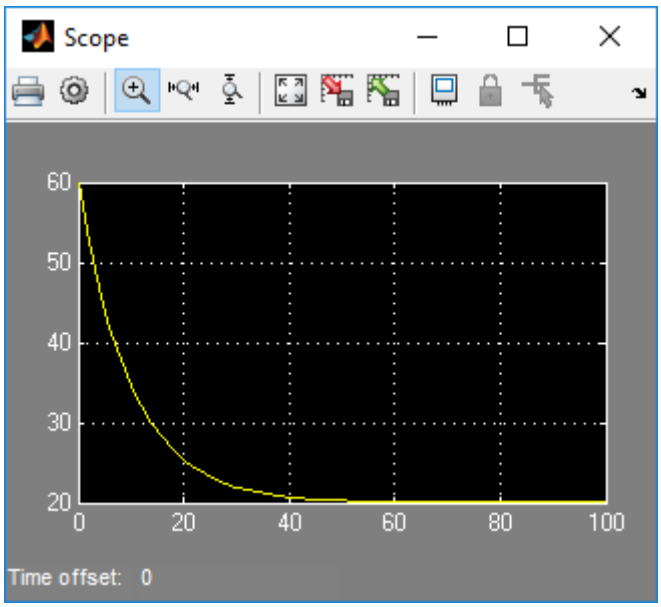


Figure 12 Solution for T0 = -k(T - Ta), T(0) = T0. Here we set k = 0.1 s-1, Ta = 20°C, and T0 = 60°C.

From the Figure 12, the cup temperature at 60s and 80s is 20°C

Note that example 2.1 shows the mathematical method where the equation to solve is $T(t) = T_a + (T_0 - T_a)e^{-kt}$

Where, T_a is the ambient temperature and T_0 is the body initial temperature.

2.2.4. Free Fall with Drag

Consider an object falling to the ground with air resistance? Free fall is the vertical motion of an object solely under the force of gravity. It has been experimentally determined that an object near the surface of the Earth falls at a constant acceleration

in the absence of other forces, such as air resistance. This constant acceleration is denoted by −g, where g is called the acceleration due to gravity. The negative sign is an indication that we have chosen a coordinate system in which "up" is positive.

We are interested in determining the position, $y(t)$, of a falling body as a function of time. The differential equation governing free fall is have

$$\ddot{y}(t) = -g$$

However, the differential equation we need to solve is

$$\dot{v} = Kv^2 - g$$

Where, g is acceleration due to gravity, K is drag and v is velocity.

The Simulink model is presented in Figure 13 with K set as 0.00159

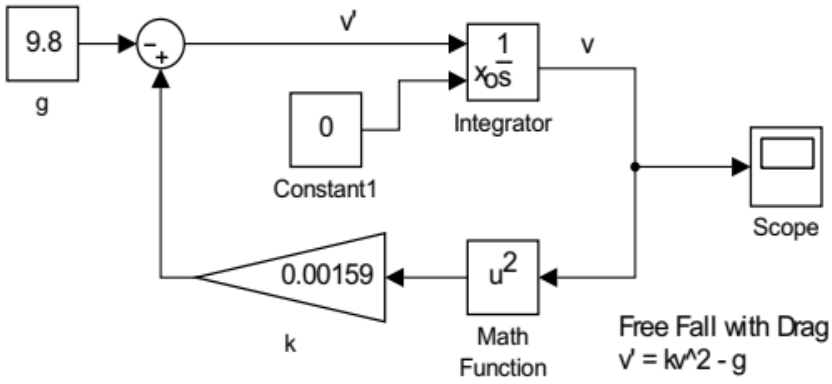


Figure 13 Simulation model for free fall

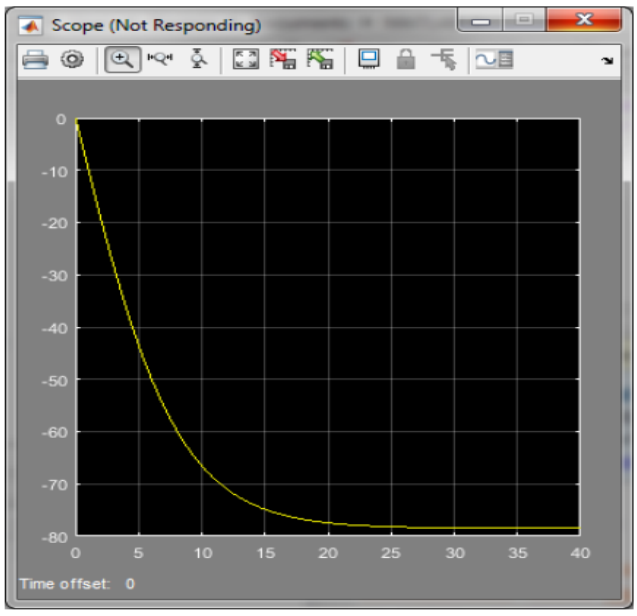


Figure 14 Solution for free fall with drag with **k** = 0.00159 starting from rest.

2.3. Working with Simulink Output

Often we might want to access the solutions in MATLAB. Using the model in Fig. 8 add the *To Workspace block from* Sinks Library. Double-click and rename the *simout* variable name as *y* and change the output type to array Figure 15. Run the simulation. This will put *tout* and *y* data into the MATLAB *workspace*. In MATLAB you can plot the data using plot (tout,y). You can add labels with xlabel('t'), ylabel('y'), title('y vs t').

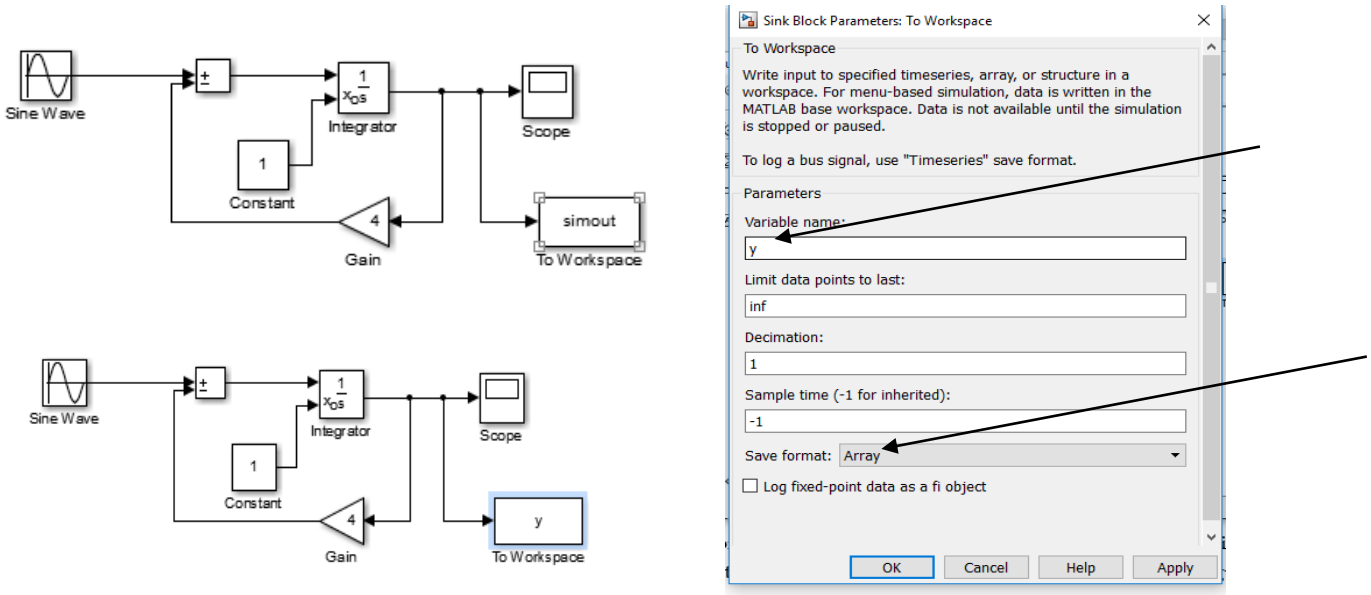


Figure 15 Outputting to MATLAB

2.4. Printing Simulink Scope Images

For example, one might want to copy images produced by the scope or your model into an MS Word document. Select the Scope Figure window in Fig. 5, then hit ALT+PrintScrn to copy the Figure to a clipboard and paste the Figure into your application.

2.5. Printing Models

Once you have made a model, you might want to include it in a report; this can be done by typing the following in the MATLAB command window:

1. To print the open model to an encapsulated postscript file:

```
print('-sModelName','-desp,'model.esp')
```

2. For jpg files, you can use

```
print('-sModelName','-djpeg','model.jpeg')
```

3. For figure model, you can use

```
print -djpeg -FigureName -r300 myfigure
```

The picture file of the model will appear in MATLAB current folder.

3.0. How Simulink Works

Simulating a dynamic system is a two-step process with Simulink. First, a user creates a block diagram, using the Simulink model editor that graphically depicts time-dependent mathematical relationships among the system's inputs, states, and outputs. The user then commands Simulink to simulate the system represented by the model from a specified start time to a specified stop time.

3.1. Modeling Dynamic Systems

A Simulink block diagram model is a graphical representation of a mathematical model of a dynamic system.

3.2. Block Diagram Semantics

A classic block diagram model of a dynamic system graphically consists of blocks and lines (signals). The relationships between each elementary dynamic system in a block diagram are illustrated by the use of signals connecting the blocks. Collectively the blocks and lines in a block diagram describe an overall dynamic system.

There are two classes of blocks:

1. Nonvirtual blocks represent elementary systems. Blocks that plays specific role in the definition of the system of equations described by the block diagram model.
2. Virtual block is provided for graphical organizational convenience and plays no role in the definition of the system of equations described by the block diagram model. Examples of virtual blocks are the Bus Creator and Bus Selector which are used to reduce block diagram clutter by managing groups of signals as a "bundle." You can use virtual blocks to improve the readability of your models.

Because we use the term block diagrams in other fields, the term "*time-based block diagram*" is used to distinguish block diagrams that describe dynamic systems in Simulink from that of other forms of block diagrams.

Summary meaning of time-based block diagrams:

i. Simulink block diagrams define time-based relationships between signals and state variables. The solution of a block diagram is obtained by evaluating these relationships over time, where time starts at a user

specified "start time" and ends at a user specified "stop time." Each evaluation of these relationships is referred to as a time step.

ii. Signals represent quantities that change over time and are defined for all points in time between the block diagram's start and stop time.

iii. The relationships between signals and state variables are defined by a set of equations represented by blocks.

3.3. Time

Time is an inherit component of block diagrams in that the results of a block diagram simulation change with time. Simply put, a block diagram represents the instantaneous behavior of a dynamic system.

3.4. States

Typically, the current values of some system, and hence model, outputs are functions of the previous values of temporal variables. Two types of states can occur in a Simulink model: discrete and continuous states. A continuous state changes continuously. Examples of continuous states are the position and speed of a car. A discrete state is an approximation of a continuous state where the state is updated (recomputed) using finite (periodic or aperiodic) intervals. An example of a discrete state would be the position of a car shown on a digital odometer where it is updated every second as opposed to continuously.

Assignment

Using Simulink, model, simulate and analysis the system below:

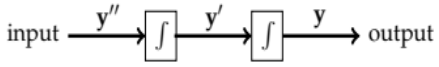
1. $\frac{d}{dt}y(t) + 5y(t) = x(t)$ where $x(t) = u(t)$, initial condition $y(0) = -2$
2. $\frac{dy}{dt} = \frac{6}{t}\, y$, where $y(1) = (1)$,
3. $\frac{dy}{dt} = \frac{6}{t}\, y + t^2$, where $y(1) = (1)$
4. Solution to the logistic equation, $y' = ry\,(1 - y)$, with $r = 1$ and $y(0) = 0.1$.
5. $\frac{dy}{dt} = y^2\,(1 + t^2)$ where $y(1) = (1)$
6. $\frac{ds}{dt} + 2s = st^2$ where $s(1) = (1)$
7. $x' + 2x = te^{2t}$ where $s(1) = (1)$
8. The temperature inside your house is 70°F and it is 30°F outside. At 1:00 A.M. the furnace breaks down. At 3:00 A.M. the temperature in the house has dropped to 50°F. Assuming the outside temperature is constant and that Newton's Law of Cooling applies, determine when the temperature inside your house reaches 40°F.

9. A body is discovered during a murder investigation at 8:00 P.M. and the temperature of the body is 70^{o}F. Two hours later the body temperature has dropped to 60^{o}F in a room that is at 50^{o}F. Assuming that Newton's Law of Cooling applies and the body temperature of the person was 98.6^{o}F at the time of death, determine when the murder occurred.

4.0. Second Order Differential Equations

These are equations involving the second derivative,$y^{"}(x)$. Let's assume that we can write the equation as $y^{"}(x) = F\ (x, y(x), y'(x))$.

We would like to solve this equation using Simulink. This is accomplished using two integrators in order to output $y'(x)$ and $y(x)$.



Example

Model the initial value problem $y^{"}(x) + 5y' + 6y = 0, y'(0) = 1, y(0) = 0$ in Simulink

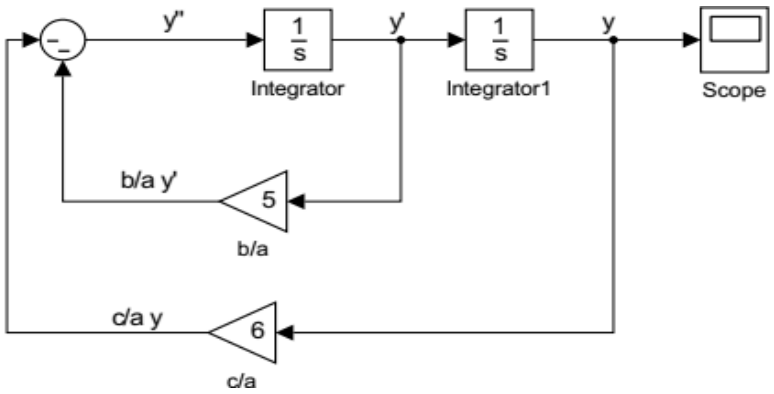


Figure 16 Model for the second order constant coefficient ODE $y^{"}(x) + 5y' + 6y = 0$

4.1. Harmonic Oscillation

A typical application of second order, constant coefficient differential equations is the simple harmonic oscillator as shown in Figure 17. Consider a mass, m, attached to a spring with spring constant, k. According to Hooke's law, a stretched spring will react with a force $F\ = -kx$, where x is the displacement of the spring from its un-stretched equilibrium. The mass experiences a net for and will accelerate according to Newton's Second Law of Motion, F = ma. Setting these forces equal and noting that $a = \ddot{x}$, we have

$$m\ddot{x} + kx = 0$$

Hence, the differential equation we intend to solve with Simulink is $\ddot{x} = -\frac{1}{m}\ (kx)$.

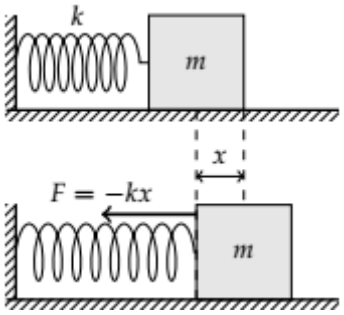


*Figure 17 A simple harmonic oscillator consists of a mass, **m**, attached to a spring with spring constant, **k***

Example

A Simulink model for simple harmonic motion where k = 5 and m = 2. We also specify the initial conditions $x(0) = 1$ and $\dot{x}(0) = 0$ is

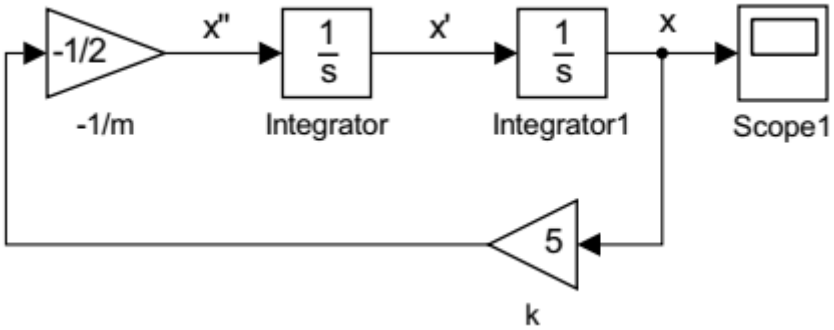
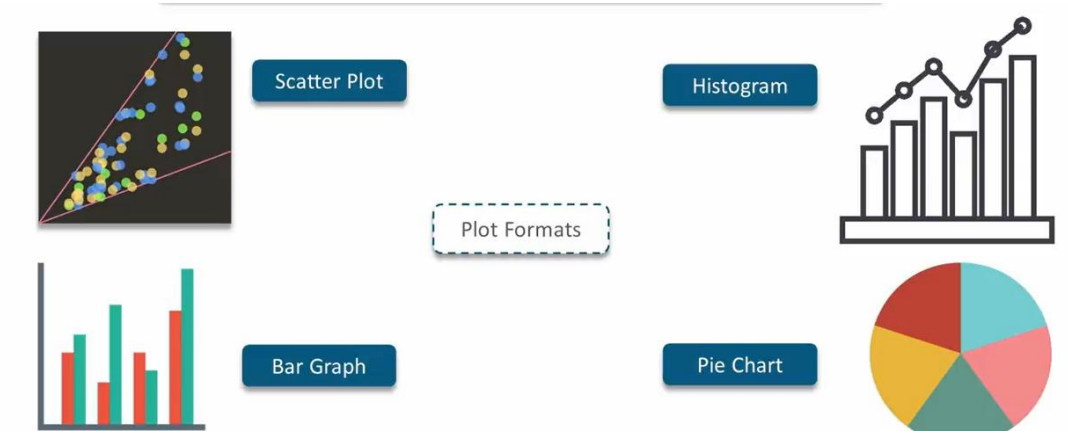


Figure 18 A model for damped simple harmonic motion, $m\ddot{x} + kx = 0$

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.

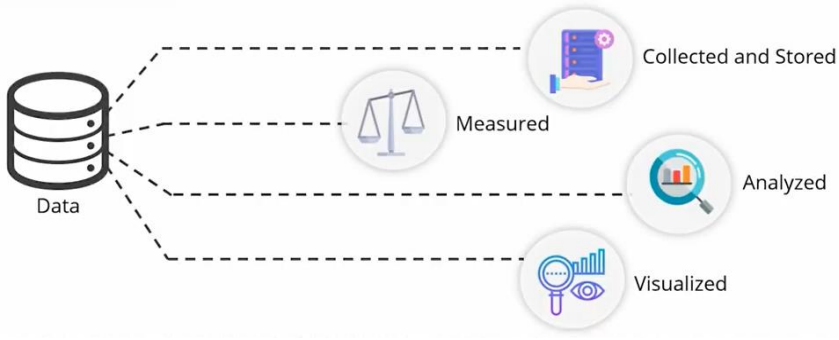
Python application includes:

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

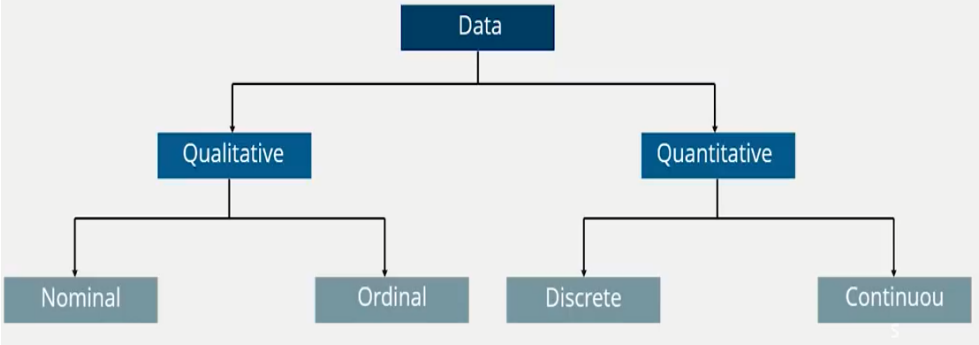
After the installation of Python, the next step is to start working on python. We will discuss some important attributes then move to data analysis with python.

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.

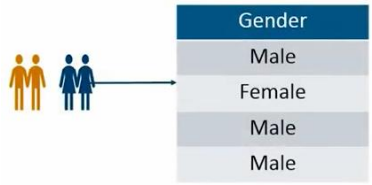


Data is subcategorized as depicted below:



Qualitative 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.



Gender
Male
Female
Male
Male

- Ordinal data are data with an ordered series.

Customer ID	Rating
001	Good
002	Average
003	Average
004	Bad

Quantitative 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 students 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

```
help> keywords

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

and                 elif                if                  print
as                  else                import              raise
assert              except              in                  return
break               exec                is                  try
class               finally             lambda              while
continue            for                 not                 with
def                 from                or                  yield
del                 global              pass
```

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

'''

Presented below is a demo of the python interface

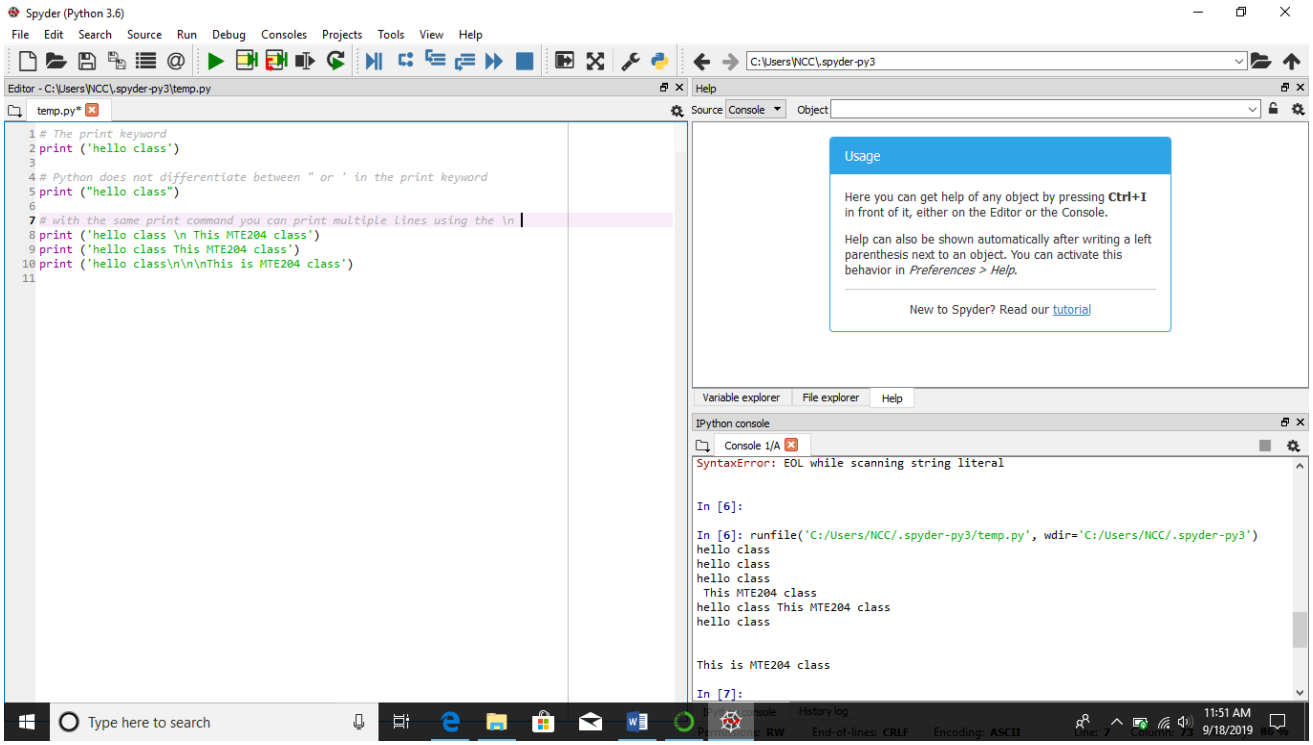


Figure 19 Character printing

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.

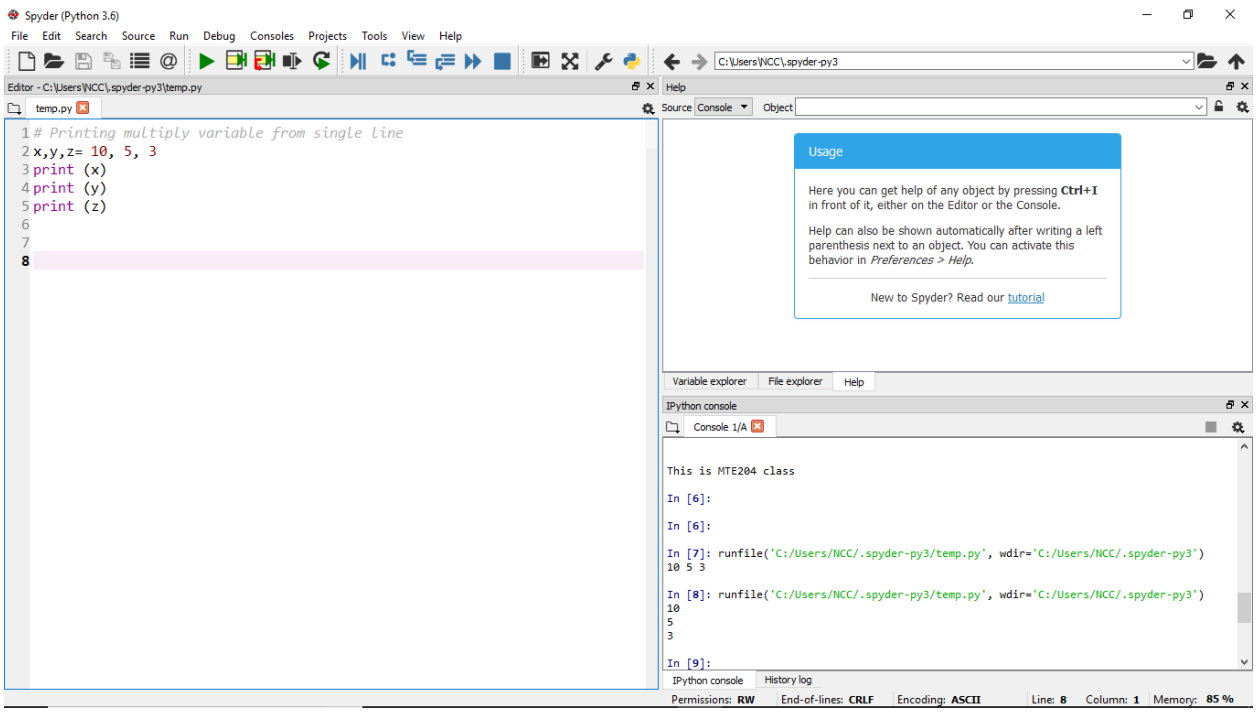


Figure 20 Printing multiply variable form a single line input

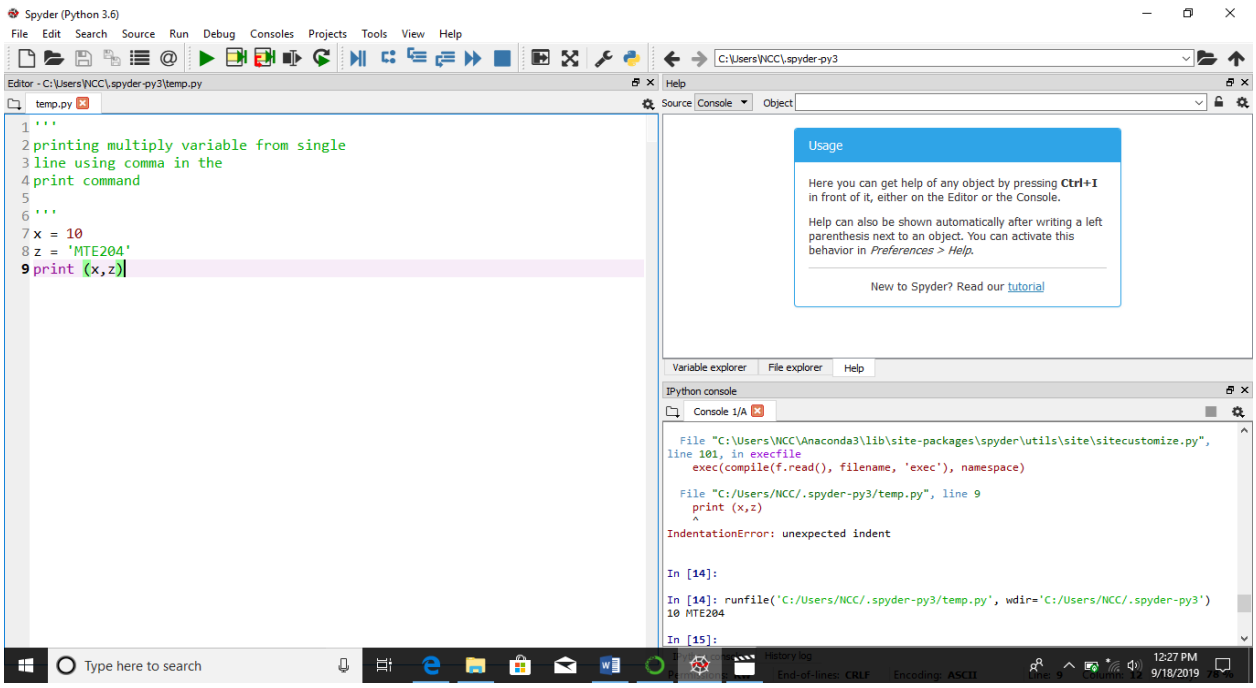


Figure 21 Printing multiply variable with comma in print command brace.

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 starts 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

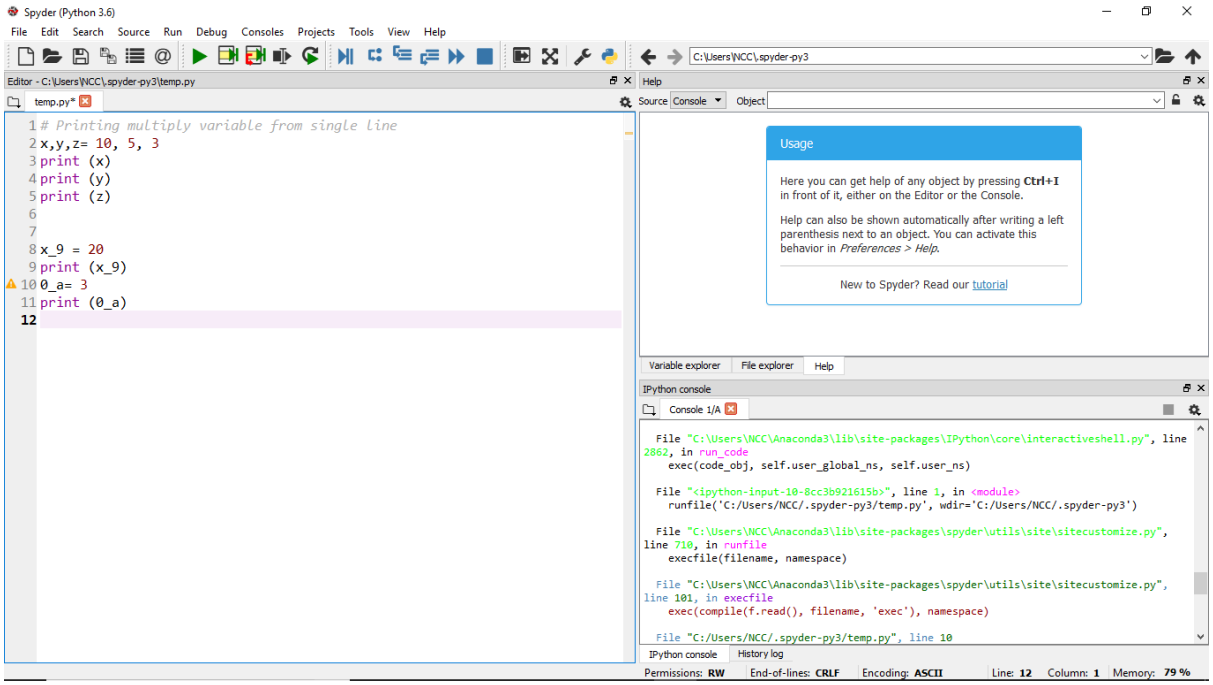


Figure 22 identifier

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:

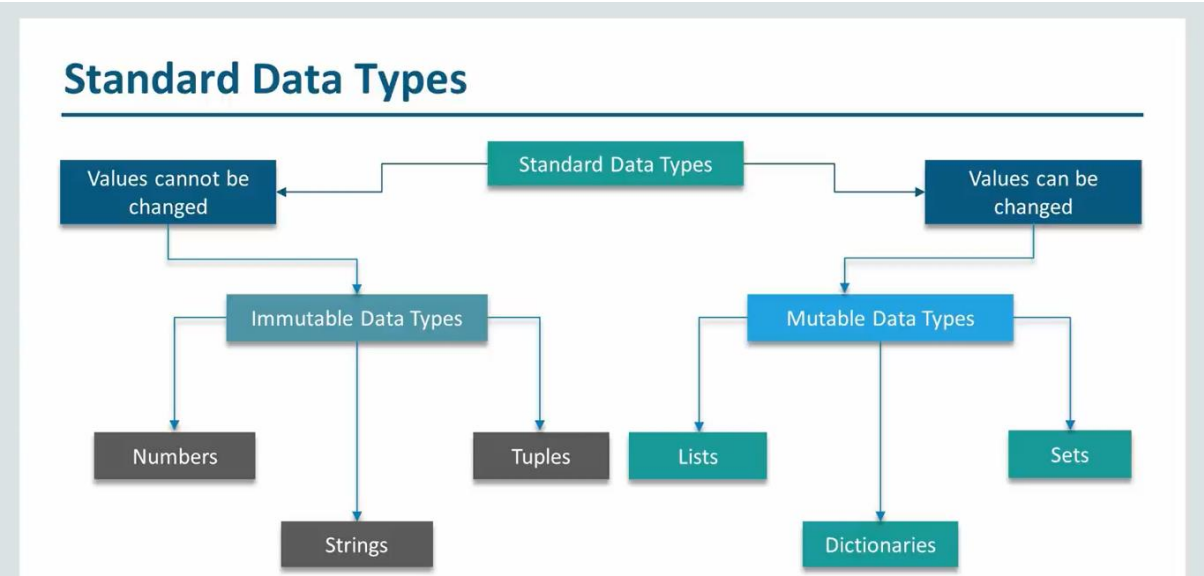
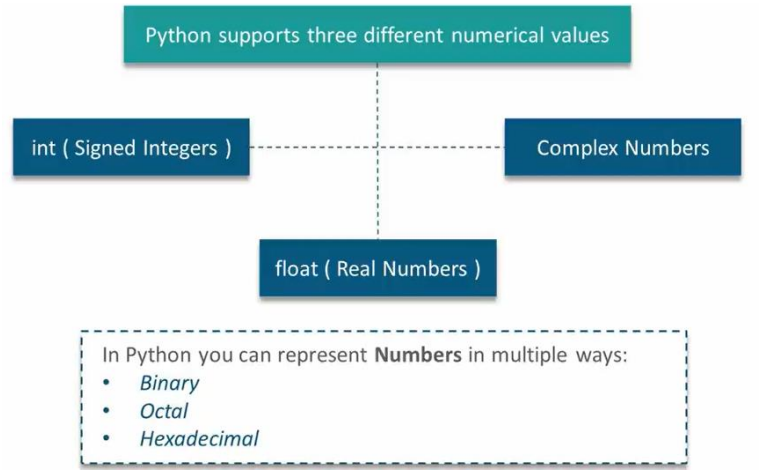
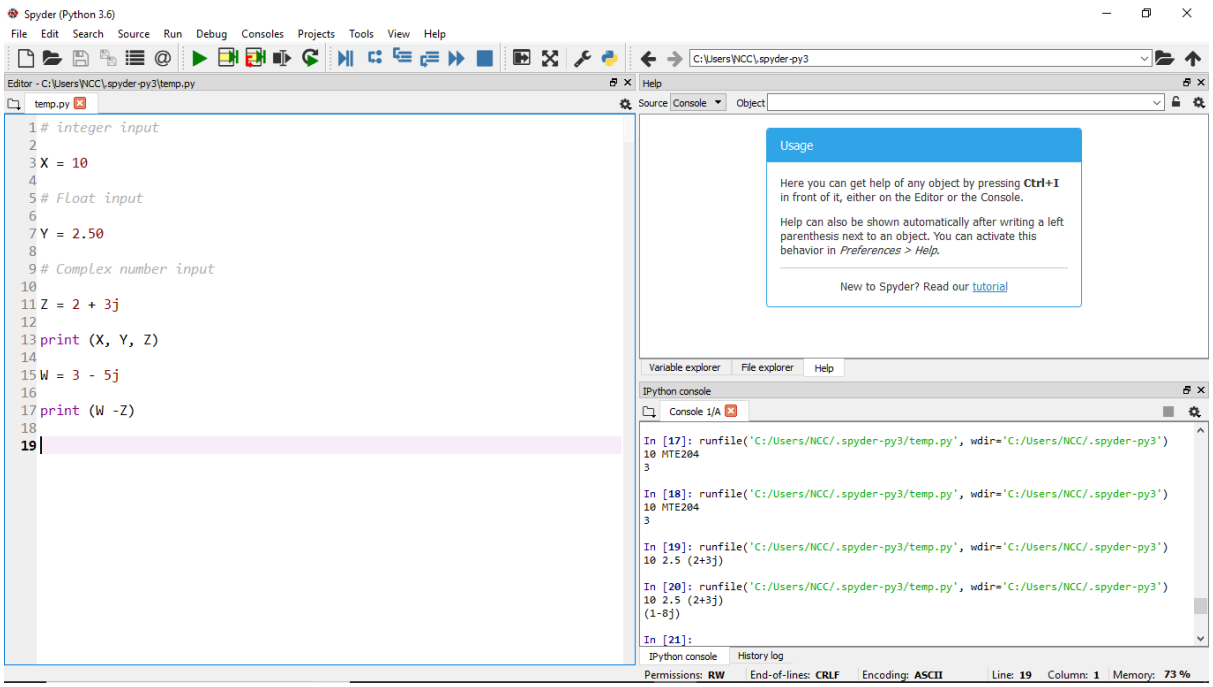


Figure 23 Standard Data Type

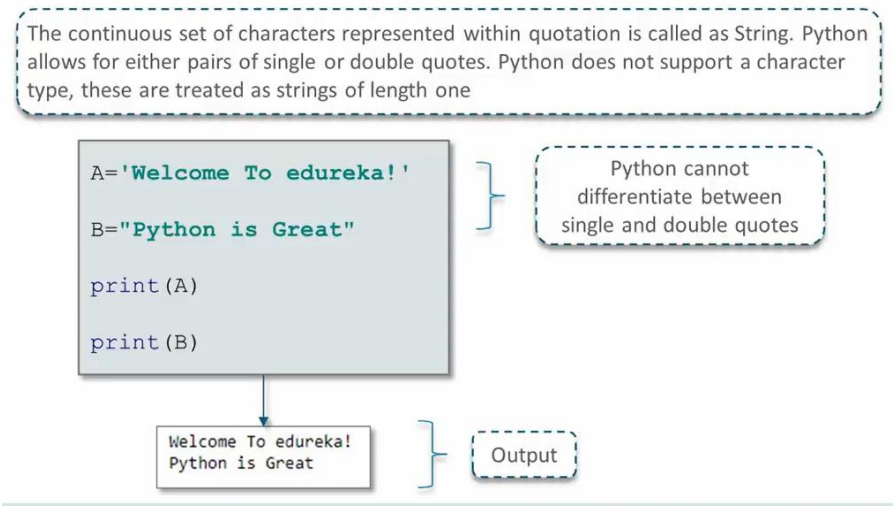
Immutable Data Type –

1. Numeric Data Type

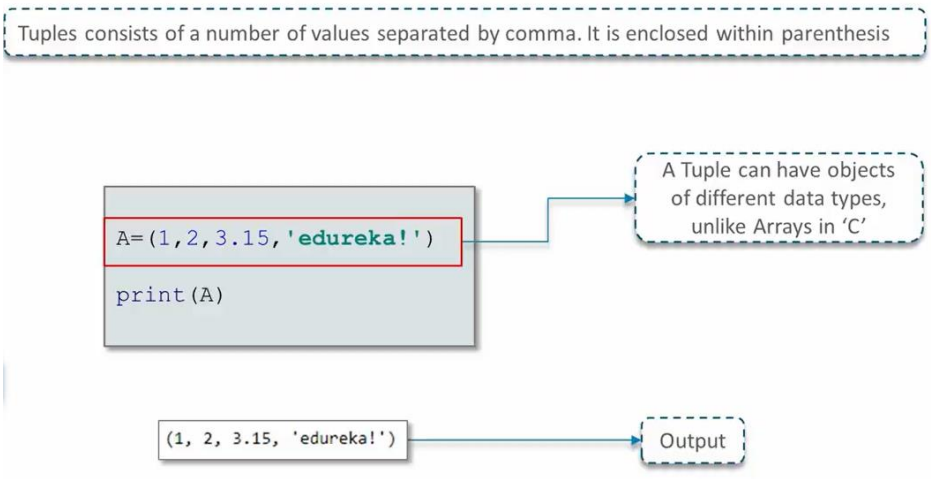




2. Sting


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


Tuples consists of a number of values separated by comma. It is enclosed within parenthesis
A=(1,2,3.15,'edureka!')
print(A)
A Tuple can have objects of different data types, unlike Arrays in 'C'
(1, 2, 3.15, 'edureka!')
Output

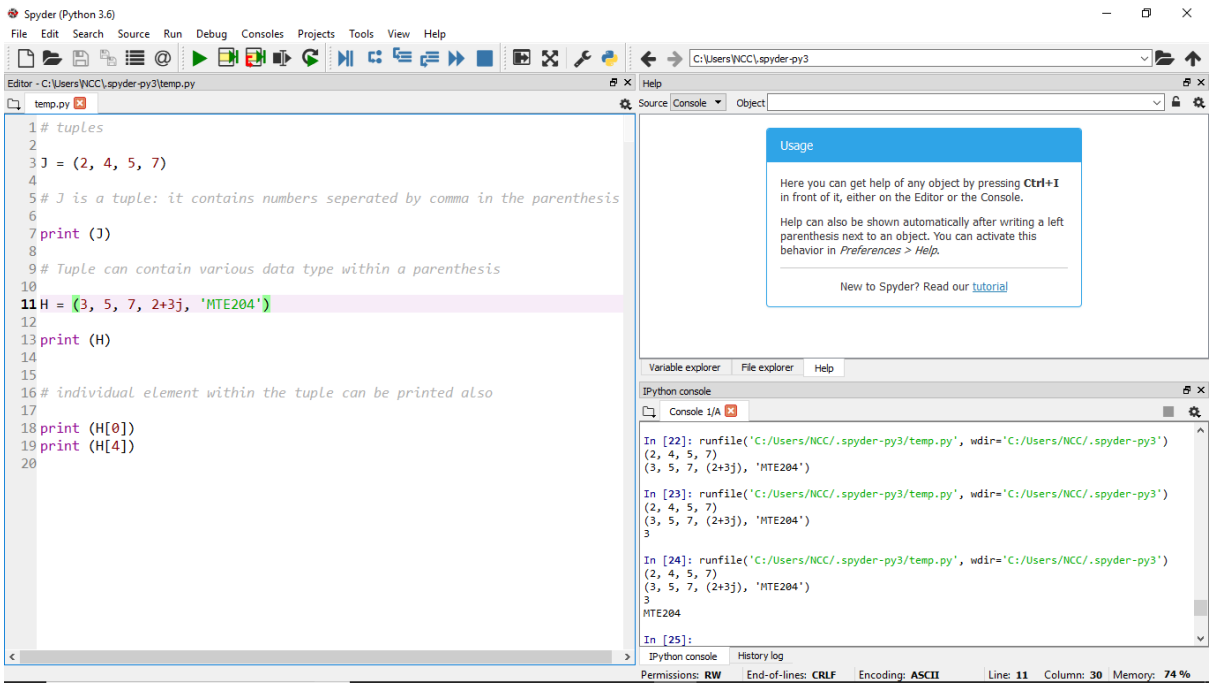


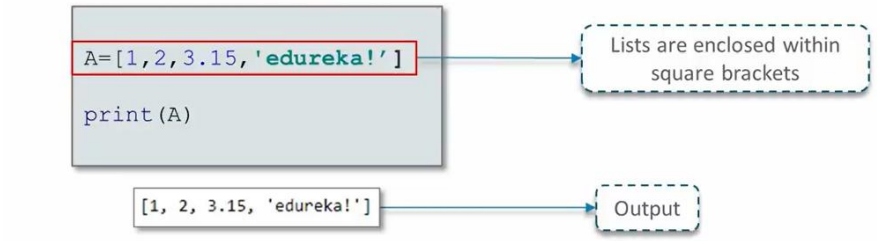
Figure 24 Tuples

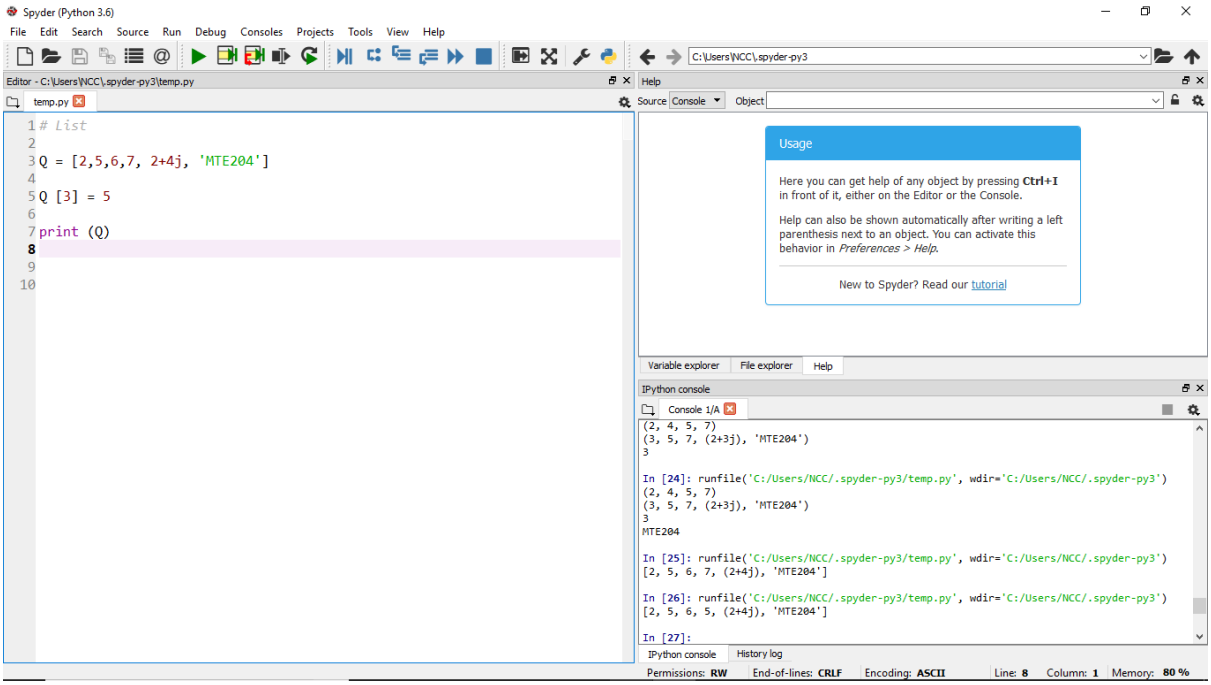
Mutable Data Type

1. List

List is an ordered set of elements enclosed within square brackets. The main differences between Lists and Tuples are:

- Lists are enclosed in brackets[] and Tuples are enclosed within parenthesis()
- Lists are Mutable and Tuples are Immutable
- Tuples are faster than Lists





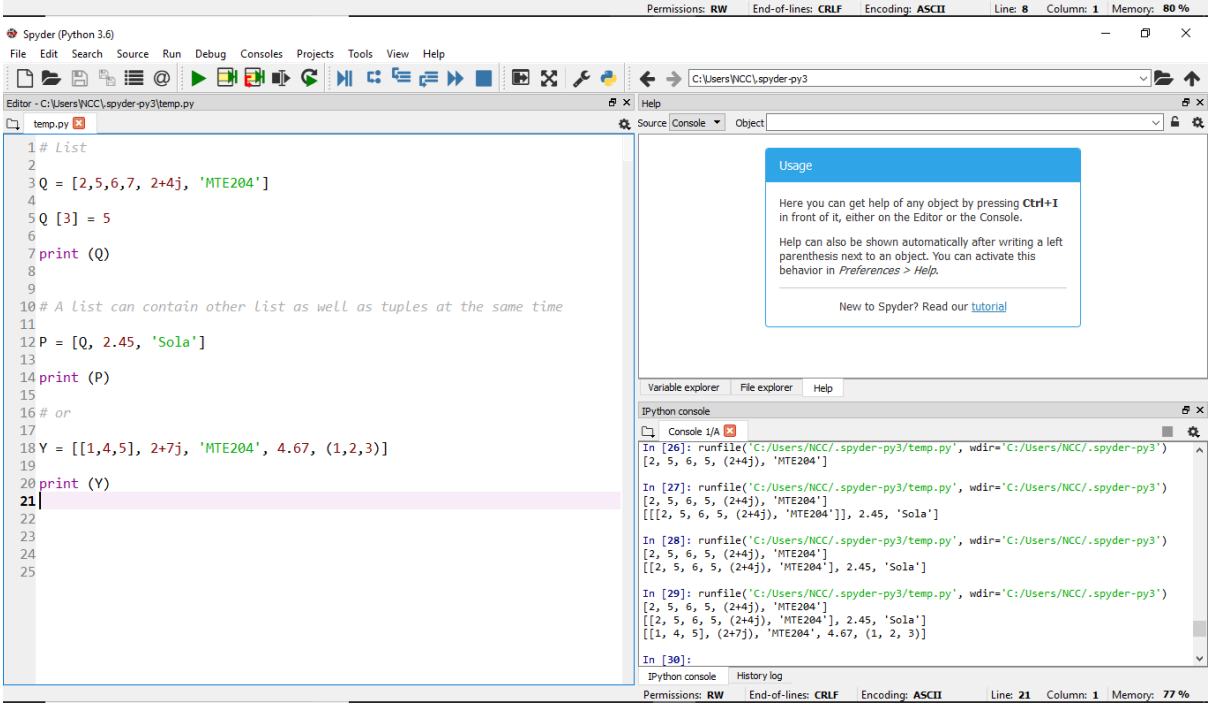


Figure 25 List data type

2. Dictionaries

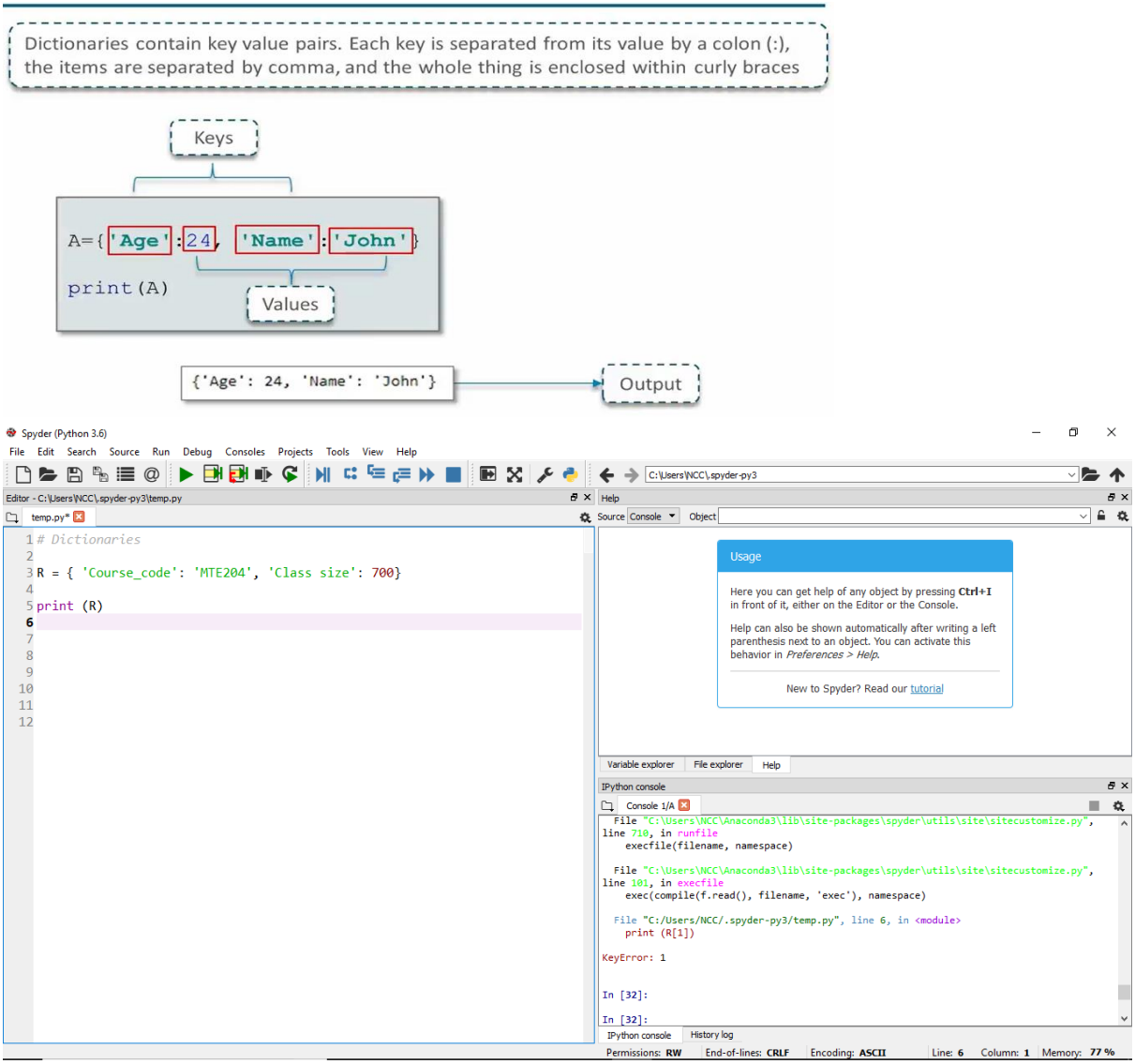


Figure 26 Dictionaries

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

3. Sets

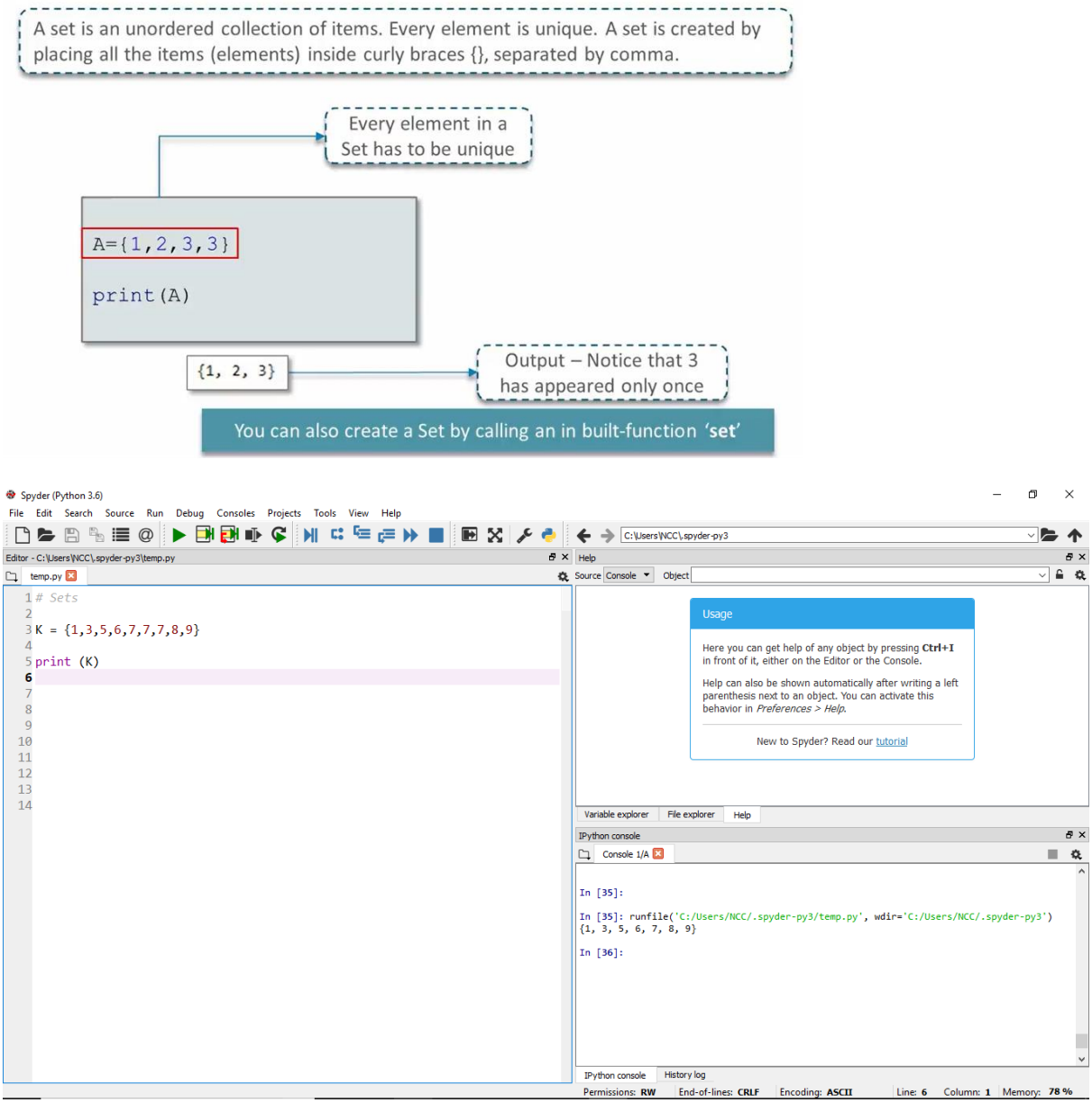
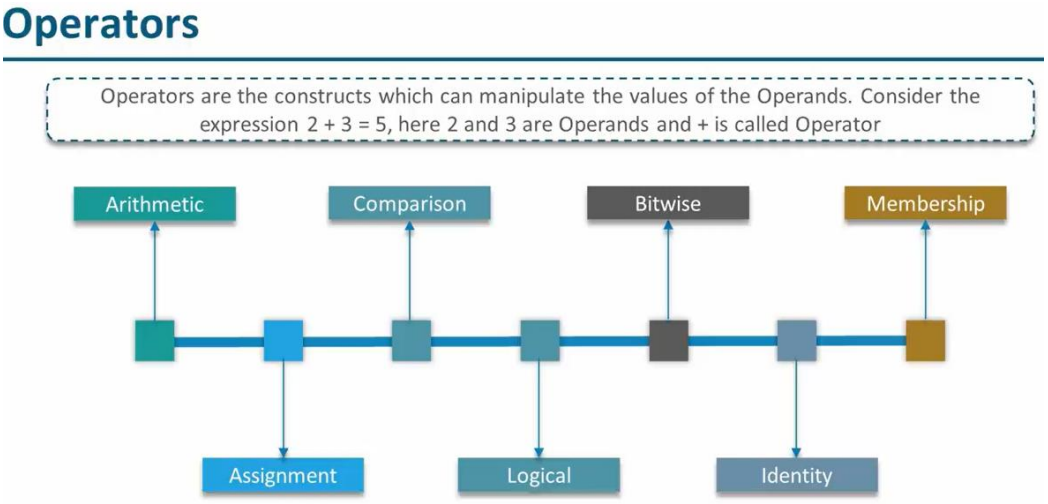


Figure 27 Sets

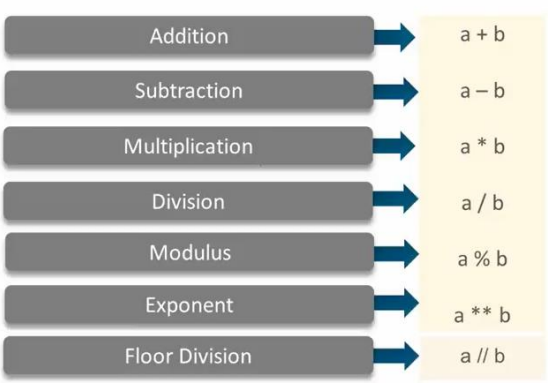
1.3.4. Operators

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

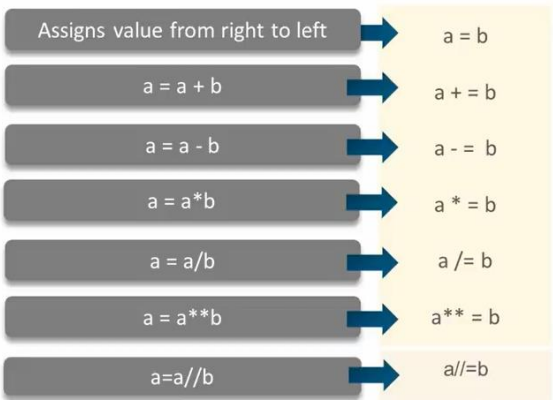
Operators



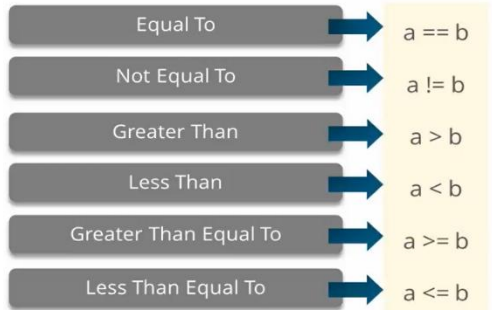
1. Arithmetic Operators

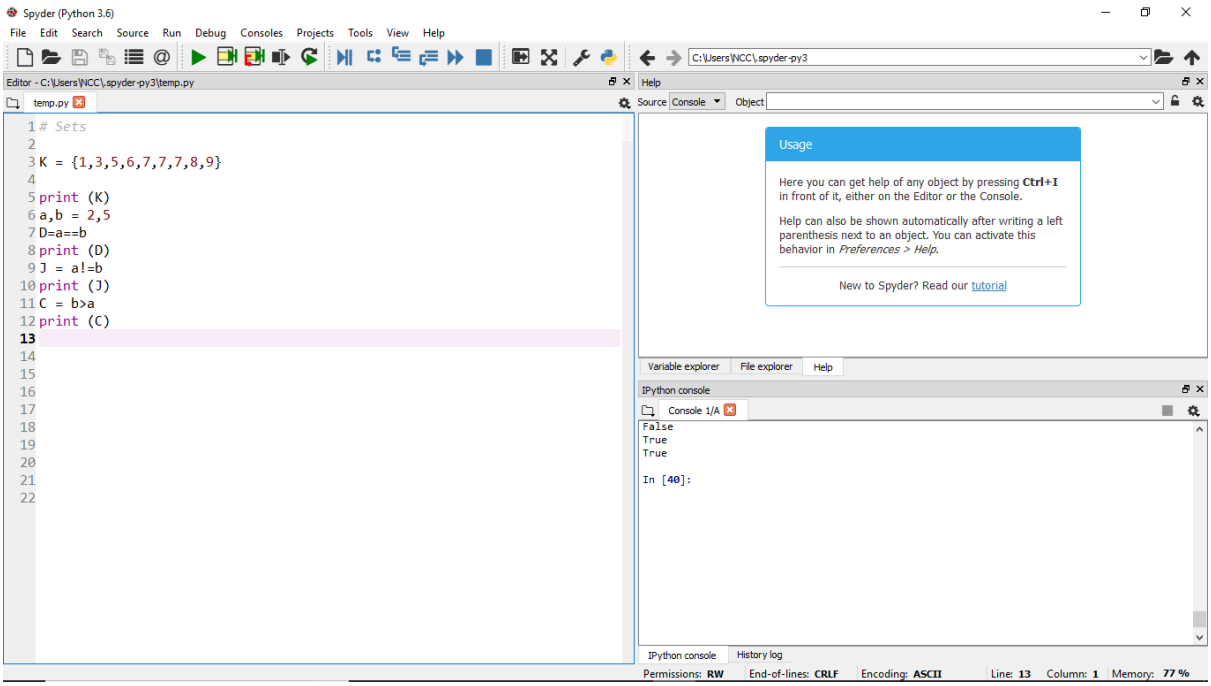


2. Assignment Operator

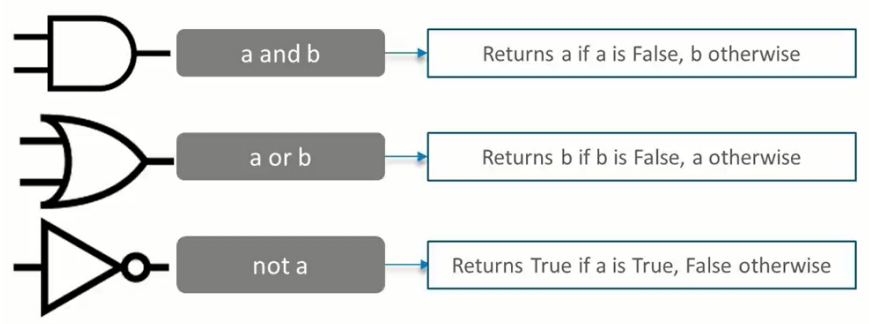


3. Comparison Operator

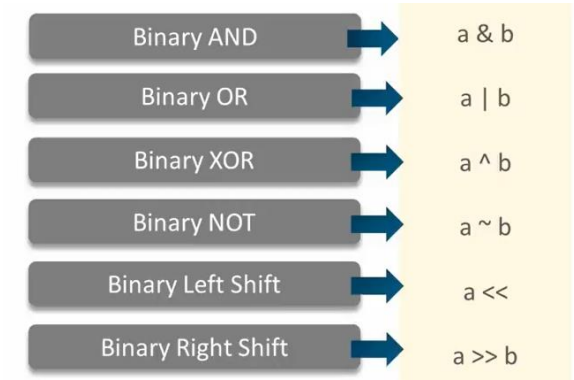




4. Logic Operators



5. Bitwise Operator



6. Identity Operator

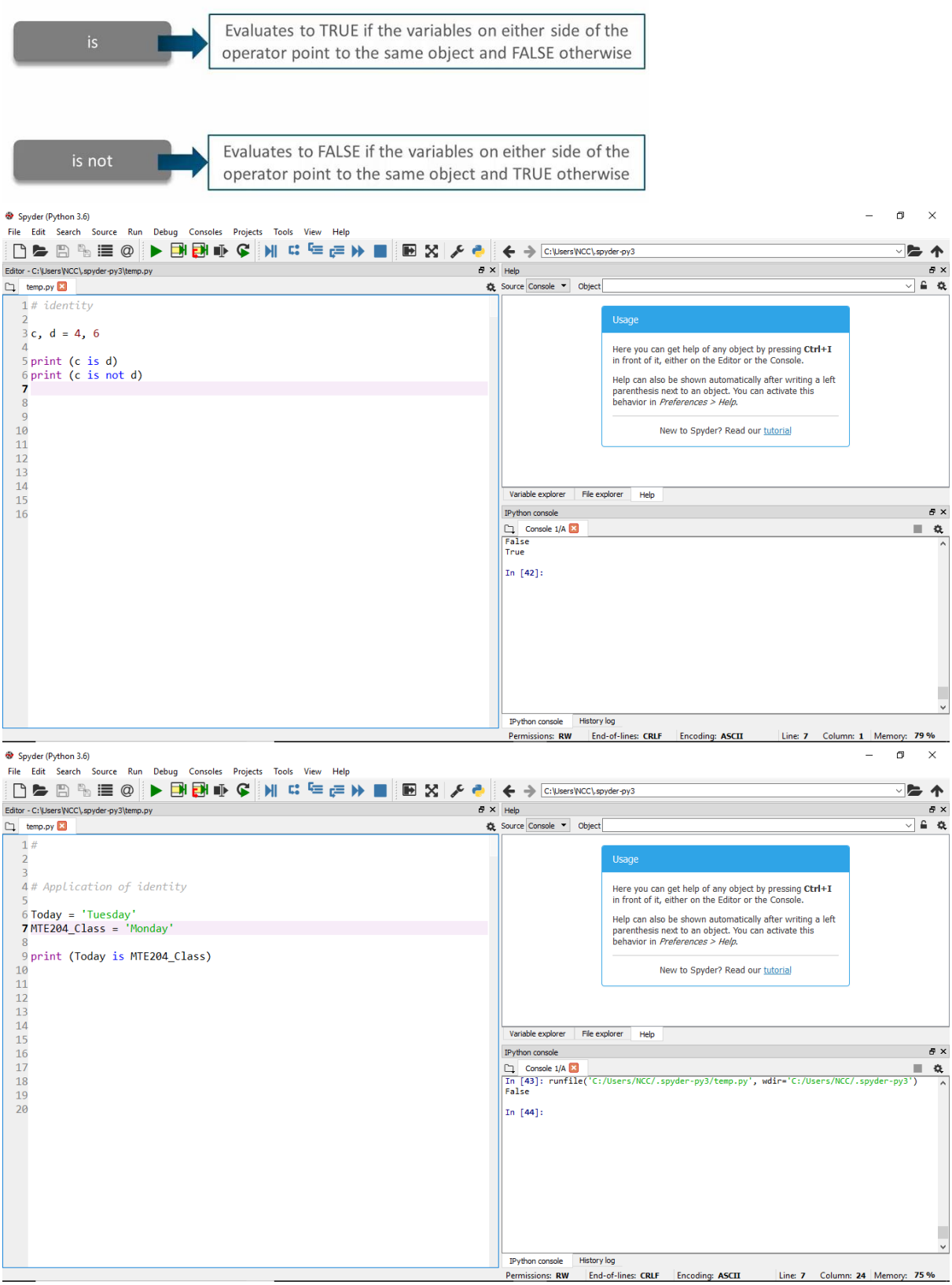
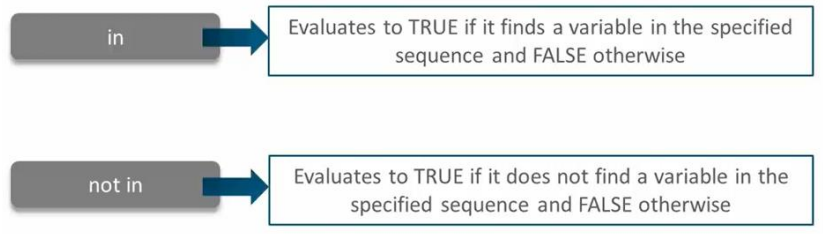


Figure 28 Identity operator

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



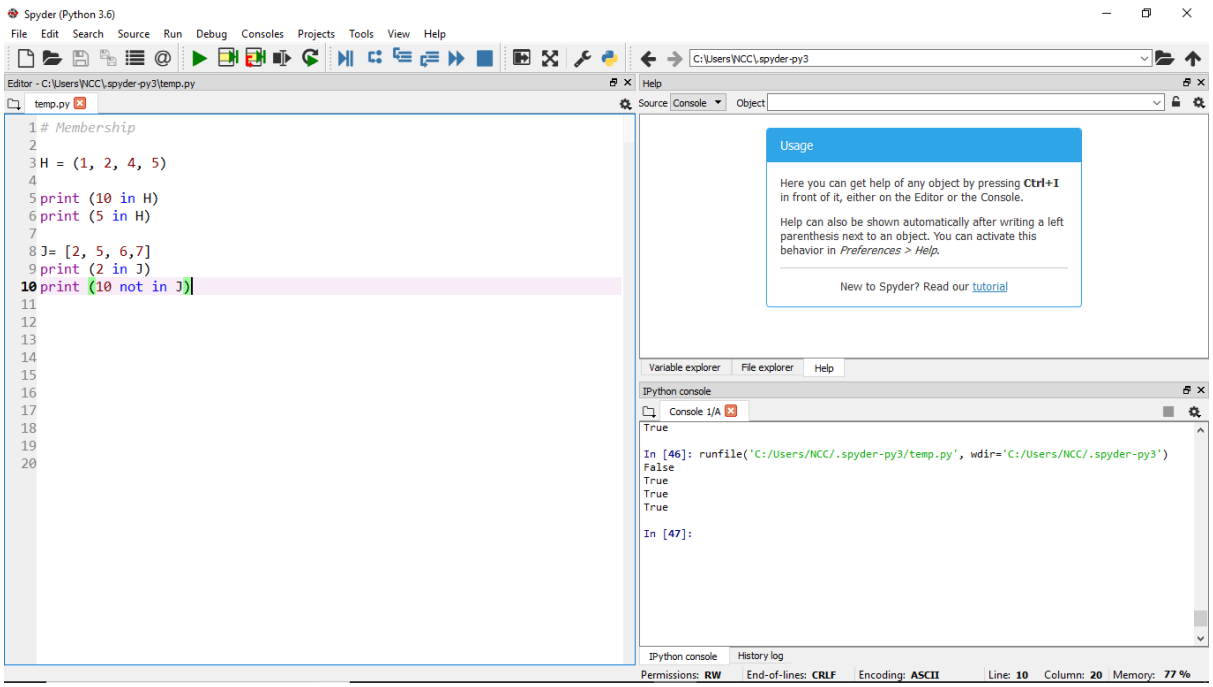


Figure 29 Membership

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.

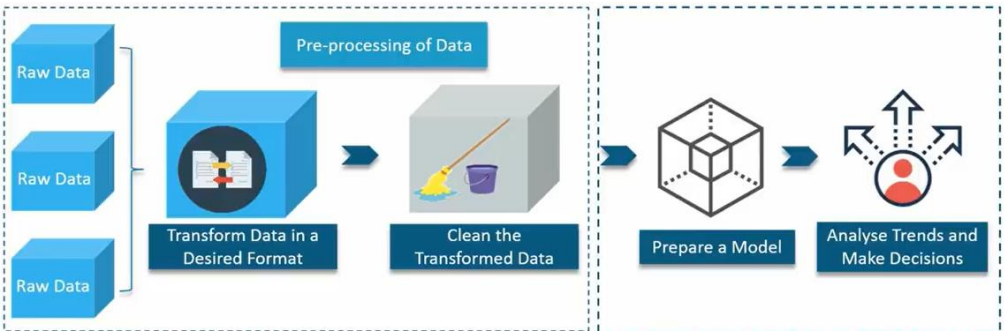


Figure 30 Data life cycle

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

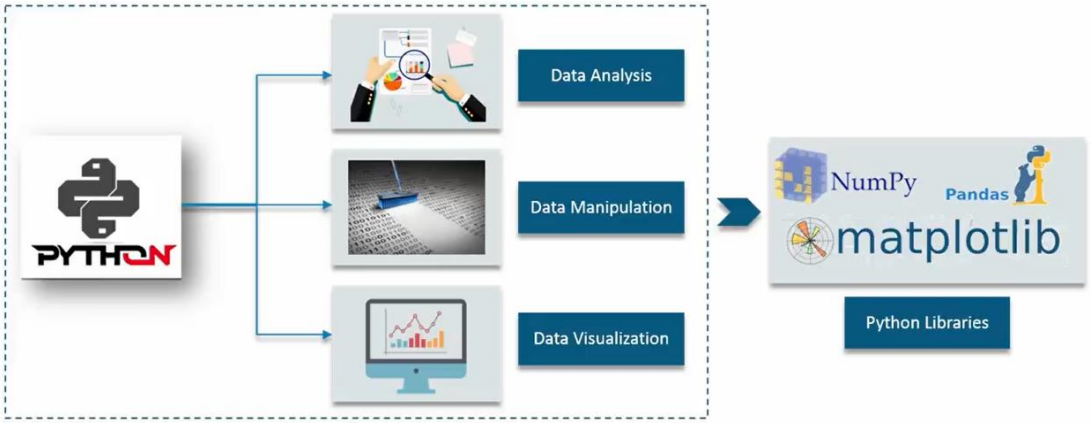


Figure 31 Data Analysis in Python

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.

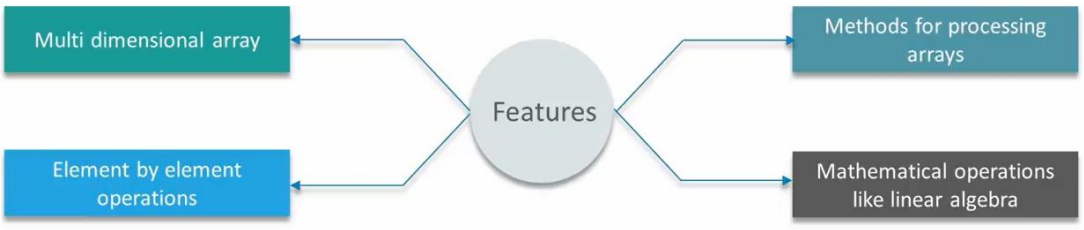


Figure 32 NumPy Features

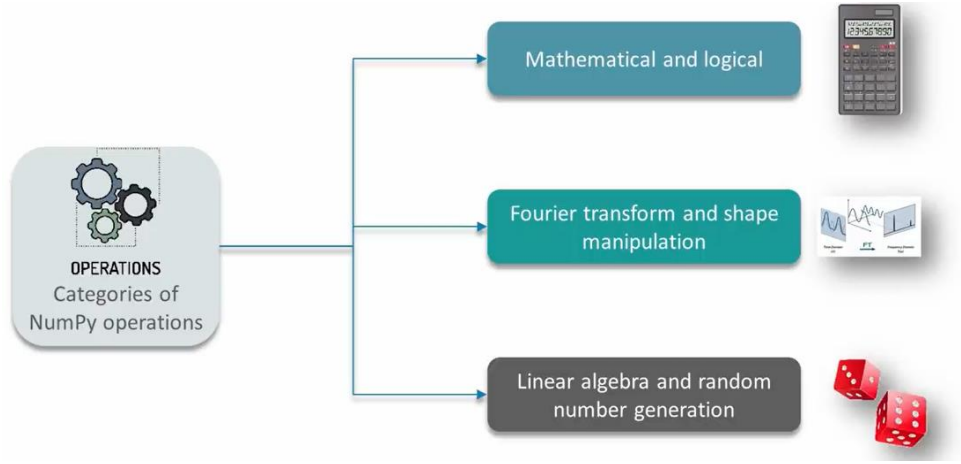
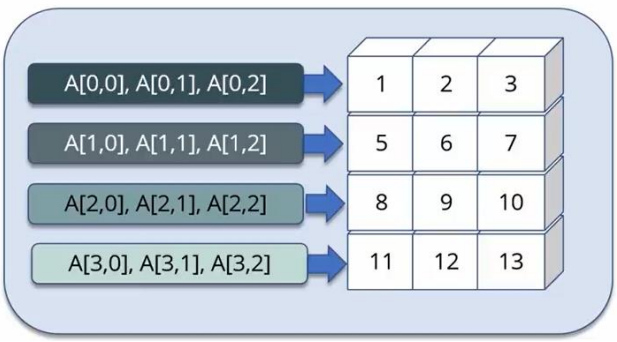
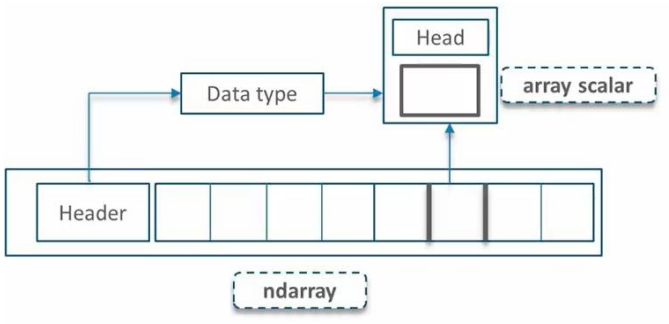


Figure 33 Operations in NumPy

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

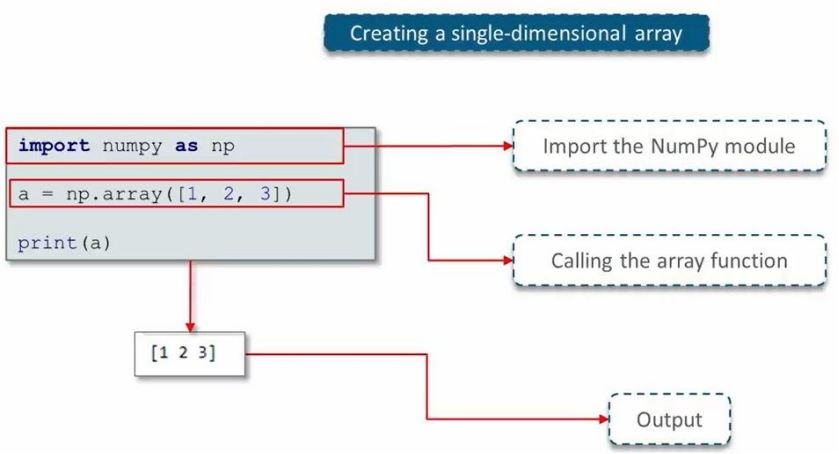


Each element in ndarray is an object of data-type object called *dtype*. An item extracted from ndarray, is represented by a Python object of an array scalar type (Please note that this done internally by Python).



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.



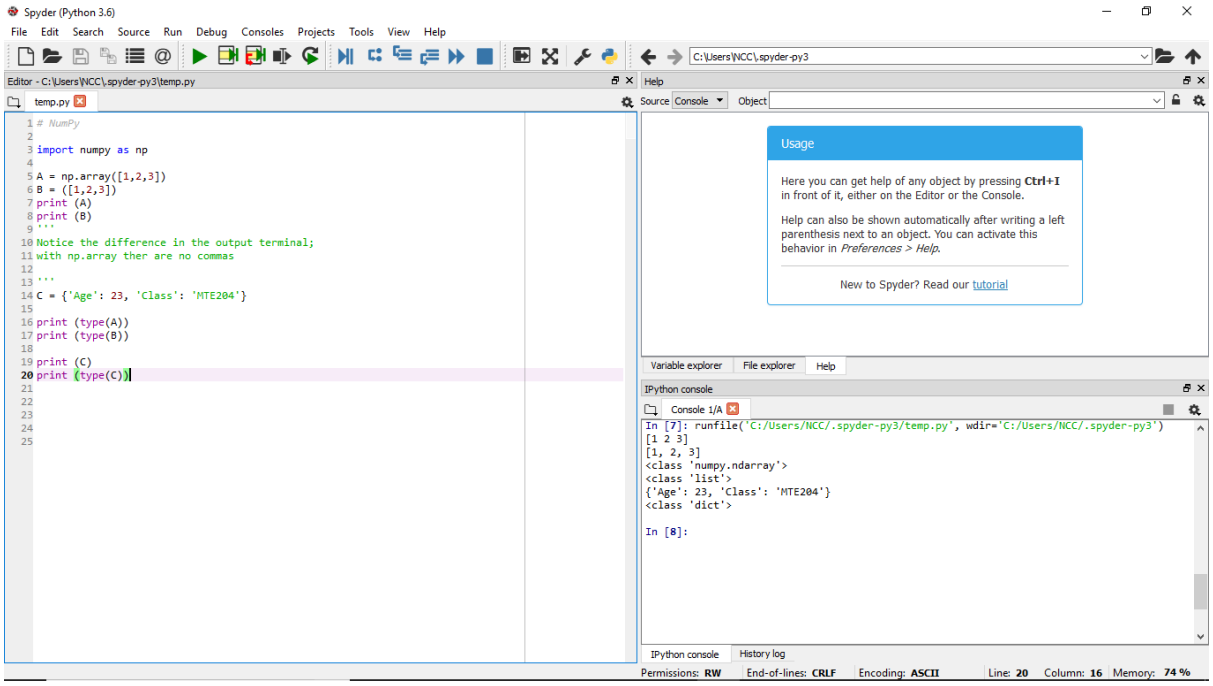


Figure 34 creating NumPy Array

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.

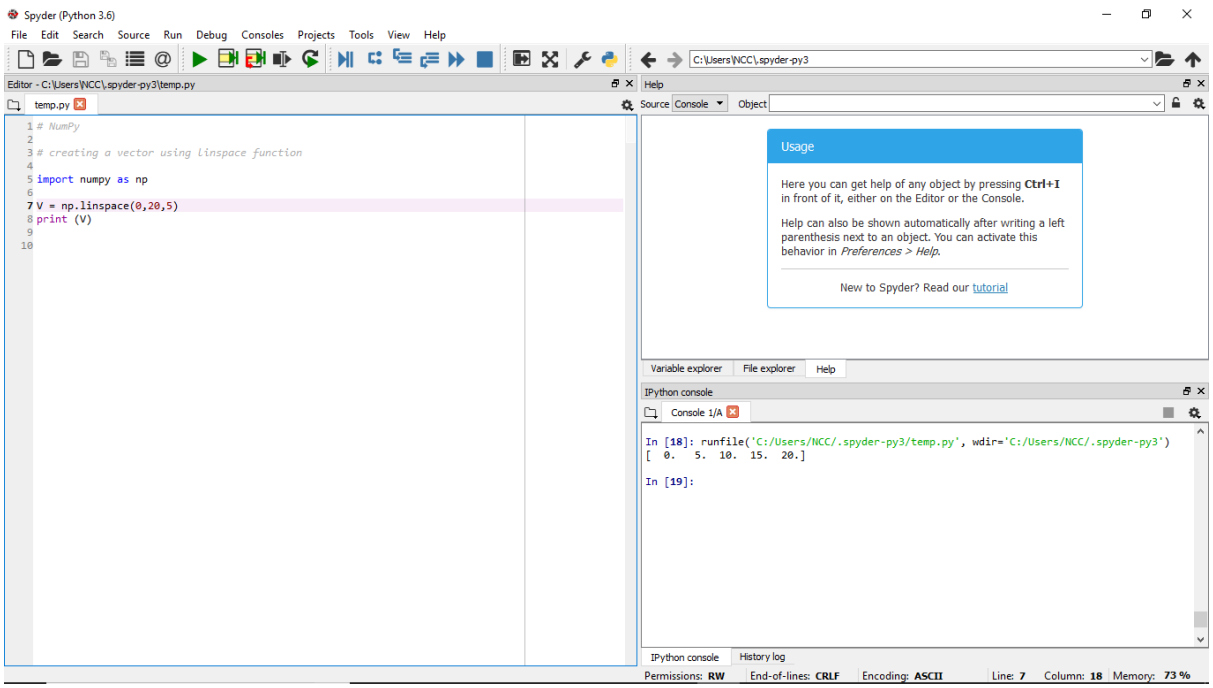


Figure 35 Linspece function

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.

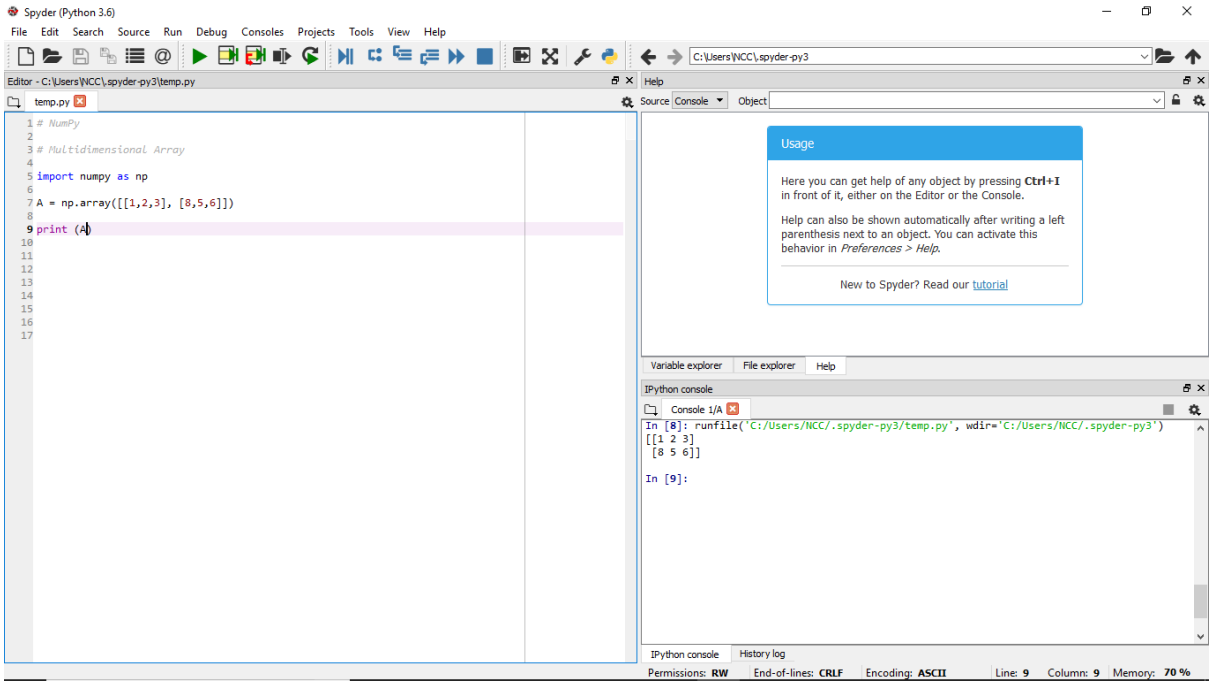


Figure 36 Multidimensional NumPy Array

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

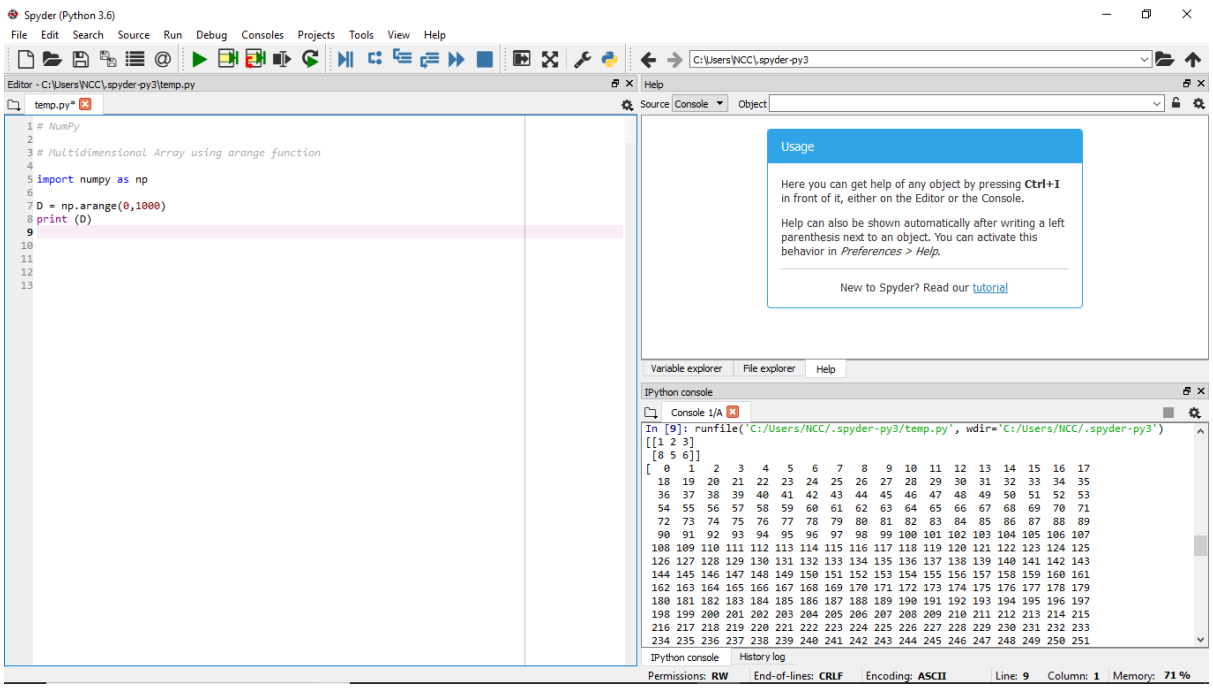


Figure 37 Arange function

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

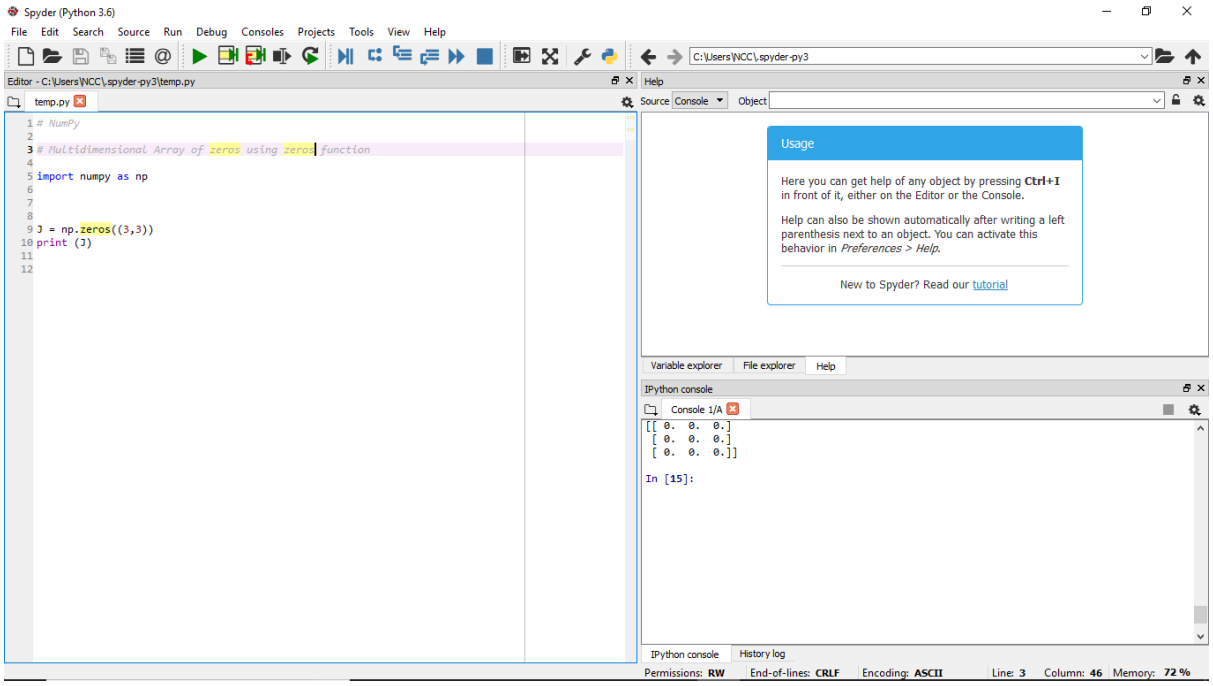
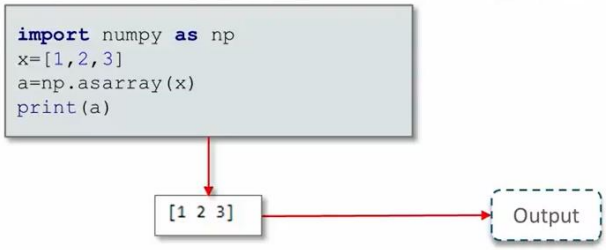


Figure 38 Array of zeros

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



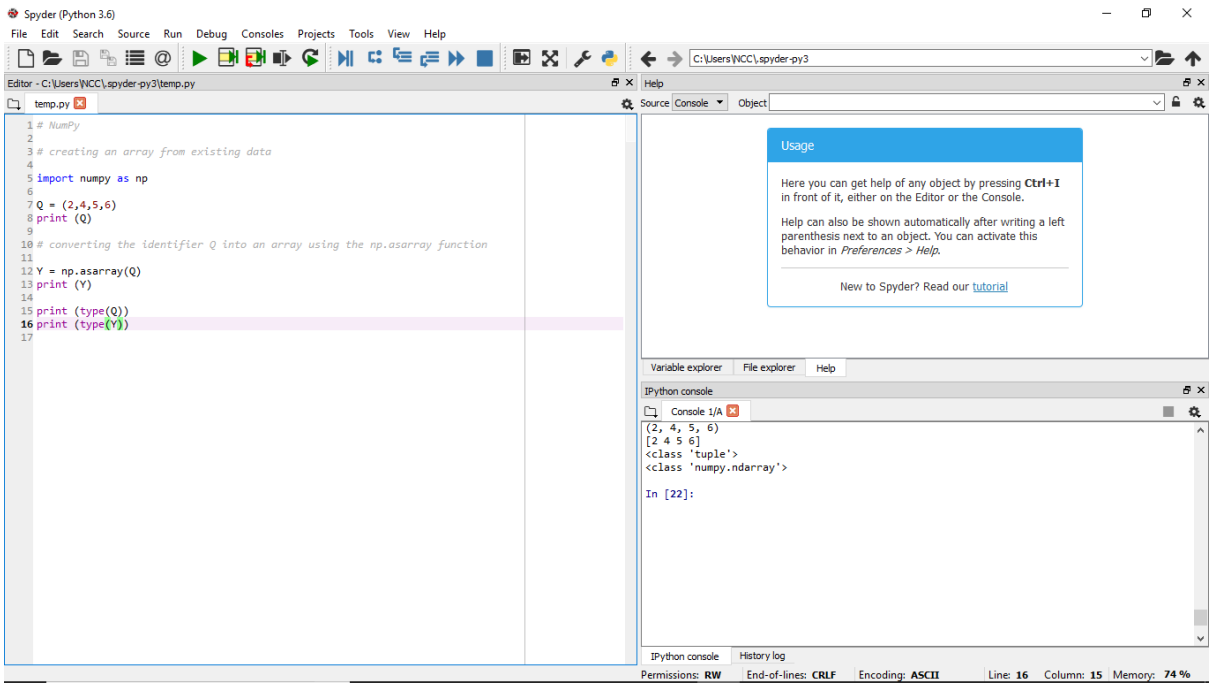
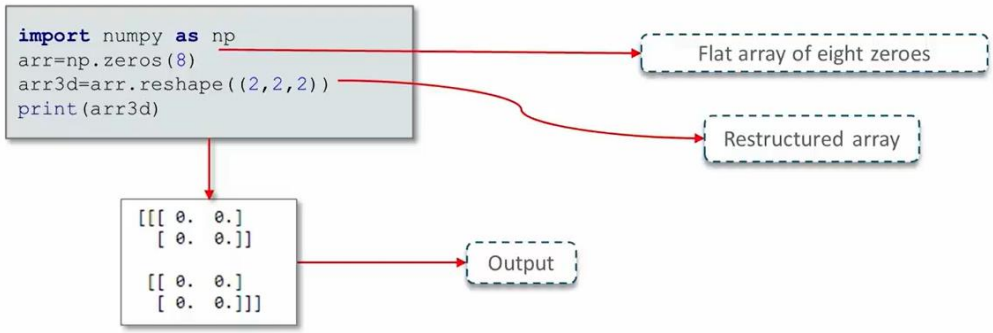


Figure 39 Creating an Array from Existing Data

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.



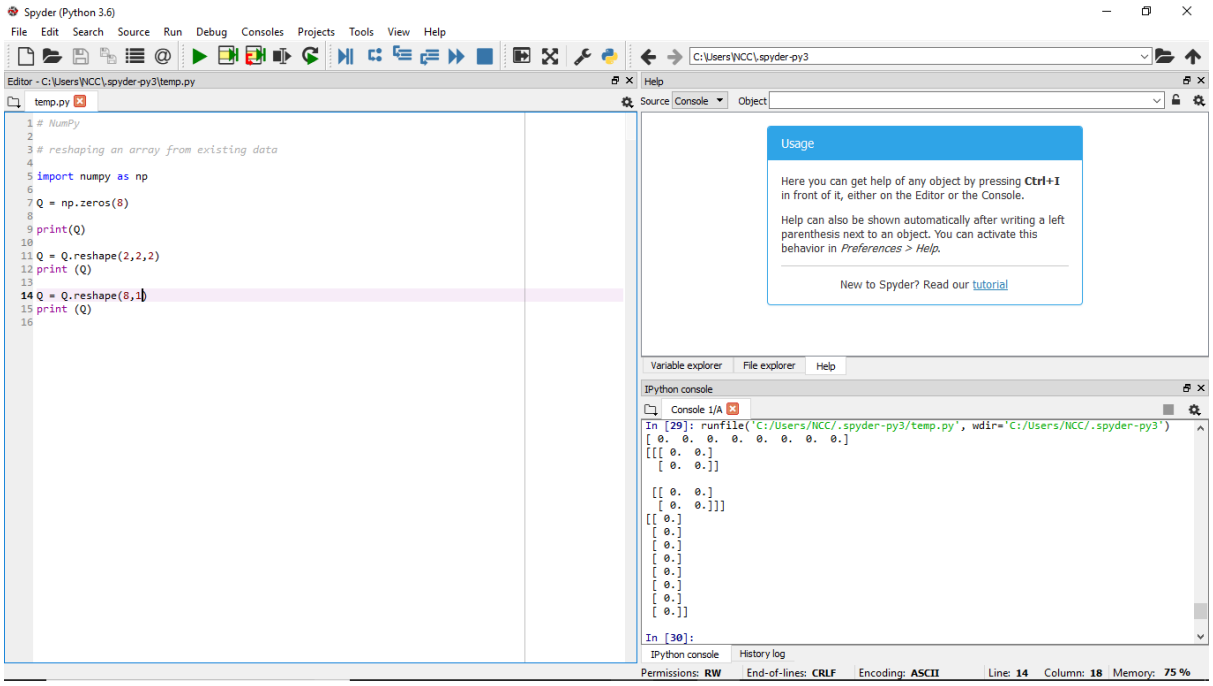


Figure 40 Restructuring a NumPy Array

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

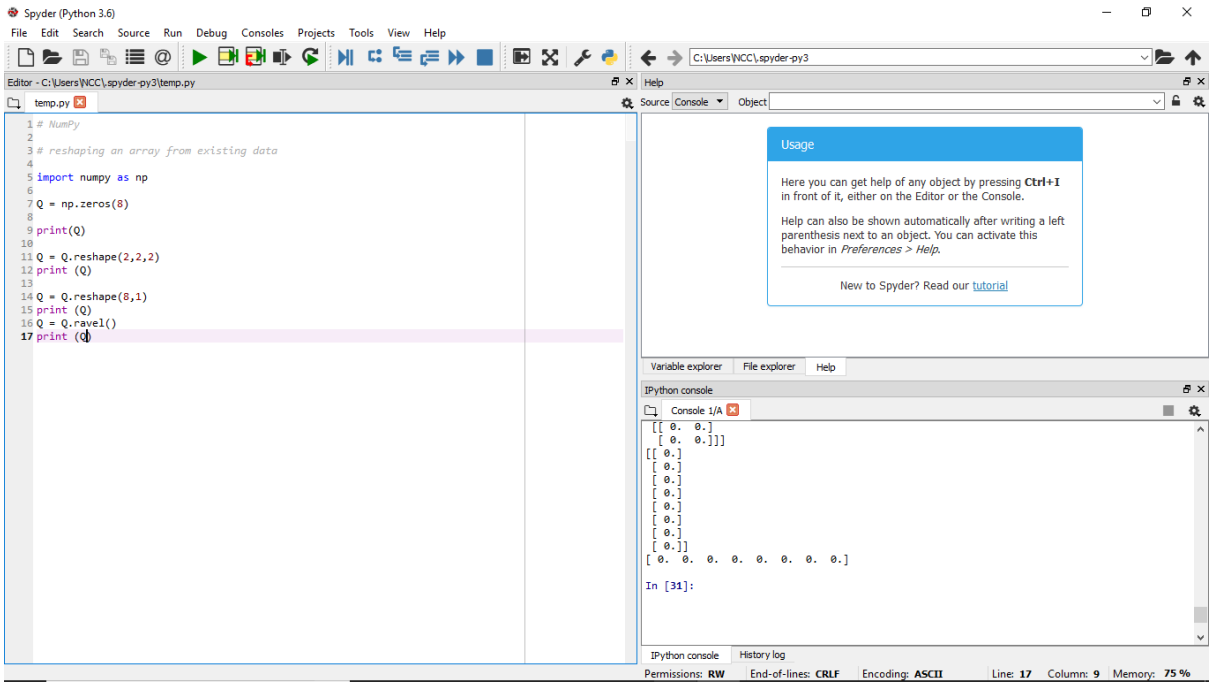


Figure 41 using the ravel function

2.1.5. Indexing a NumPy Array

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

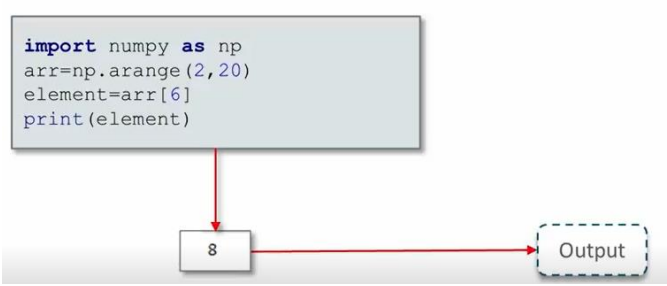


Figure 42 Indexing NumPy Array

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

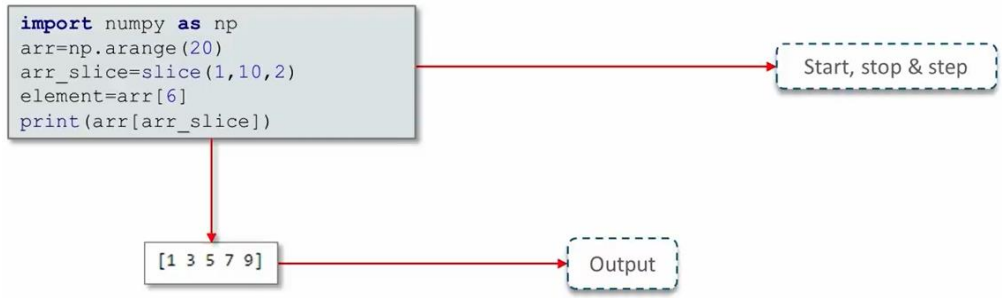


Figure 43 Slicing a NumPy Array

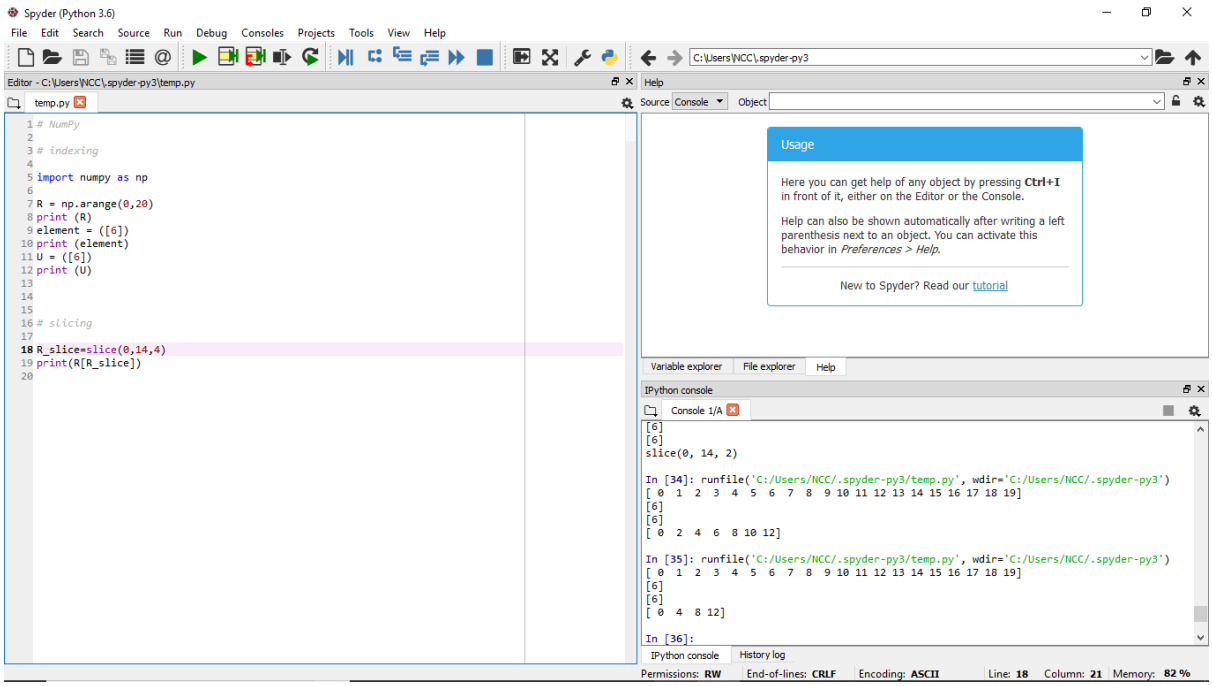


Figure 44 Indexing and Slicing of NumPy Array

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

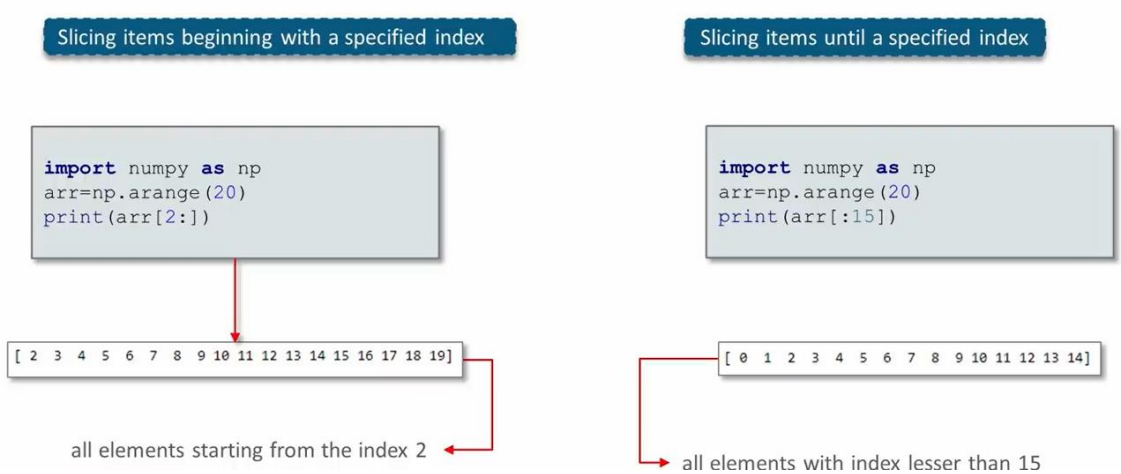


Figure 45 Other Slicing Methods

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

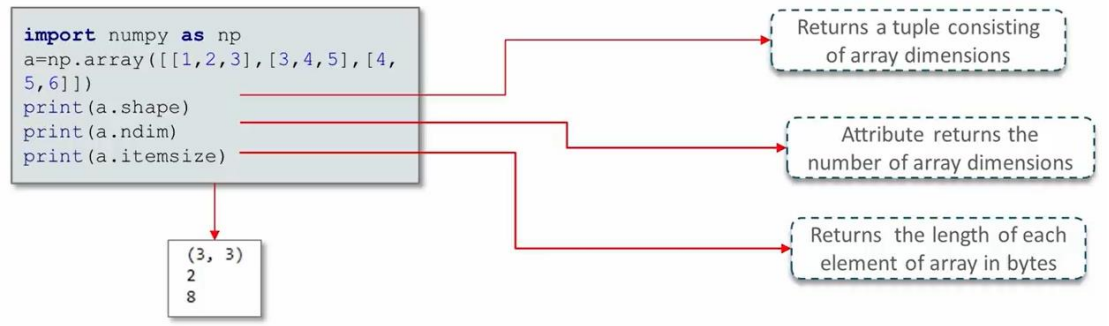
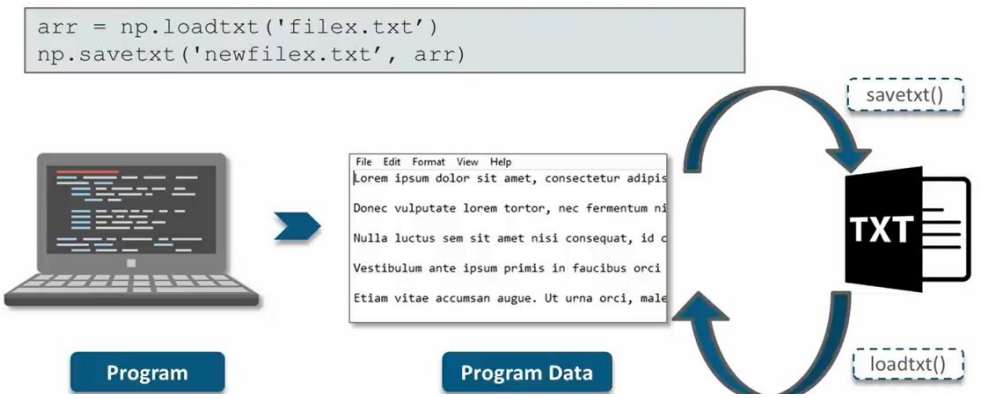


Figure 46 NumPy Array Attributes

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.



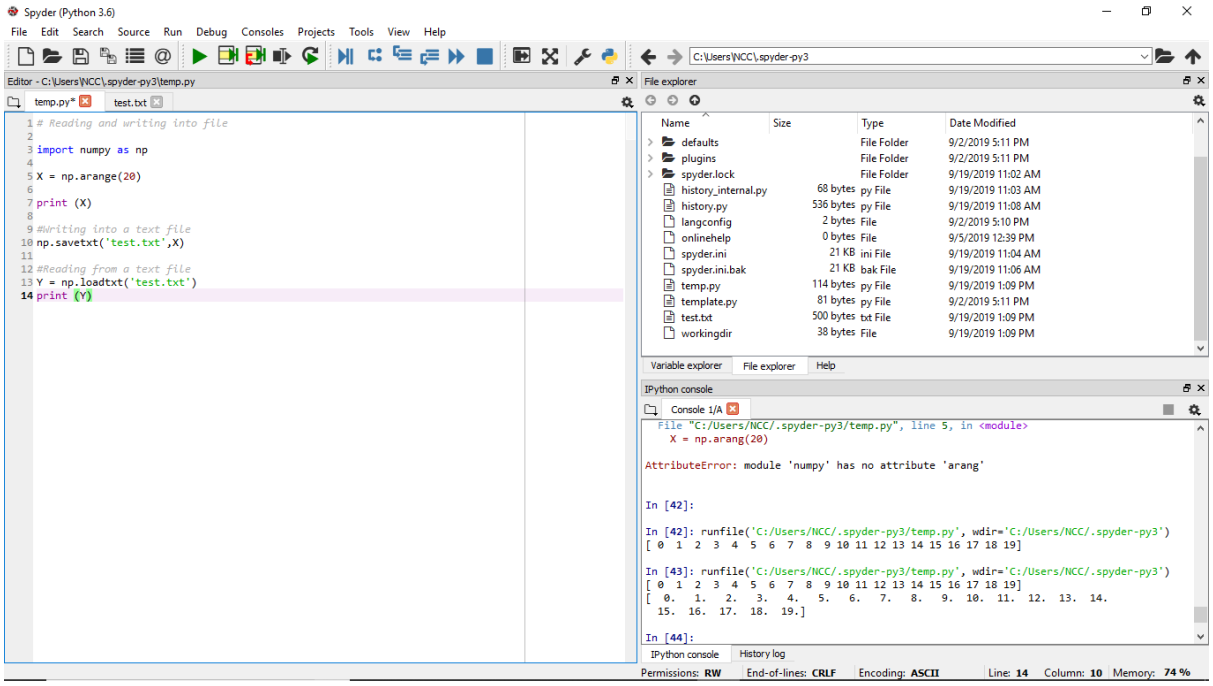
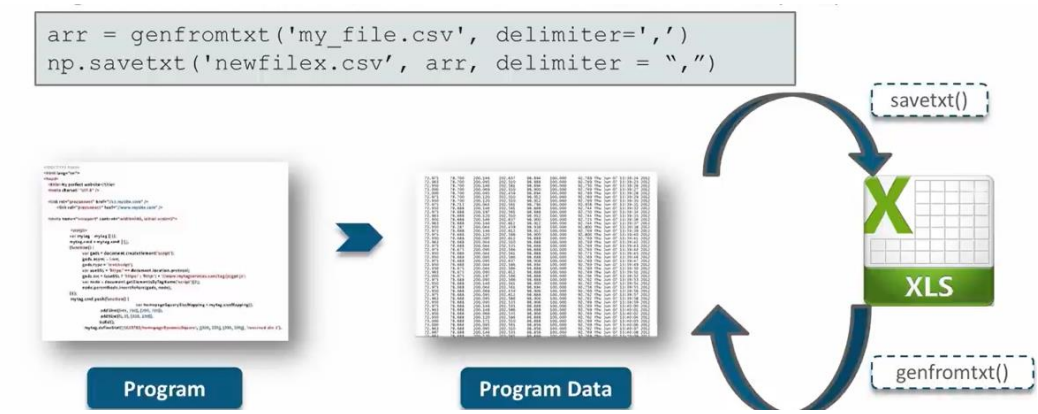


Figure 47 Writing and Reading from a Text file

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.



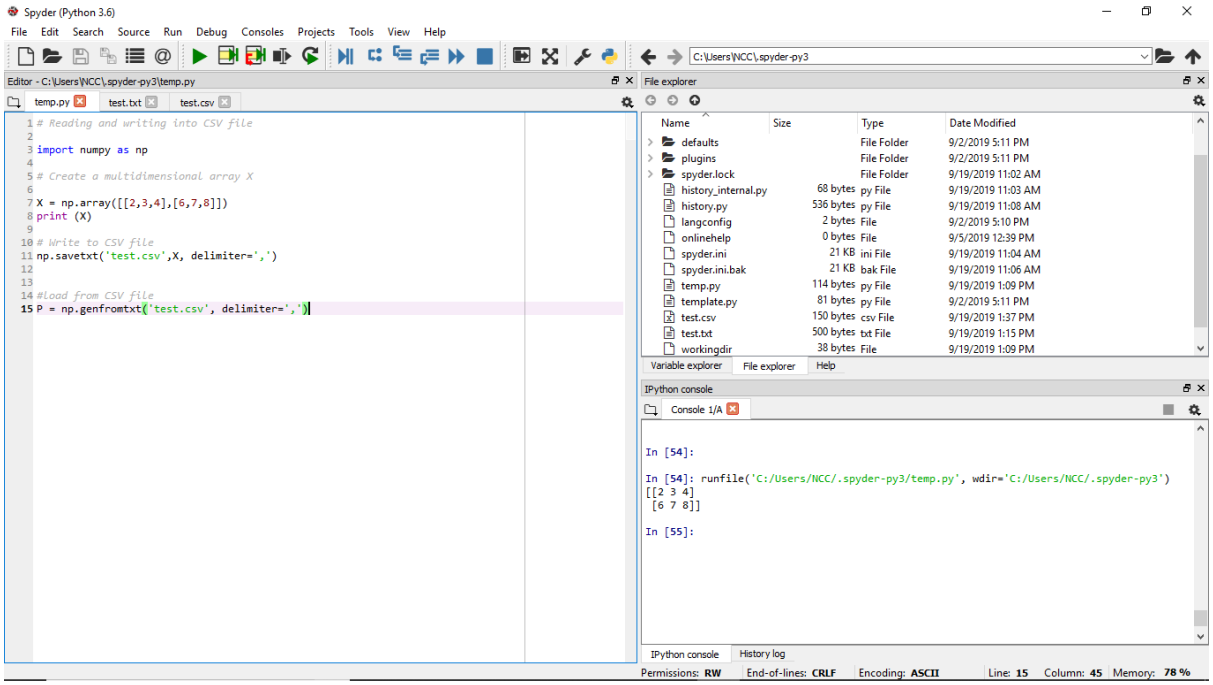


Figure 48 Writing and Reading from CSV file

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.

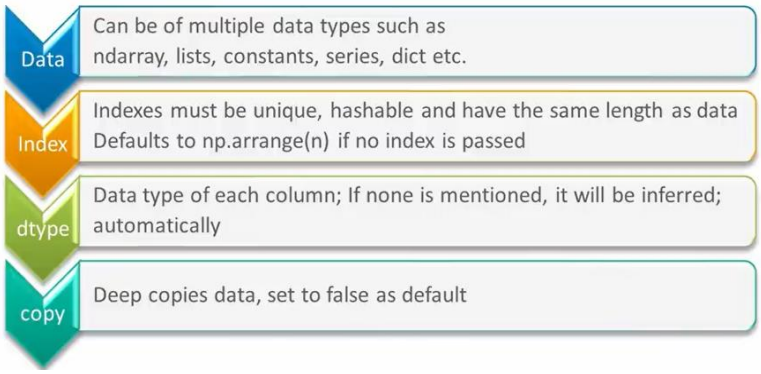


Fig. 31 presents the creation of a Series using the Pandas library.

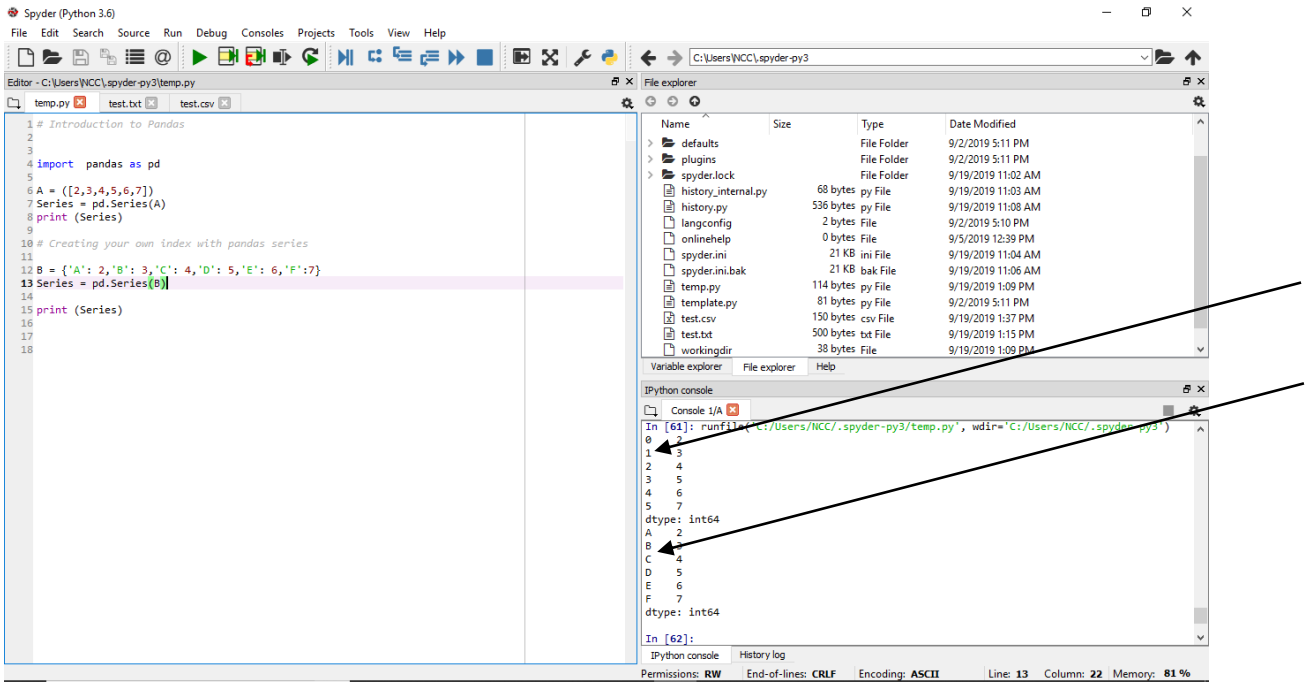


Figure 49 Creating Series Using Pandas Library in Python

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.

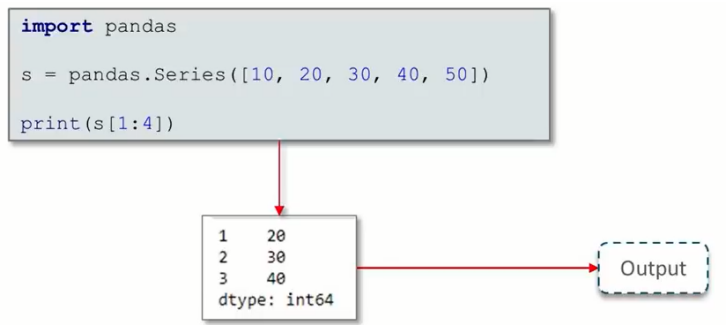


Figure 50 Accessing Data in Series Using Slicing

2.2.1.2.DataFrames

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

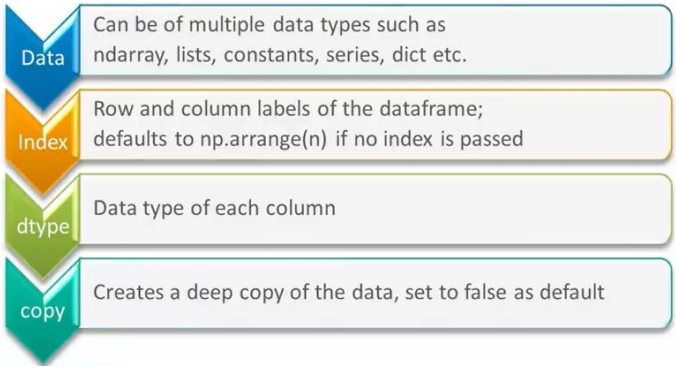


Fig. 33 shows how DataFrame is created. We will notice that Python automatically printed the default index as in the case of Series and a default column name of 0. We can input our own column name by using a dictionary to input the data.

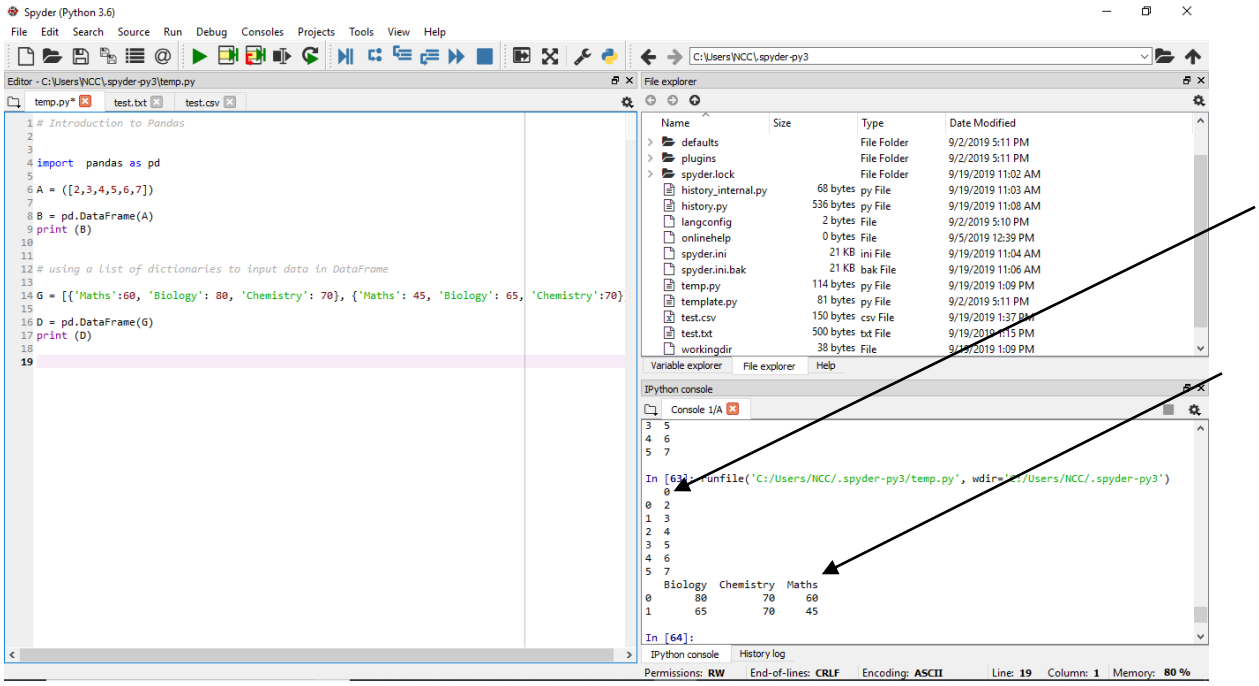


Figure 51 Creating DataFrame

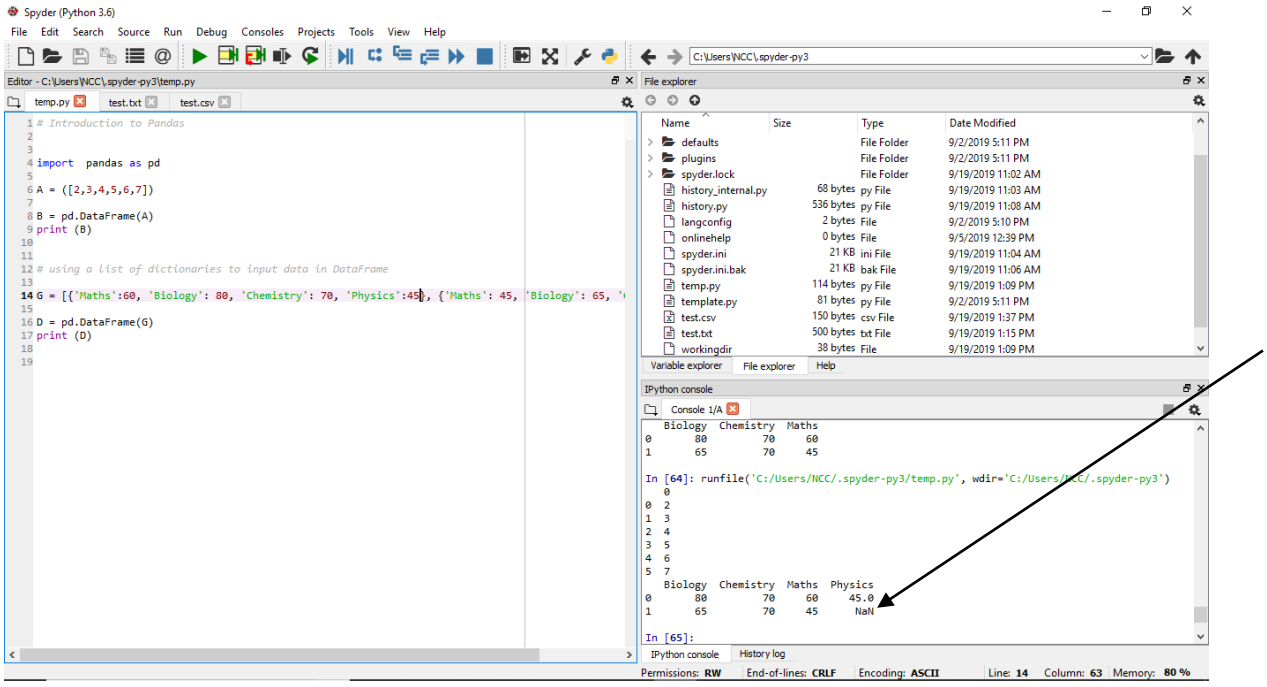


Figure 52 The NaN case

We can also specify our own index see Fig. 35

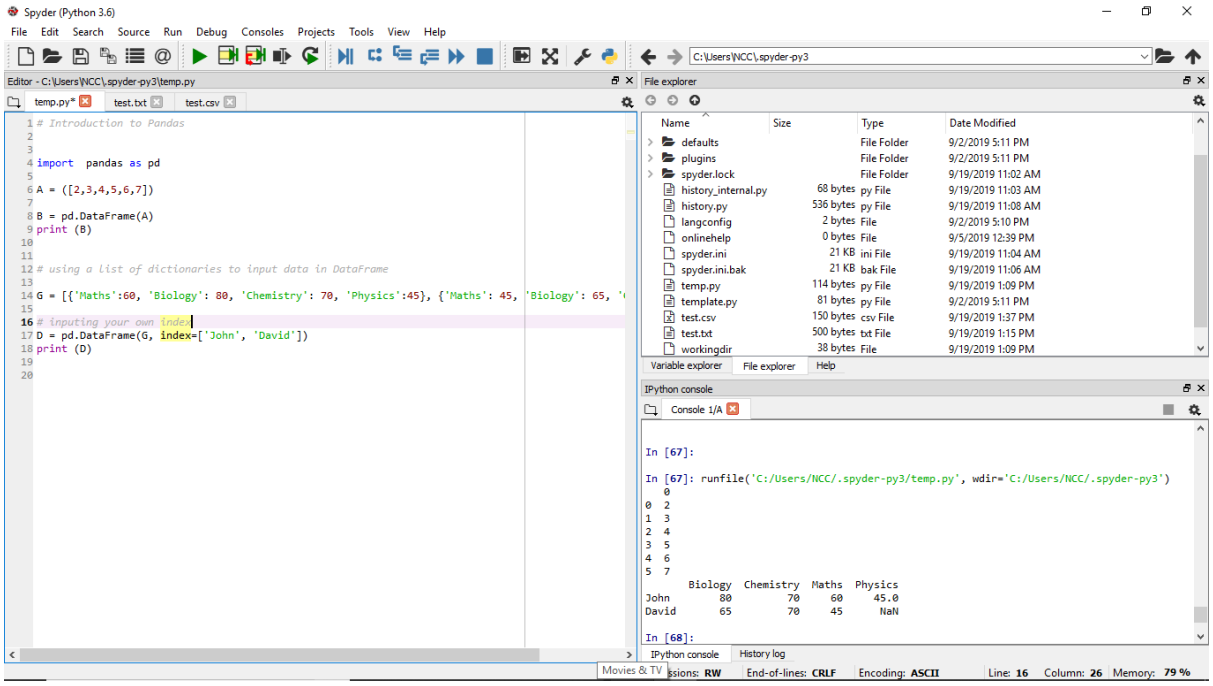


Figure 53 Using your own Index in Data Frame

Series can be converted to DataFrame see Fig. 36

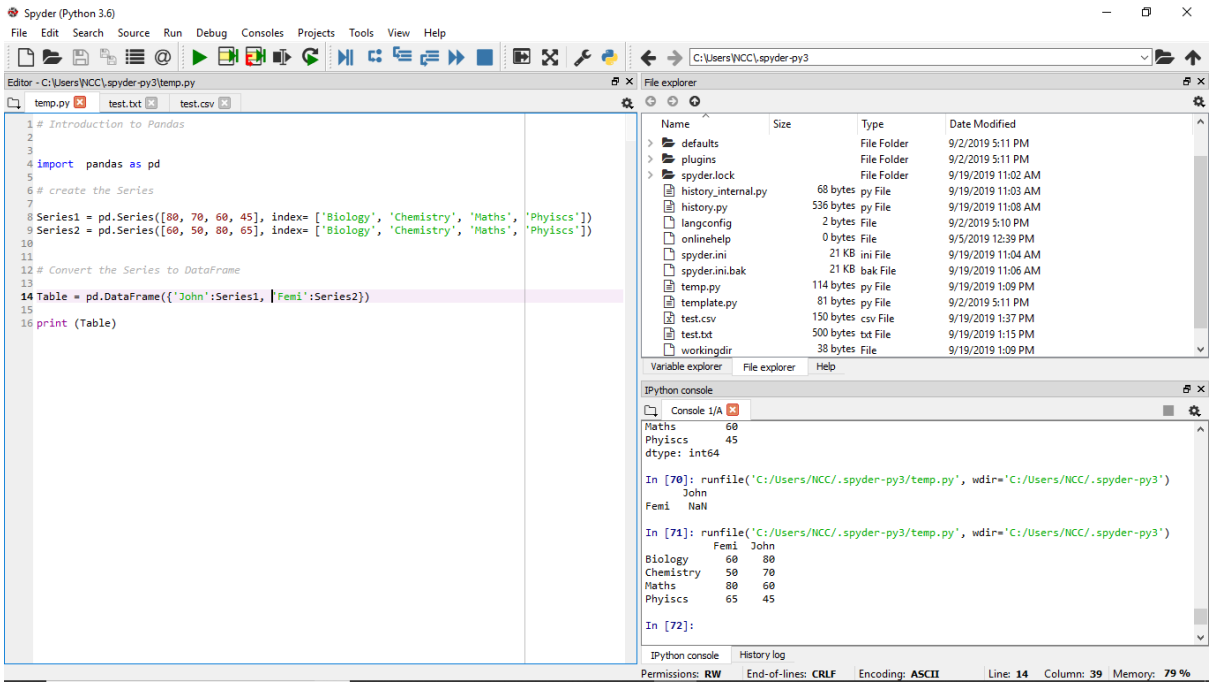


Figure 54 Converting Series to DataFrame

Column addition and deletion is possible with DataFrame

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

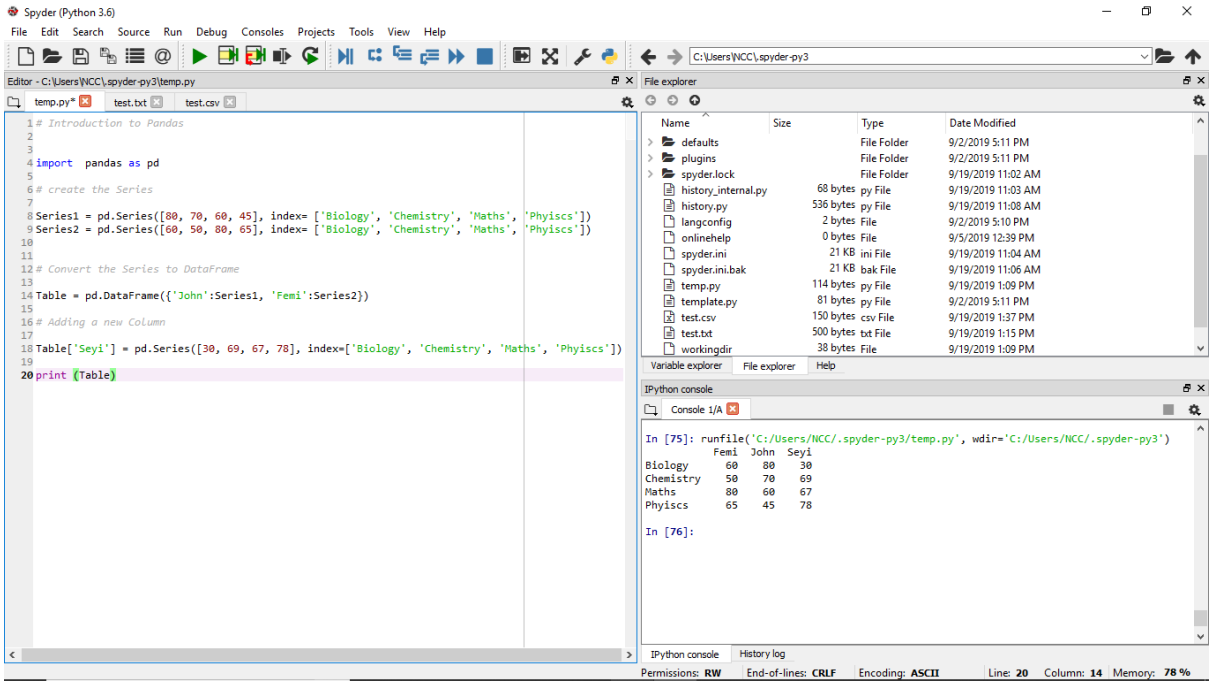


Figure 55 Adding a Column to a DataFrame

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

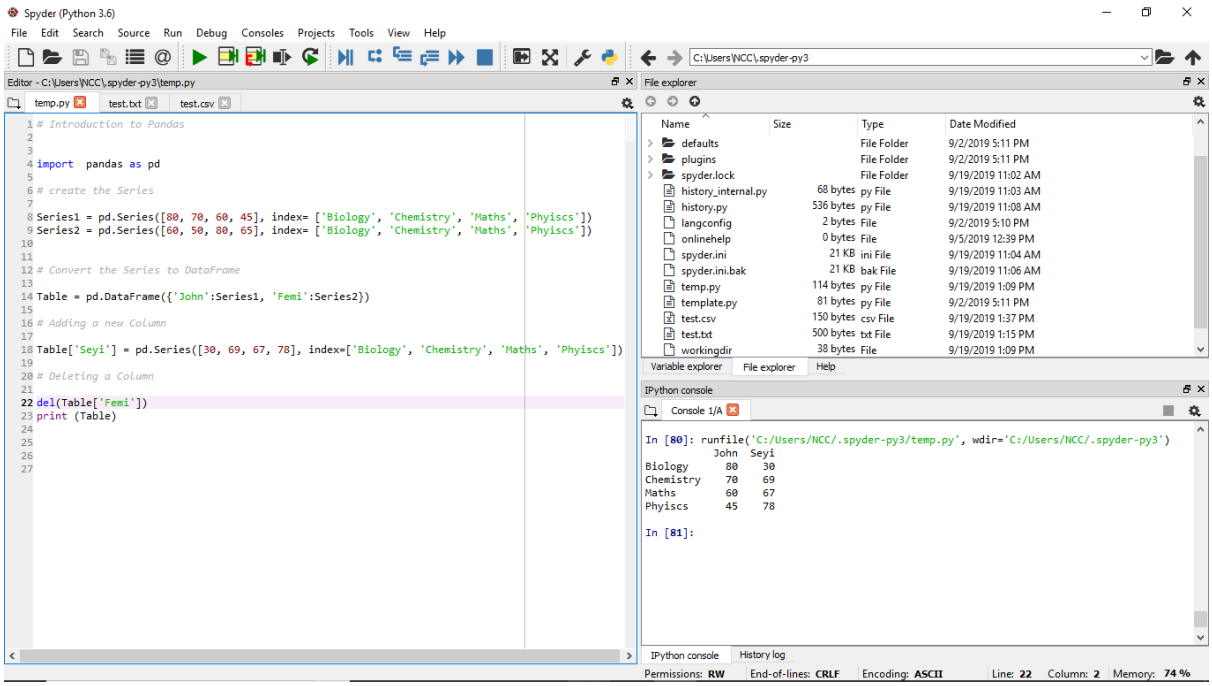


Figure 56 Deleting a Column from a DataFrame

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

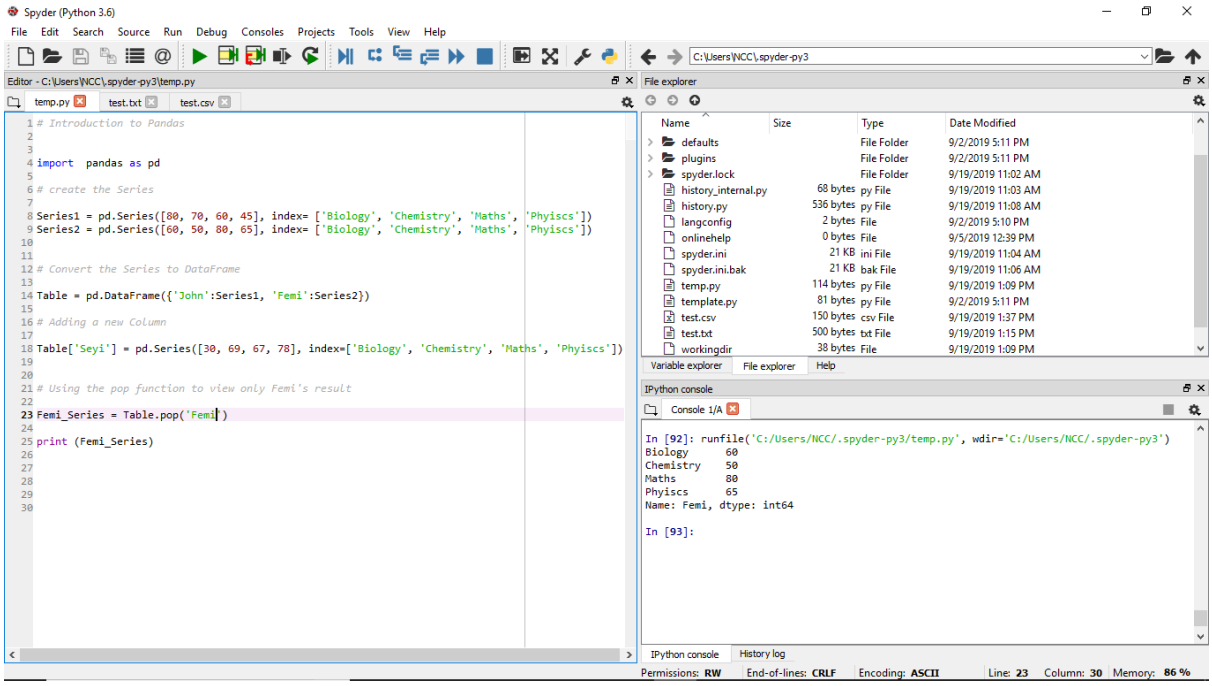


Figure 57 Using the pop function

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.

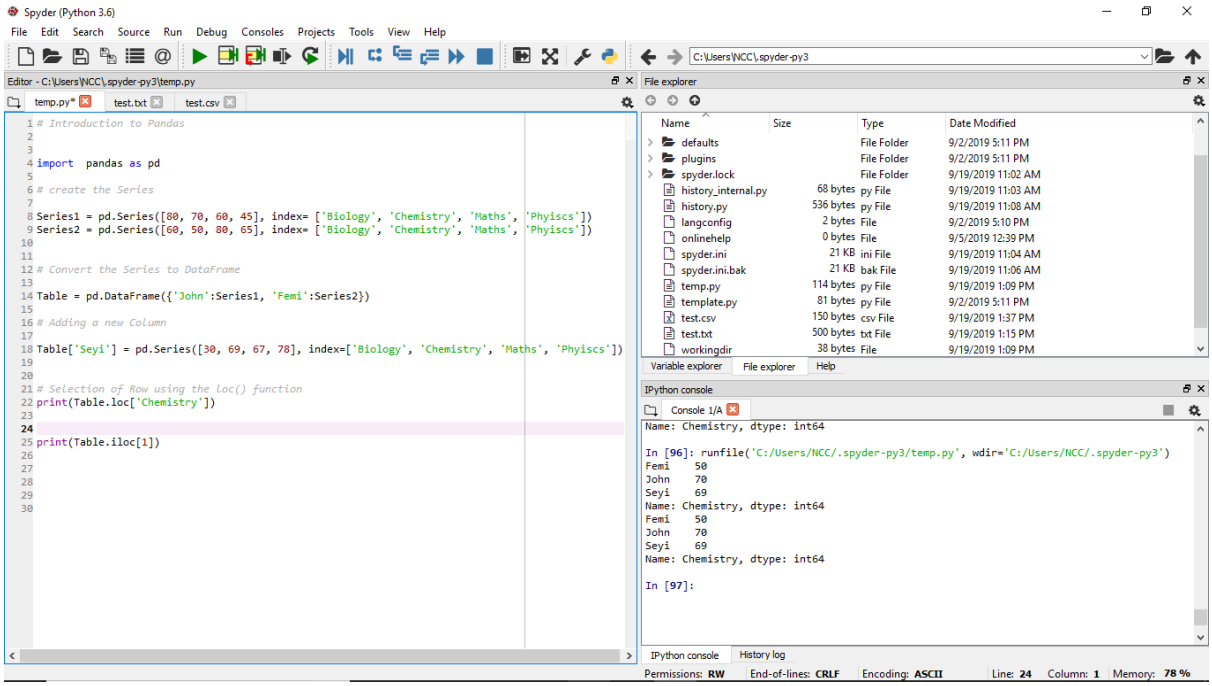
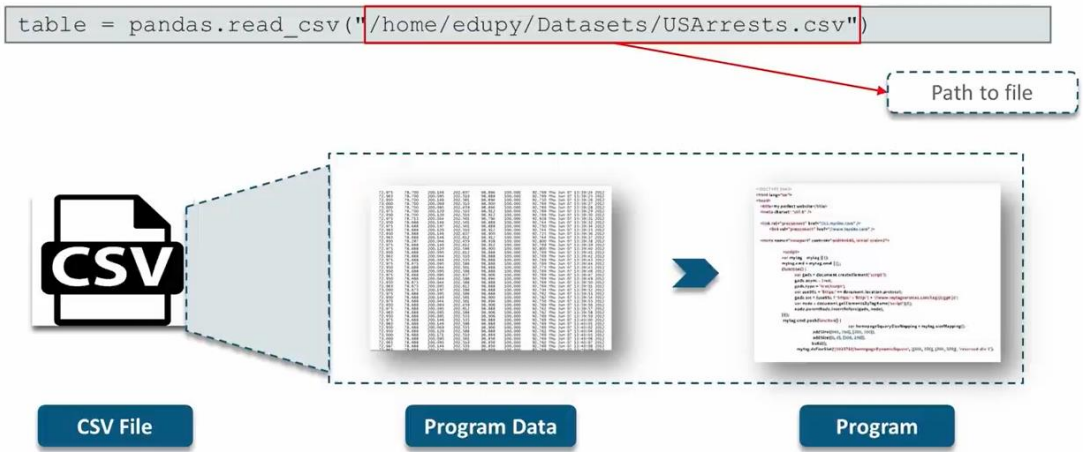


Figure 58 Using the loc function for row selection

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

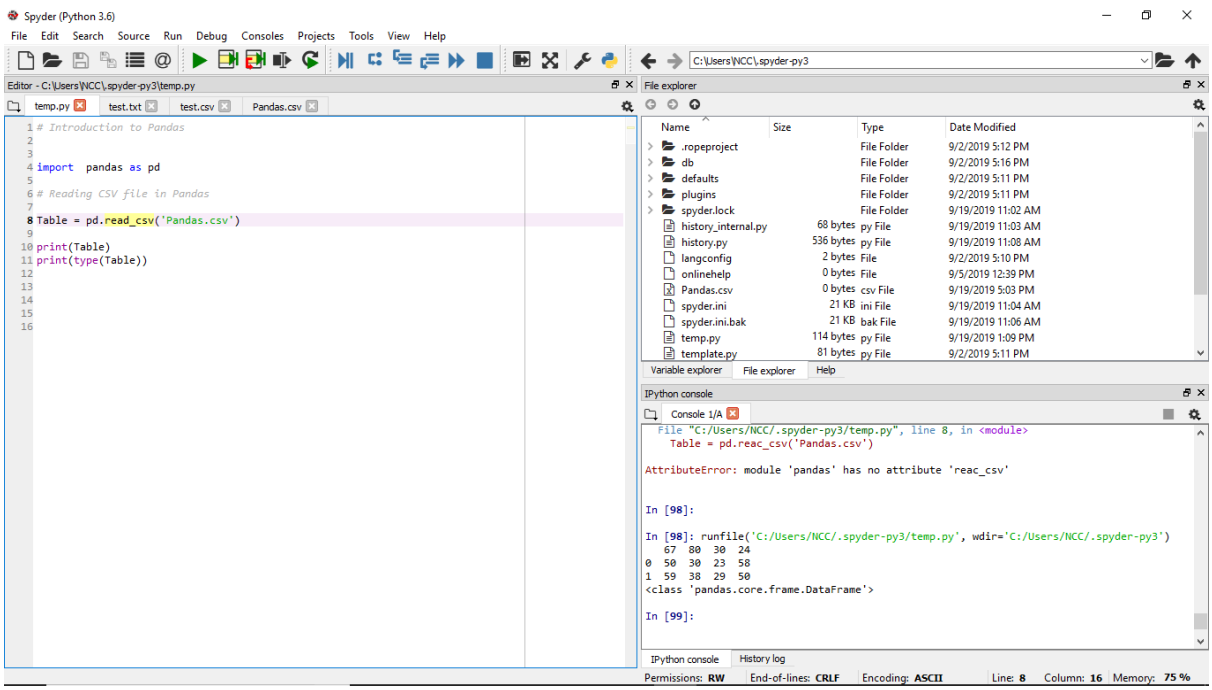
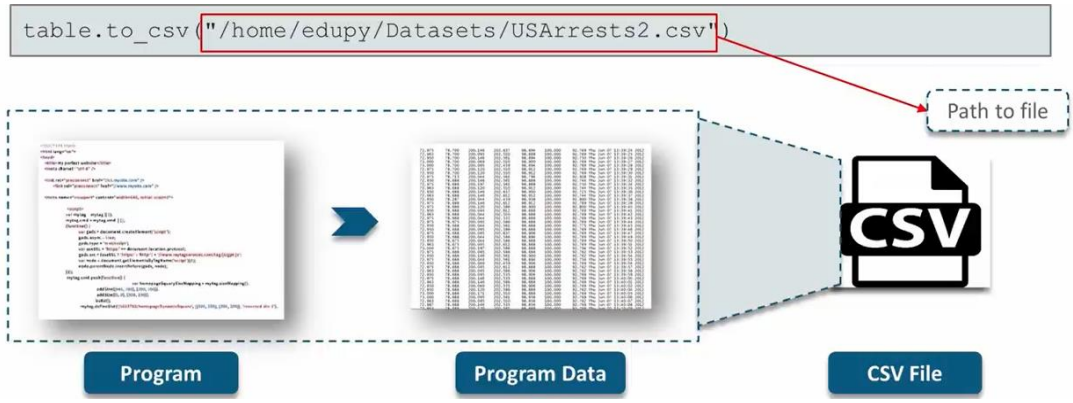


Figure 59 Reading CSV file to Pandas Library

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

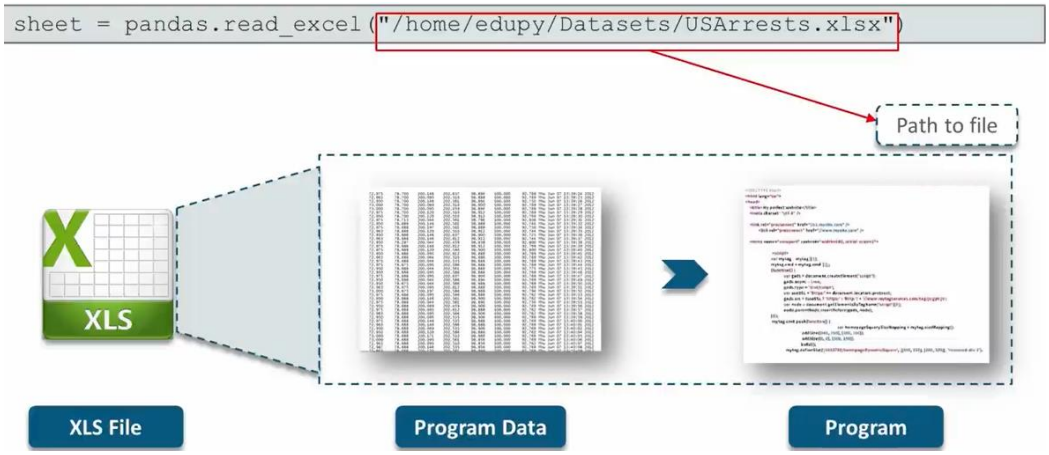


Figure 60 Reading from Excel file

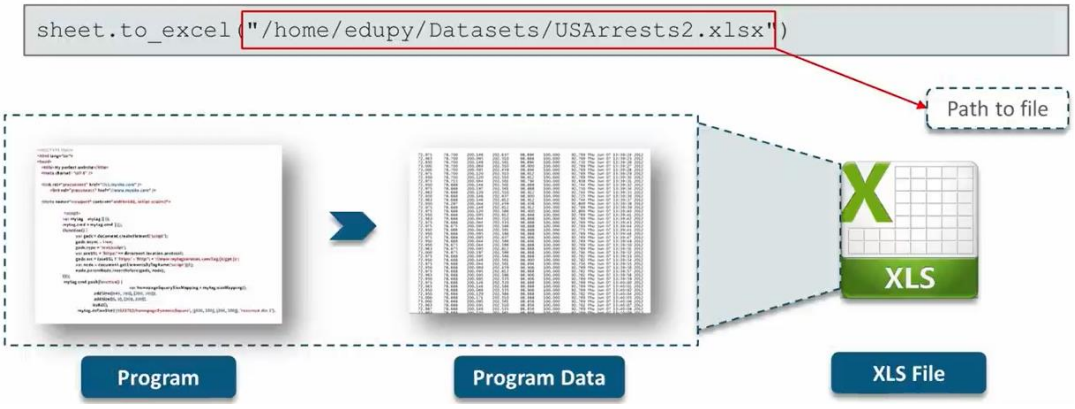


Figure 61 Writing to Excel file

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.

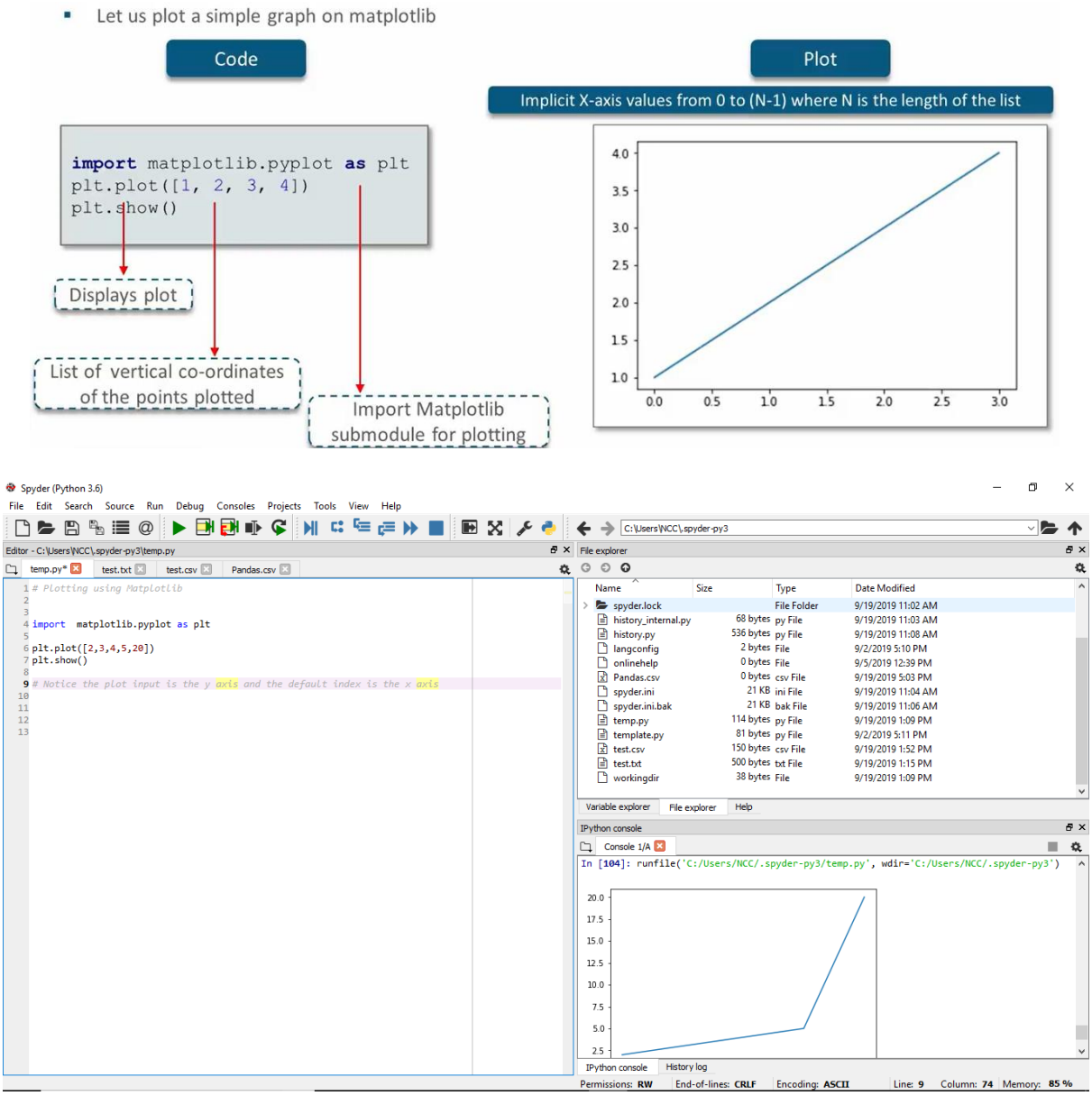


Figure 62 Using Matplotlib Library in Python

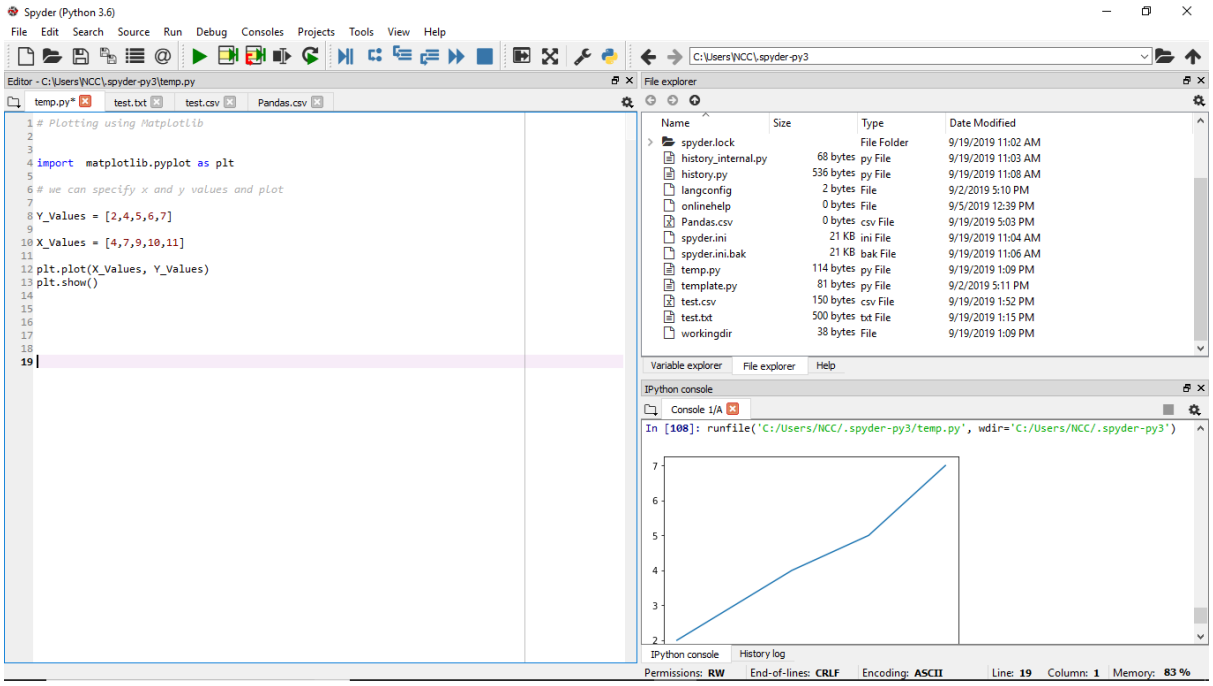
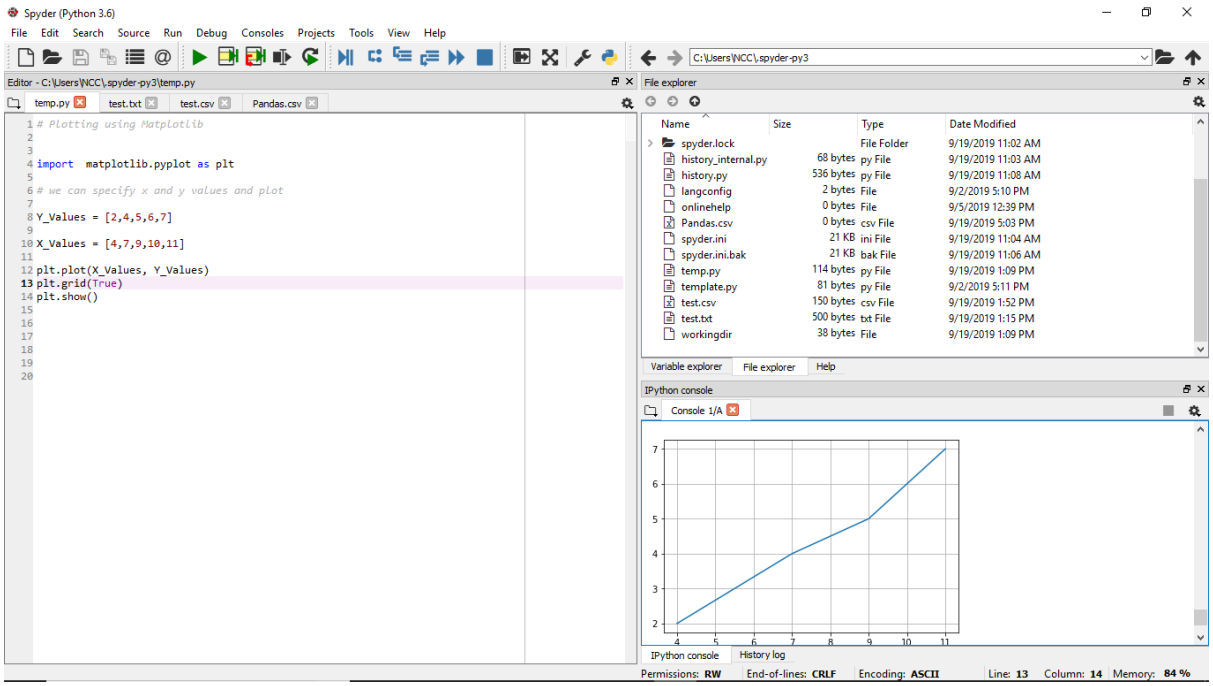


Figure 63 Specifying the x and v values in Matplotlib

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

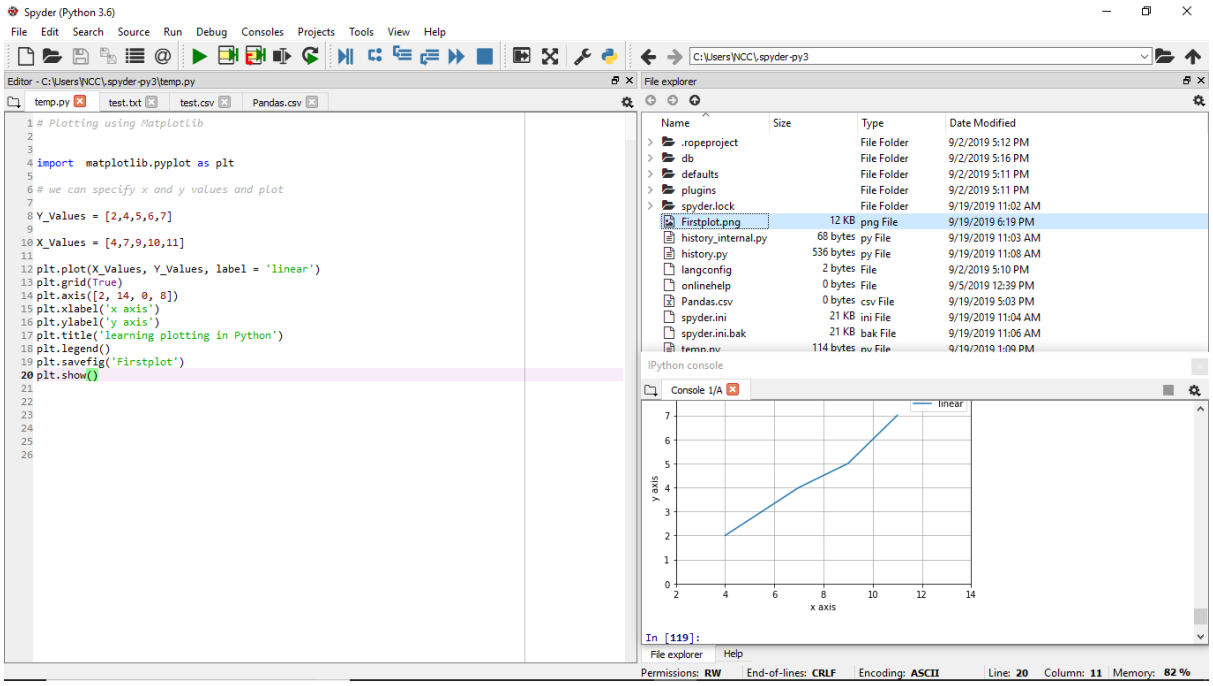


Setting of axis is done by using *plt.axis([xmin, xmax, ymin, ymax])*

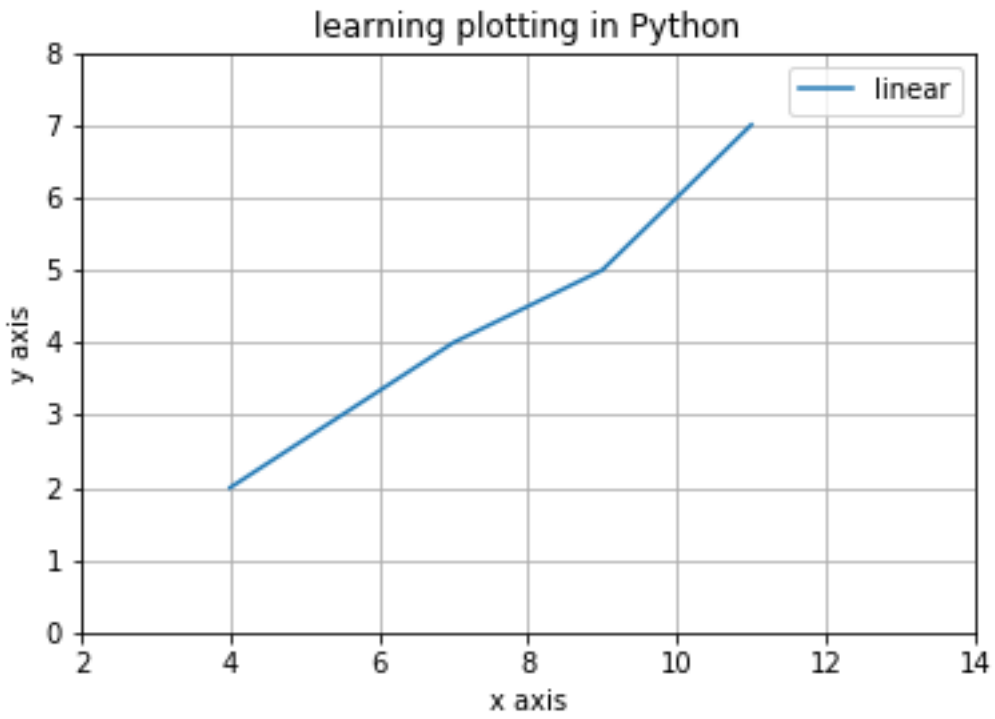
Labels can be added to x and y axis using *plt.xlabel('X axis')*and *plt.ylable('Y axis')*

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')*

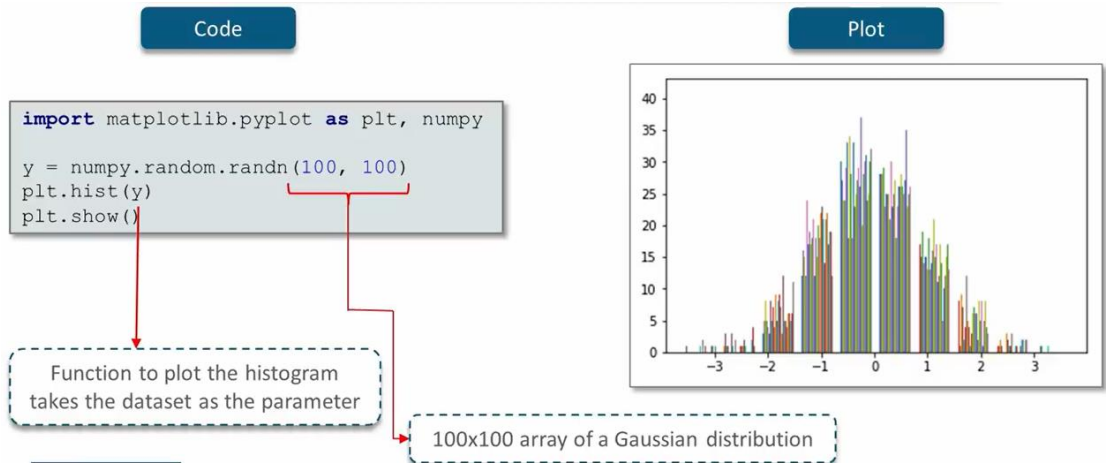


2.3.2. Plot Types

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.



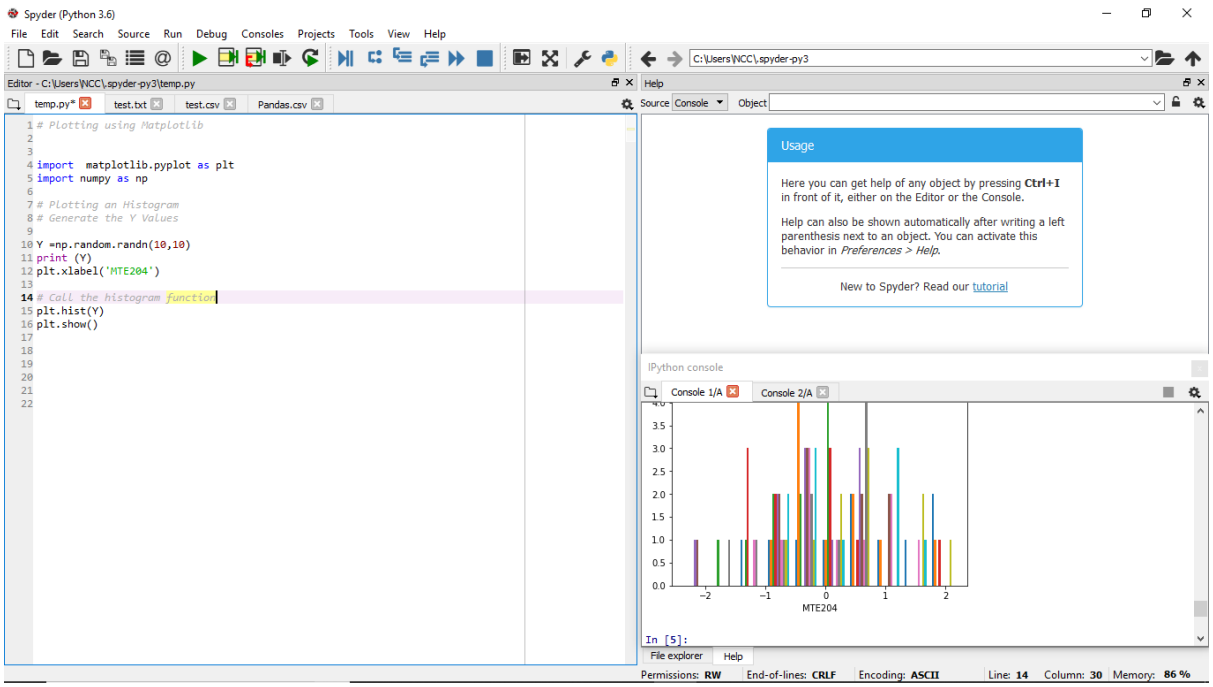


Figure 64 Plotting an Histogram in Python

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.

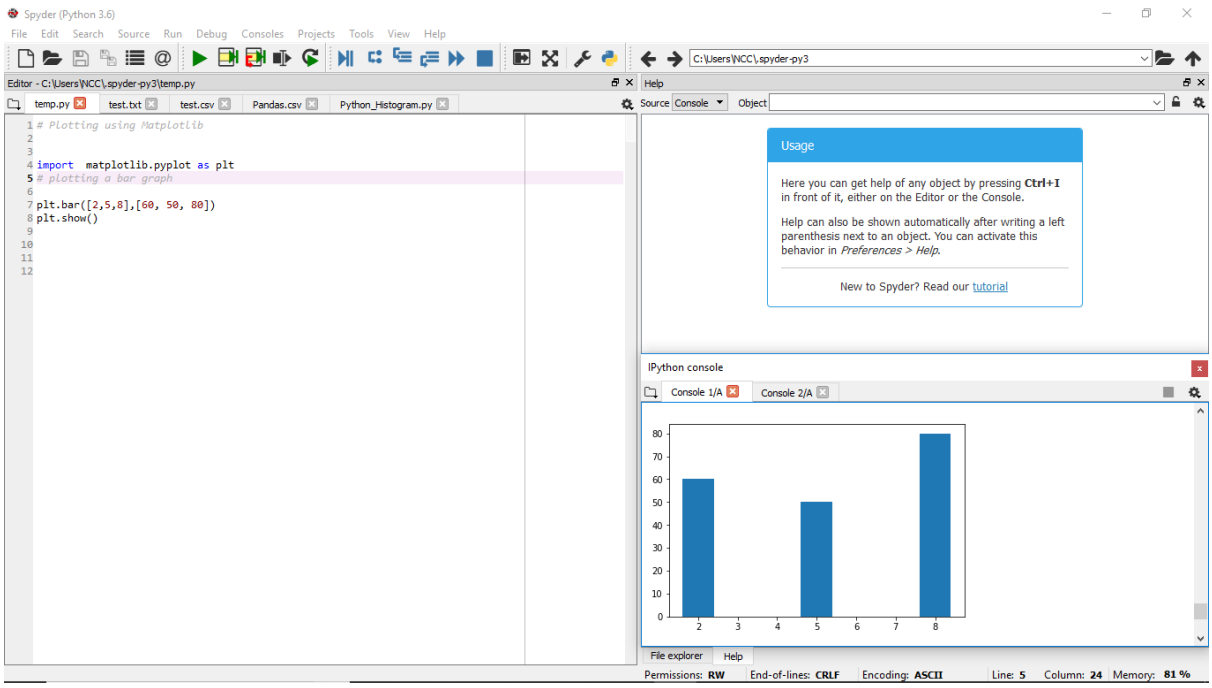
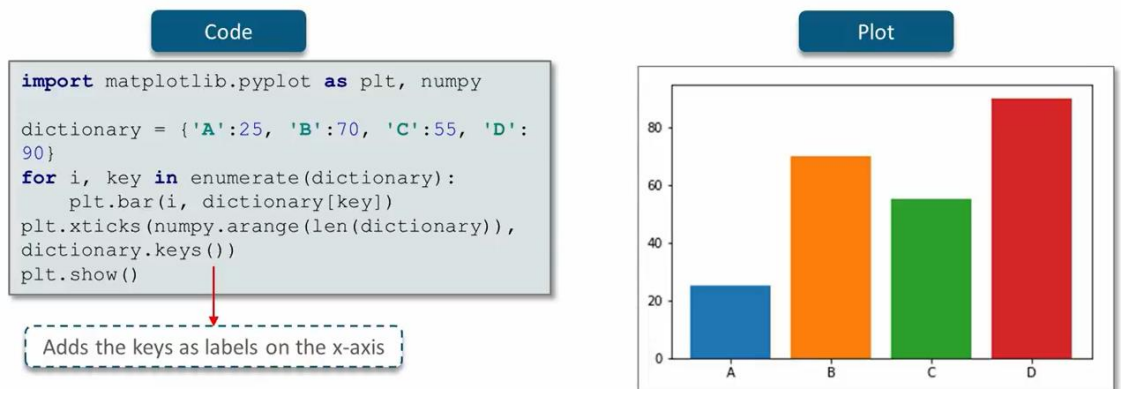


Figure 65 Bar graph in Python

We Can Plot a Dictionary Using Bar Chart



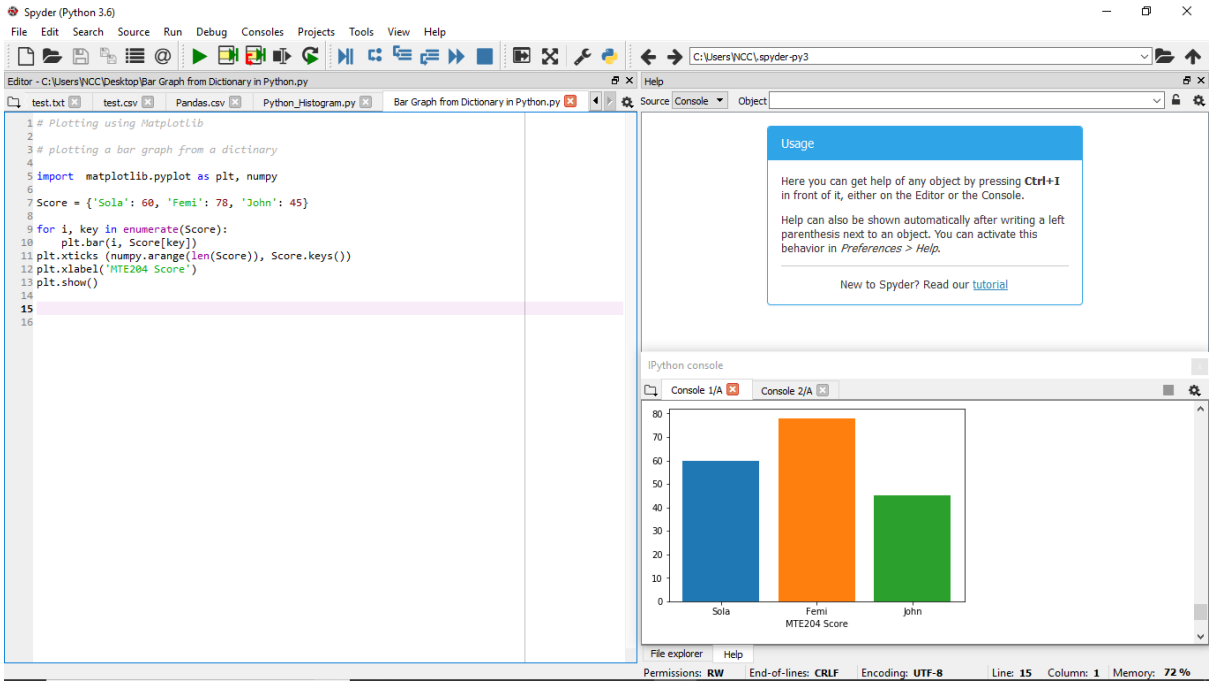


Figure 66 Plotting Bar graph from Dictionary

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

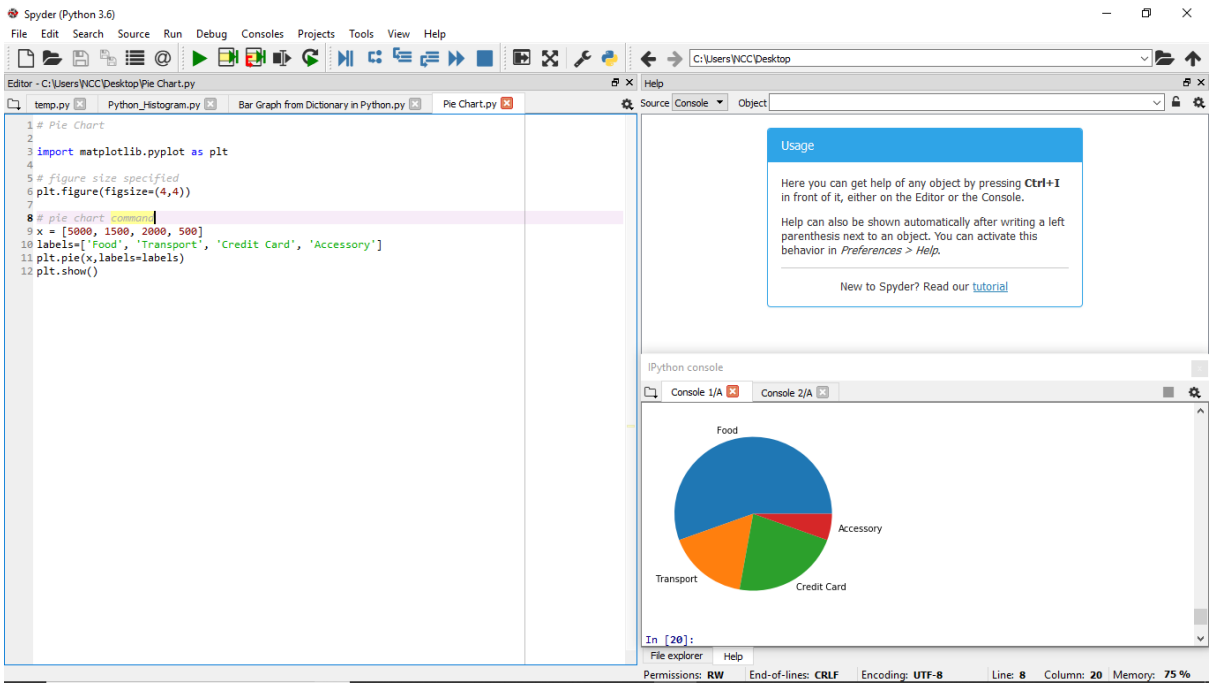
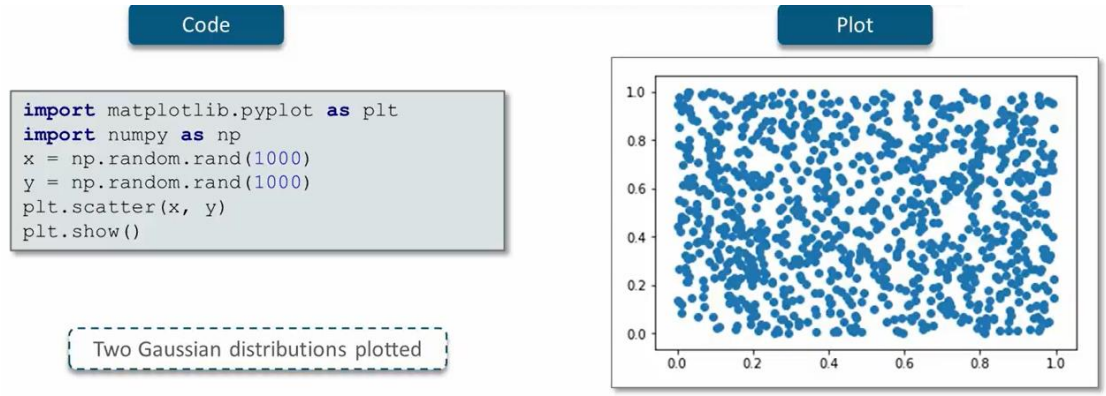


Figure 67 Pie chart in Python

4. Scatter plot

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



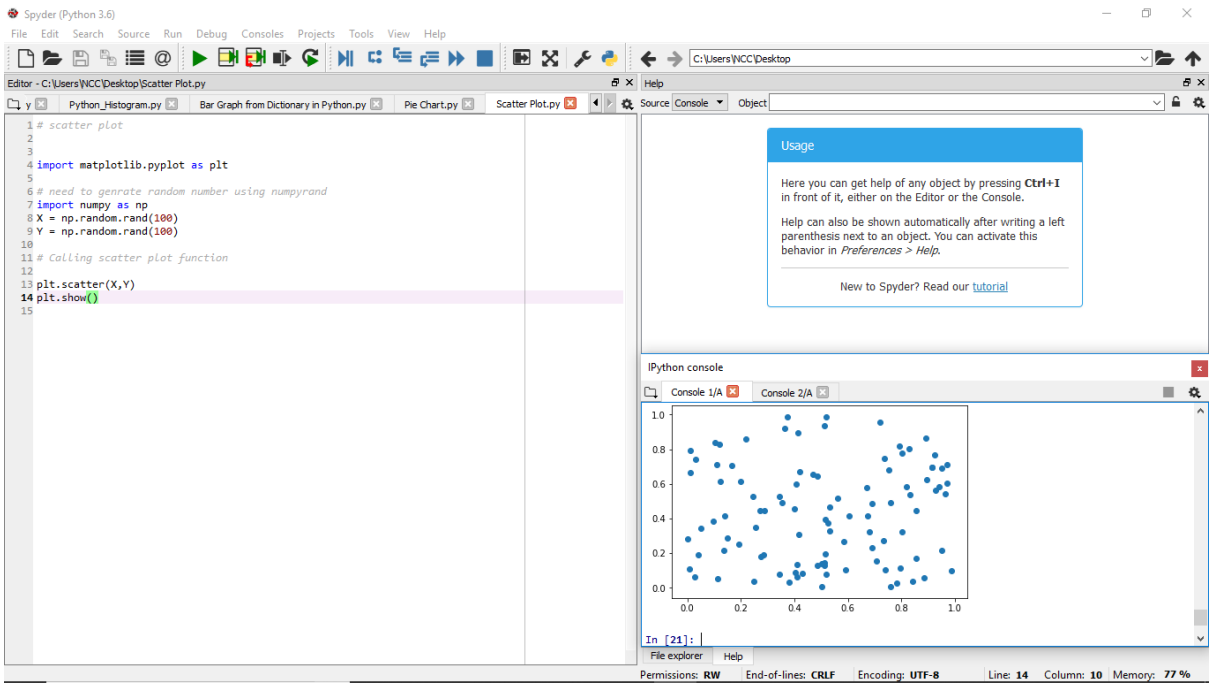


Figure 68 Scatter plot in Python

INTRODUCTION
TO
ANSYS

1.0. Introduction to Ansys Workbench

ANSYS Workbench is a project-management tool that handles the passing of data between ANSYS Geometry / Mesh / Solver / Postprocessing tools. This attribute makes it a great help in project management. A project manager need not worry about the individual files on disk (geometry, mesh etc.). Graphically, you can see at-a-glance how a project has been built. The capability of Workbench in the management of individual applications and passing of data between, makes it easy to automatically perform design studies (parametric analyses) for design optimization.

1.1. Workbench Overview

Main Screen

Figure 1 below presents the main interface of the ANSYS Workbench software. The components of this window are: Top menu – Toolbox – Project schematics – Messages – Progress

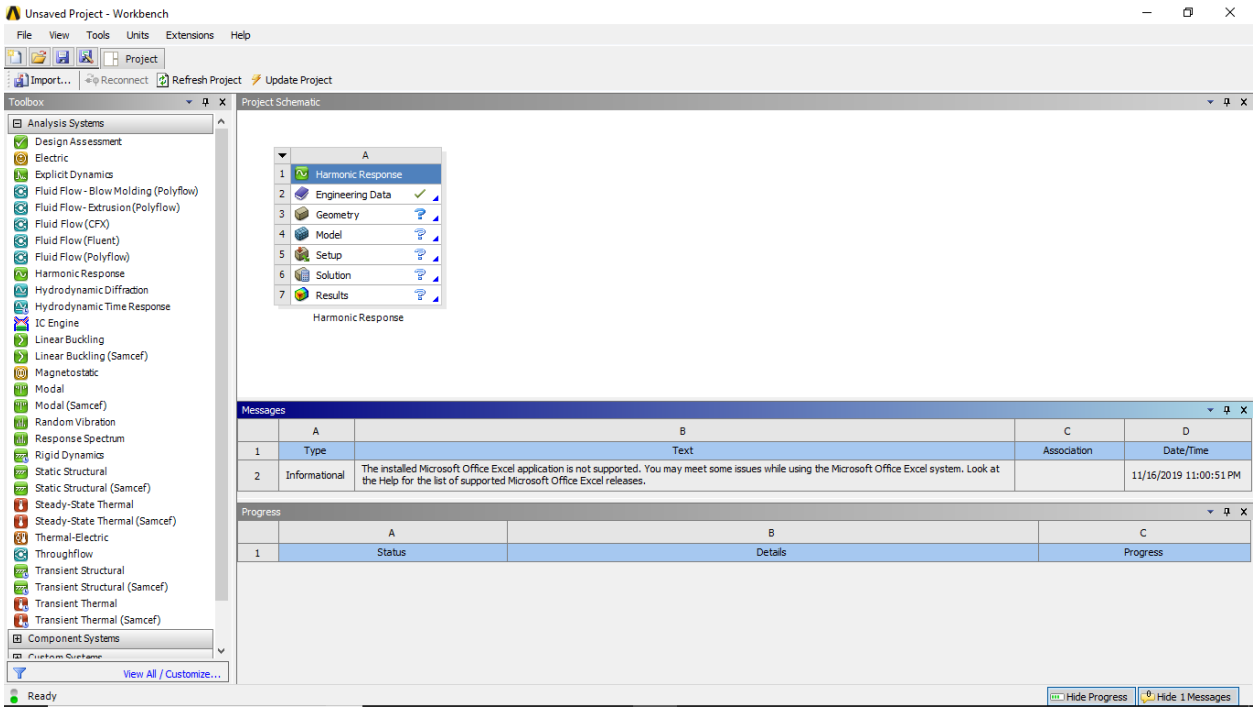
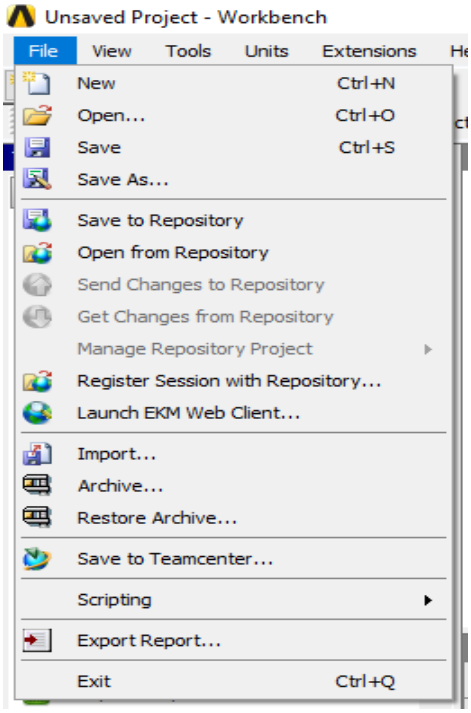


Figure 69 ANSYS Workbench Software Interface

Top Menu

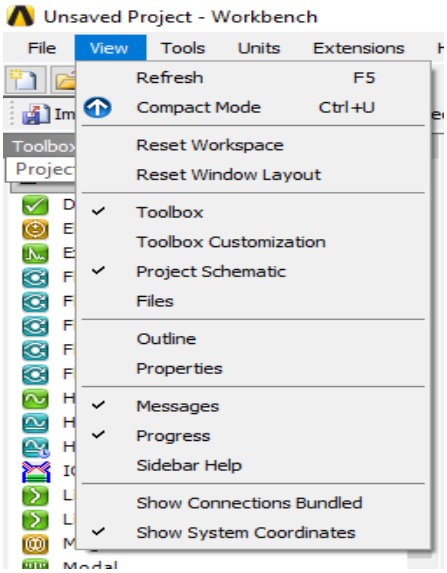
i. Top Menu – File

- Create /Open/Save a Workbench project file
- Saves data from all the components
- Import: Import file (of acceptable file types)
- Archive: Quickly generate a single compressed file (.wbzp or zip file) containing all important files for easy transfer. Zipped file can be saved at any required location
- Restore Archive: Unzip and open zipped project files
- Scripting: Record and Run journal files
- EKM: Launch and connect to EKM (Engineering Knowledge Manager)



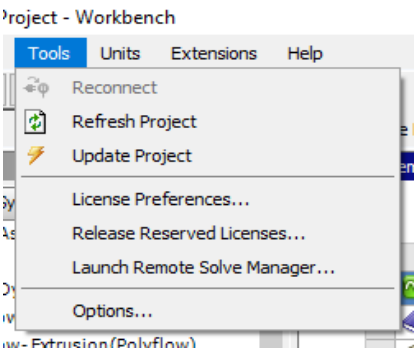
ii. Top Menu – View

- Refresh and Reset Window layout
- View Toolbox, Toolbox Customization, Project Schematic, Message, Progress. Etc. windows on Project page
- Reset Workspace: restores current workspace to default setting
- Reset window layout: restores original window layout



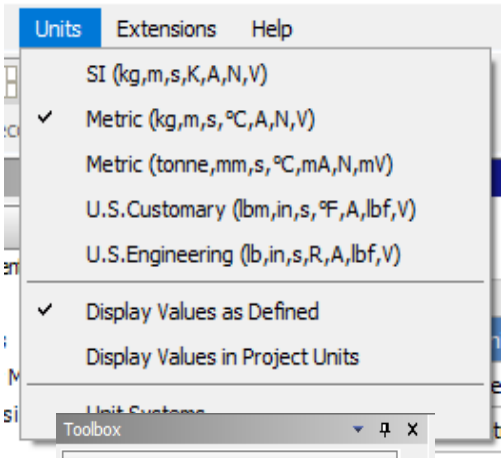
iii. Top Menu – Tools

- Refresh or Update entire project
- Modify Global Setting through options



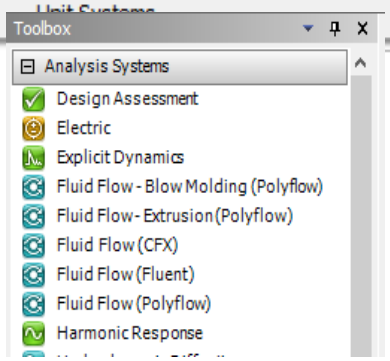
iv. Top Menu – Units

- Select Unit System to be used for the project
- Sets the default unit system for all systems
- Units can be changed within components like DM. AMP
- Display Values as Defined: displays values and unit as defined in Workbench or original source application
- Display Values in Project Unit: displays values converted to selected project unit system
- Modify Unit Systems through Unit System

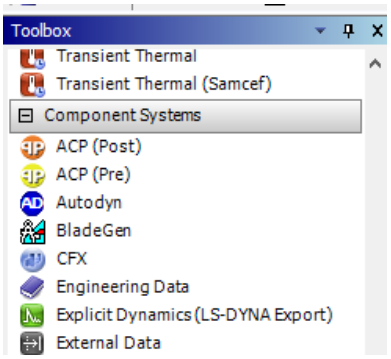


Toolbox

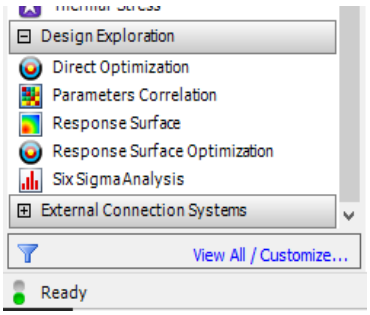
i. Analysis Systems: are ready-made stencils that include all the individual systems (applications) needed for common analyses (for example Geometry + Mesh + Solver + Post-Processor)



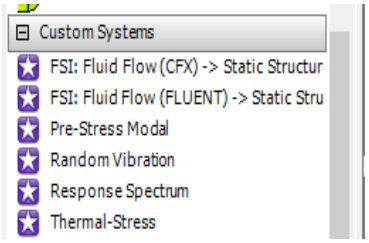
ii. Component Systems: are the individual building-blocks for each stage of the analysis which can be setup by user



iii. Design Exploration: provides tools for optimizing designs and understanding the parametric response

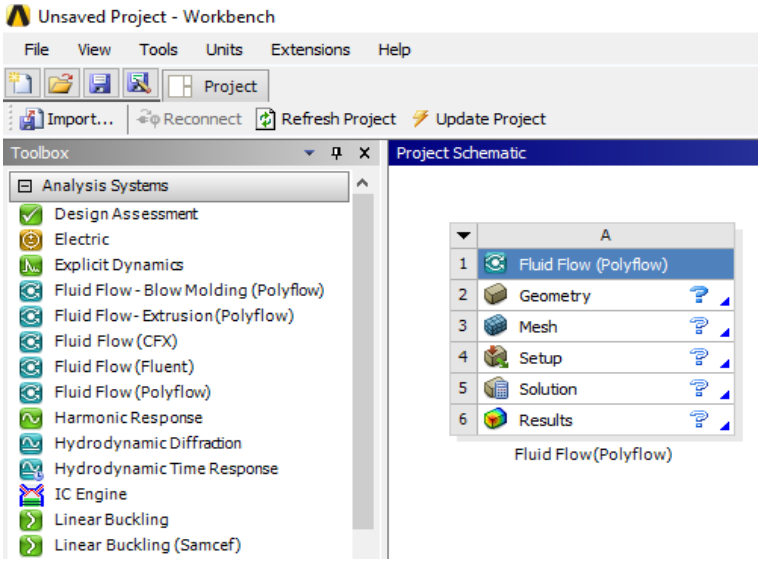


iv. Custom Systems: predefined systems with some interactions between them. It is also possible to define new custom-systems



2.0. Basic Workflow

Starting a new project in the Project Schematic Window requires dragging a corresponding Analysis System onto the Project Schematic Window. A workflow, comprising all the steps needed for a typical analysis. For instance, the Fluid Flow (Polyflow) Analysis System in Fig. 2



Workflow is from top to bottom. As each stage is complete, the icon at the right-hand side

changes

Figure 70 Analysis System in Project Schematic Window

An alternative method to creating a workflow within the Project Schematic Window is by dragging and dropping necessary systems into the Project Schematic Window form the Component Systems and linking the components accordingly with connectors Fig. 3.

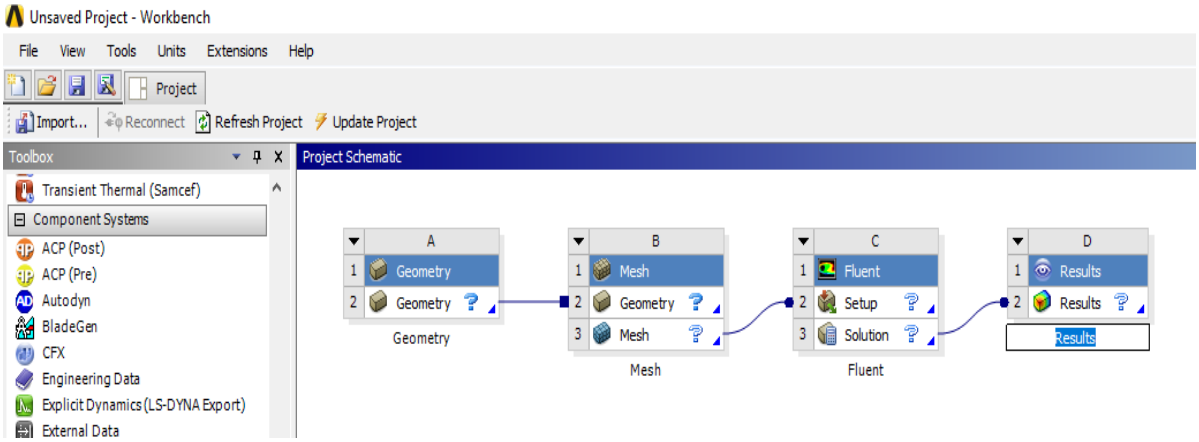


Figure 71 Alternative Method to Workflow Creation

There are two types of connectors: square and round connector. The square connector as seen in Fig. 3 shows that the output of the preceding cell is being shared with the destination cell. A round connector means the output of the preceding cell is being transferred as a setup (input) condition to be used in the destination cell.

2.1. Cell States

Status of each cell in ANSYS Workbench are denoted by distinct icons. Here these icons and their interpretation are presented.

Up to Date

Refresh required. Upstream data has changed

Refresh required. Upstream data has changed

Unfulfilled. Upstream data does not exist

Attention Required

Solving

Update Failed

Update Interrupted

Changes pending (was up-to-date, but upstream data has

changed)

2.2.Sharing Data between Different Solvers

Data can be transfer between solvers in Workbench. In this 1-way FSI (fluidstructure-interaction) example, we transfer the loads from a Fluent CFD simulation over to a Mechanical system to perform a stress analysis.

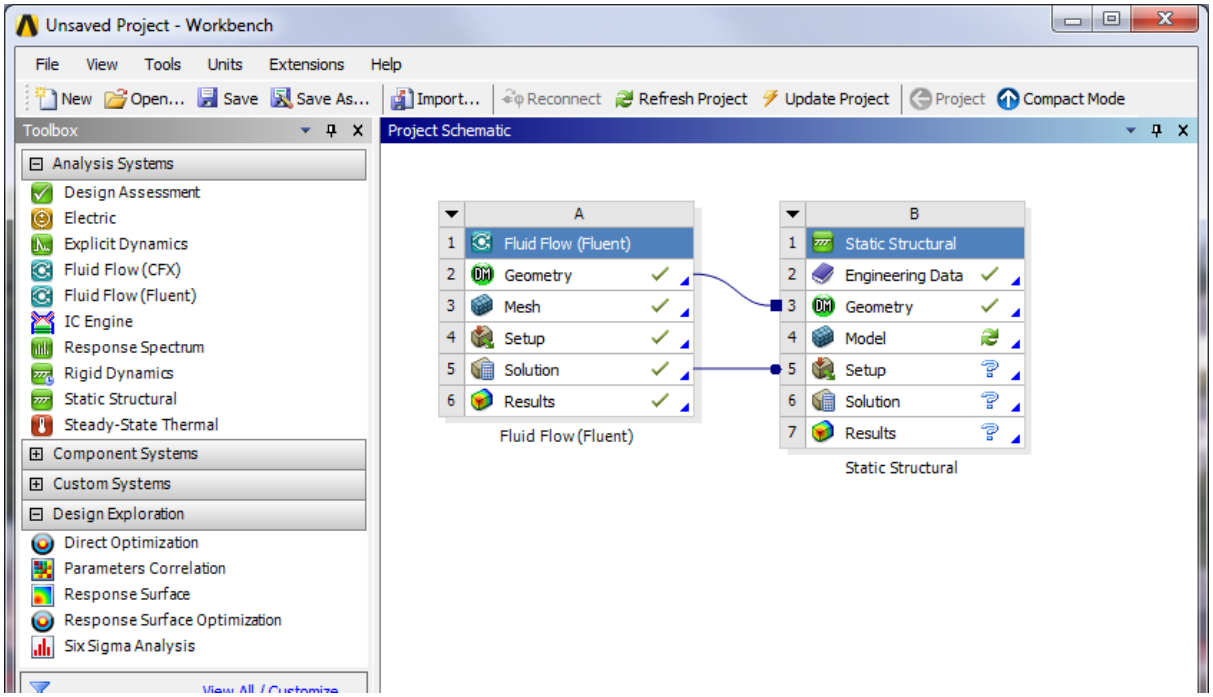


Figure 72 Sharing Data between Different Solvers

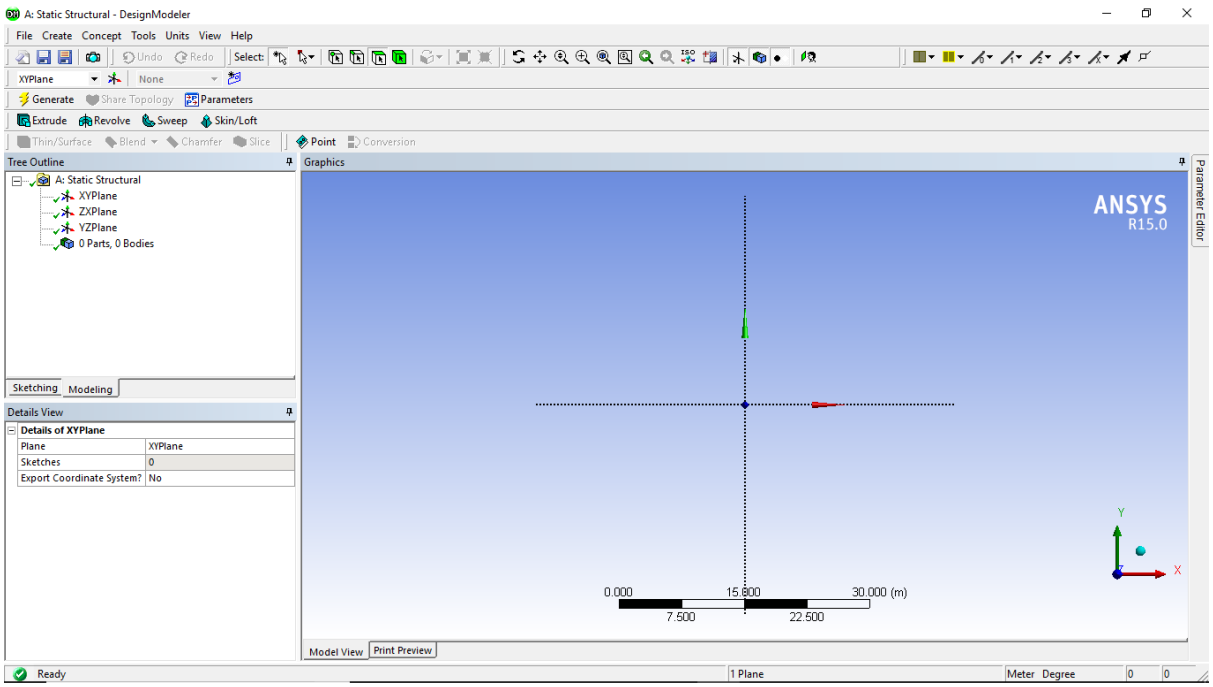


Figure 73 Design Modeler Window

Figure 5 presents the Design Modeler window in ANSYS Workbench consisting of two sub-windows in the Tree Outline: sketching window and modeling window.

In the Sketching window a new model can be created or an imported model can be improved as requisite for the intended analysis.

In the Modeling window the necessary 3D feature is applied to the model and meshing pattern is also added.

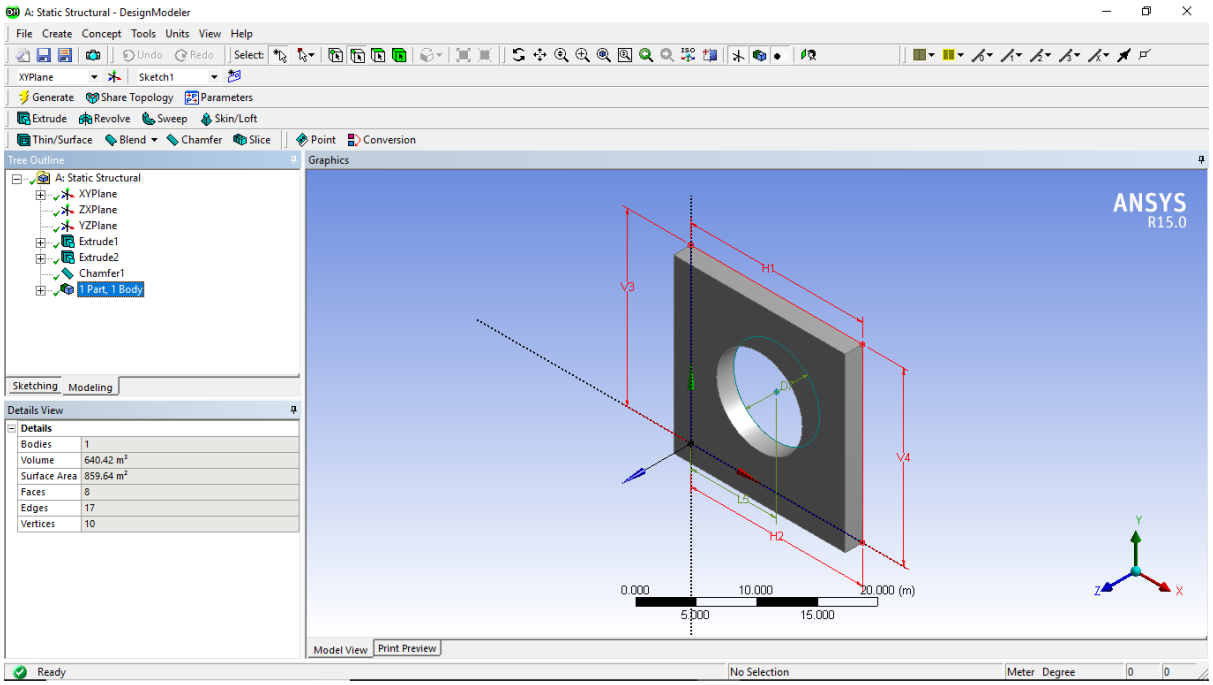


Figure 74 Design Modeler Showing a Created Model

2.3. File Location on the Disk

Should you need to identify the individual files on your disk for each stage of the project, these can be found by enabling View > Files. The resulting table will cross-reference the directory and filename with the project cells.

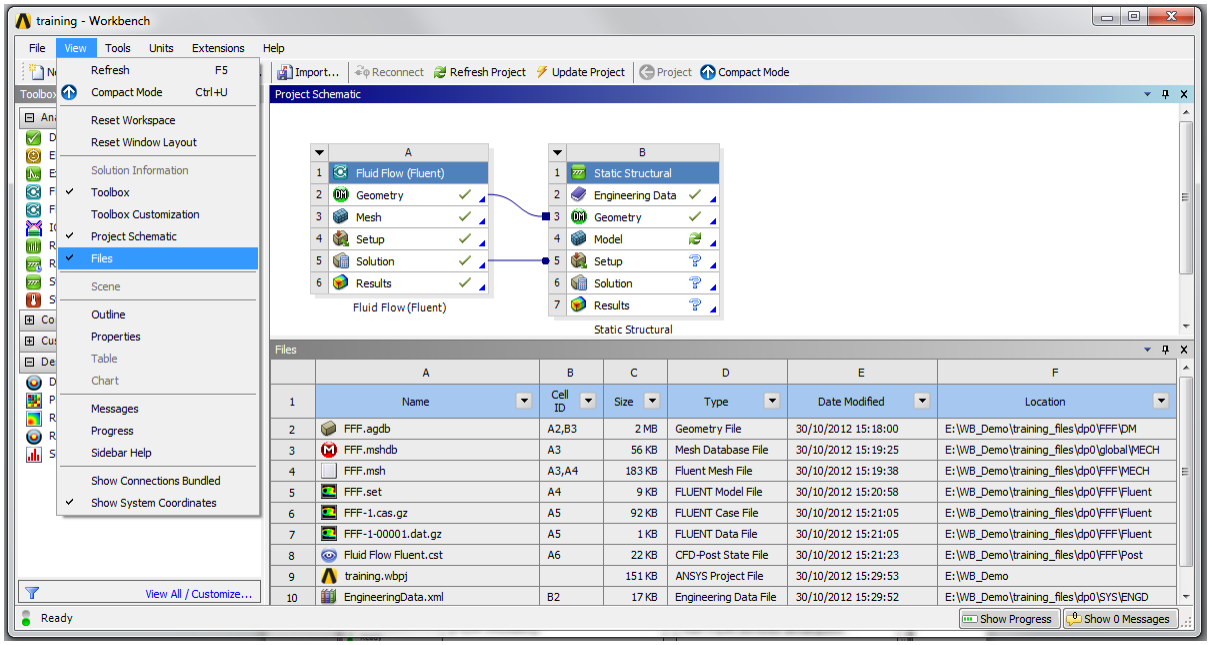


Figure 75 File location Directory

Now we will be running a demo on static structural system, Fluid Flow (Fluent) etc.

WISHING YOU
THE BEST
IN YOUR
EXAMINATION
THANK YOU