Data Sciences using Python (05101305)

Dr. Kamini Solanki, (Associate Professor)
Parul Institute of Computer Application - BCA

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-2

Python Environment Setup and Essential

Preview why Python for Data Science

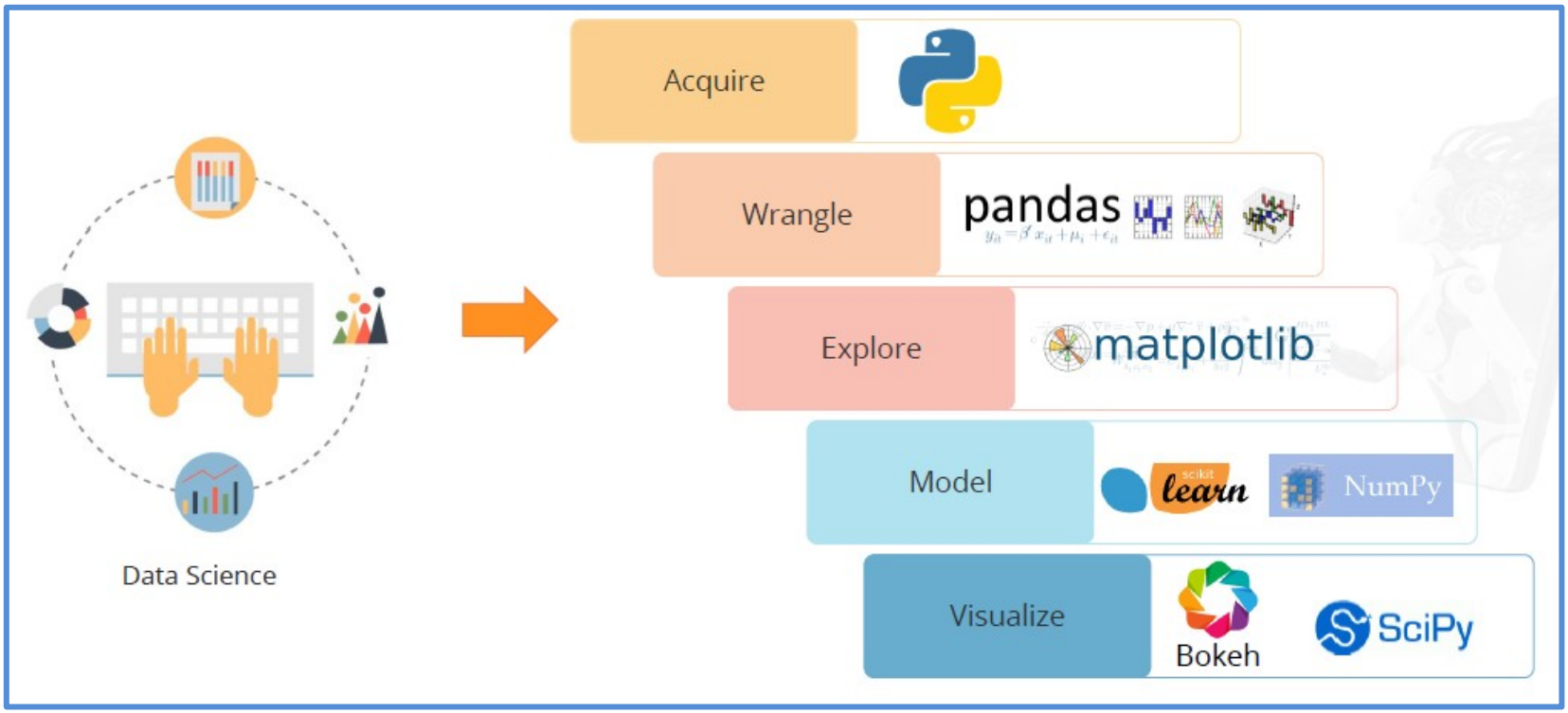


Image Source : Simplilearn

Why Anaconda for Data Science

- Is a complete, open source data science package with a community of over 6 million users.
- Easy to download and supported by all the platform like Windows, MacOs and Linux.
- Anaconda distribution comes with more than 400+ packages.
- So mostly there is no need to install individual library manually.
- Over 20 million user world wide.
- 7000+ packages for machine learning on cloud based repository.
- Anaconda Navigator, a desktop graphical user interface (GUI) system that includes links to all the applications included with the distribution including **RStudio, iPython, Jupyter Notebook, JupyterLab, Spyder, Glue**, and **Orange**.

Why Anaconda is best platform for Data Science

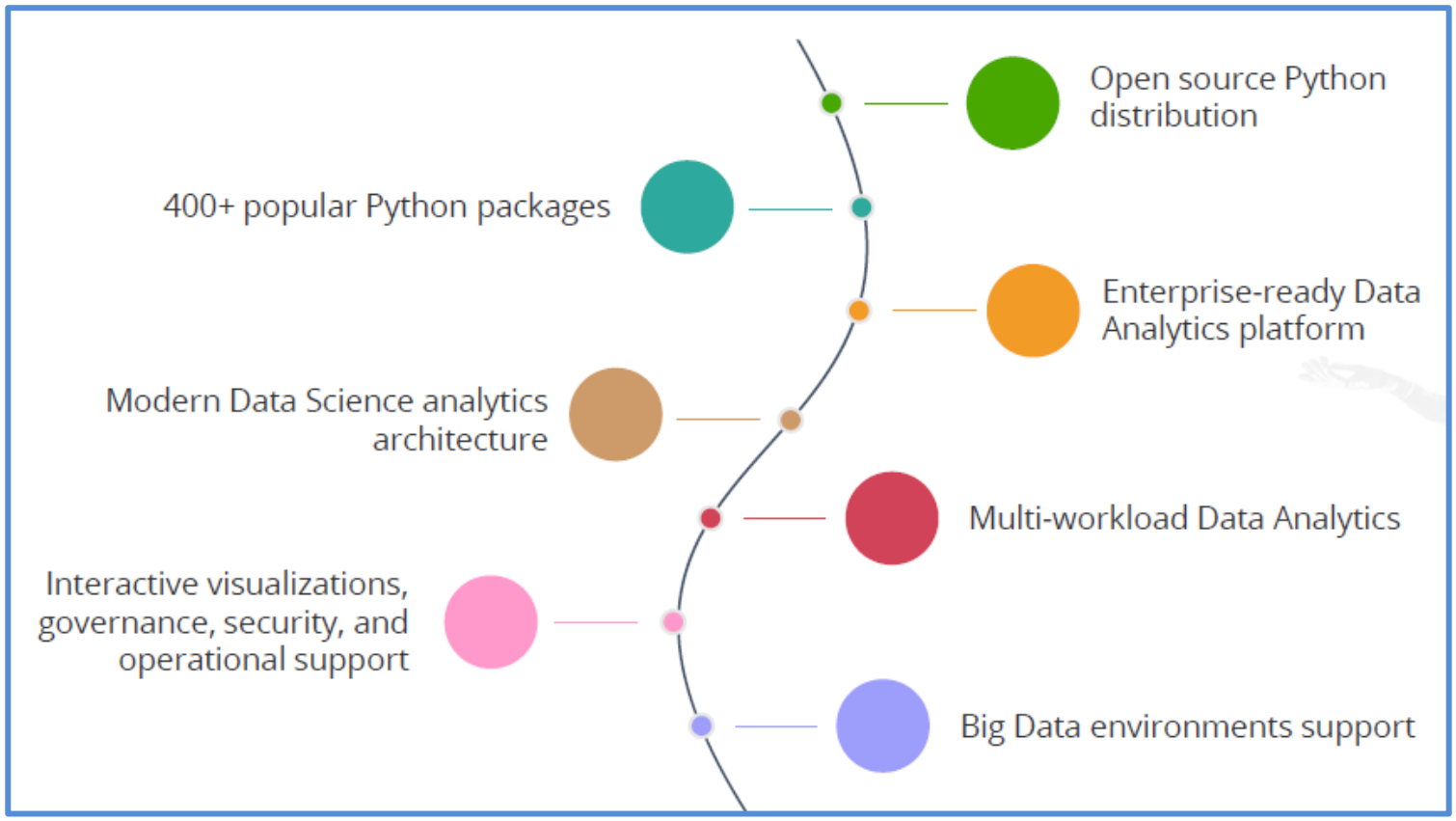


Image Source : Simplilearn

Parul® University

DIGITAL LEARNING CONTENT

Download and installation step for anaconda

- Download anaconda as per your requirement from the given link https://www.anaconda.com/products/individual
- Current version of anaconda support python 3.8 so anaconda support 3.7 and 2.7 version also but I suggest go with latest one.
- After download double click on the installer file (.exe) file.
- Press on next
- Learn and click on “I Agree” for the terms and condition for license.
- Select location and complete all the required steps for the same.

Please refer given video and webpage link for more about installation steps

https://www.youtube.com/watch?v=G3Lt1JWBvL8
https://www.youtube.com/watch?v=5mDYijMfSzs
https://docs.anaconda.com/anaconda/install/

Installation steps for windows.

After downloading and pressing on next, accepting license terms and conditions.

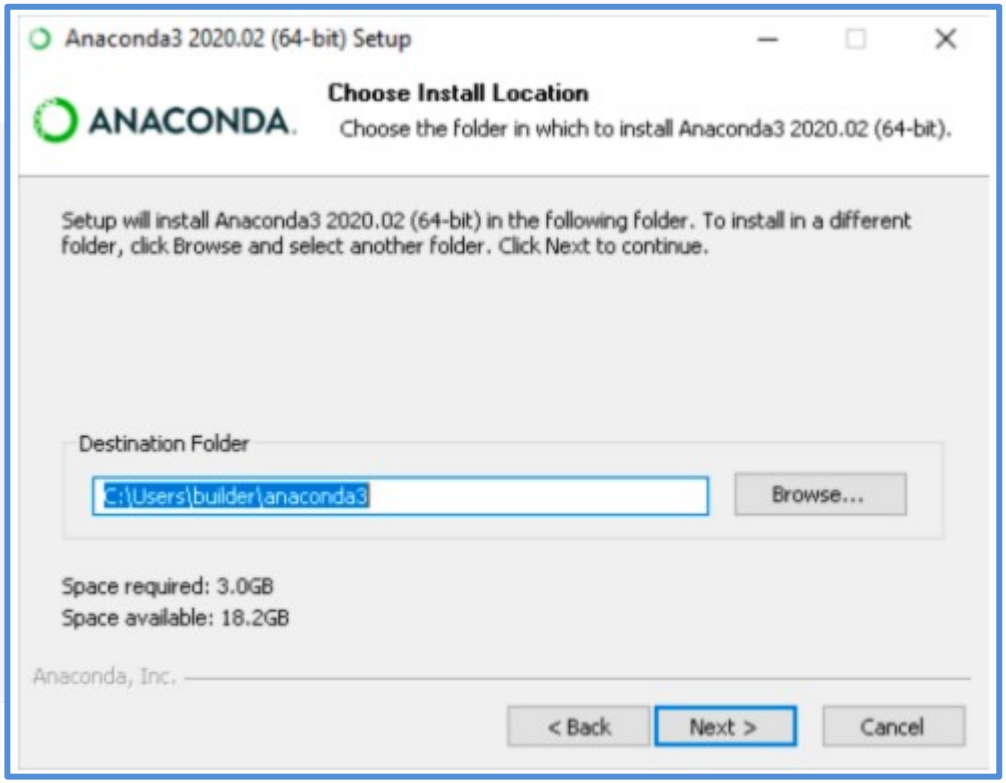
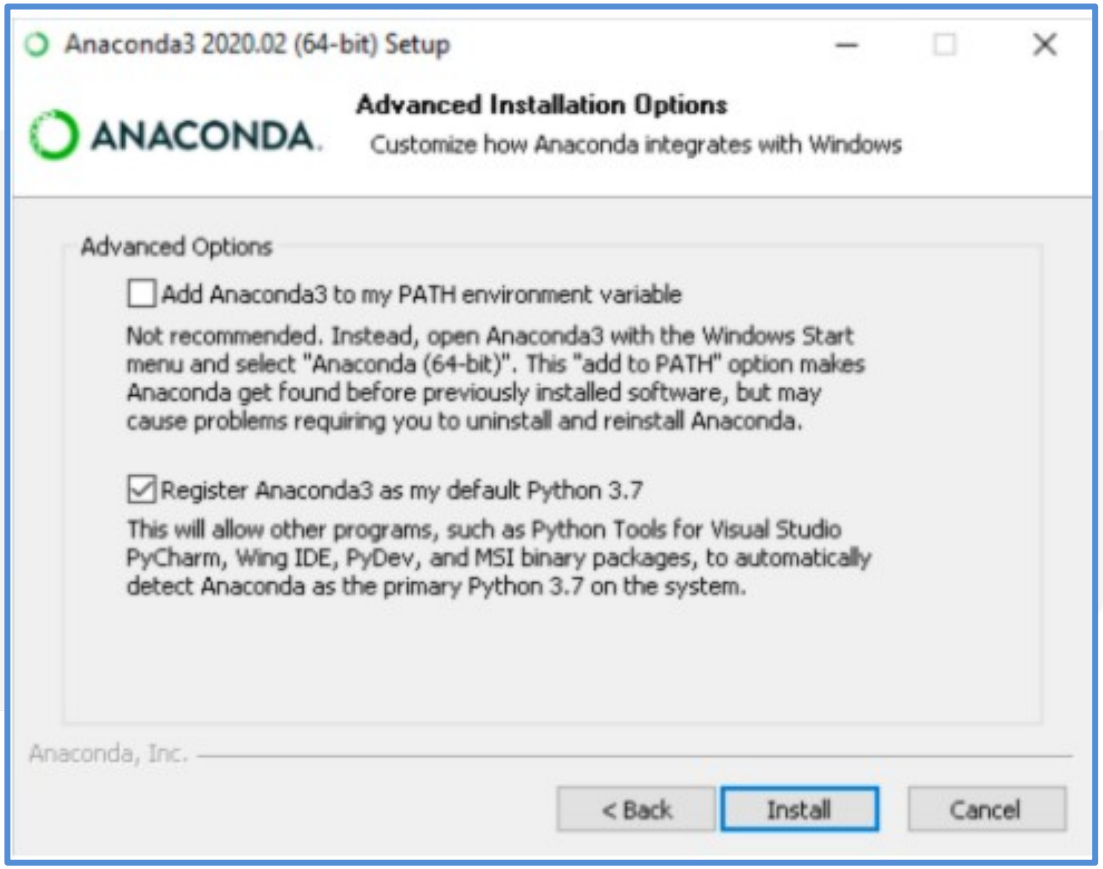
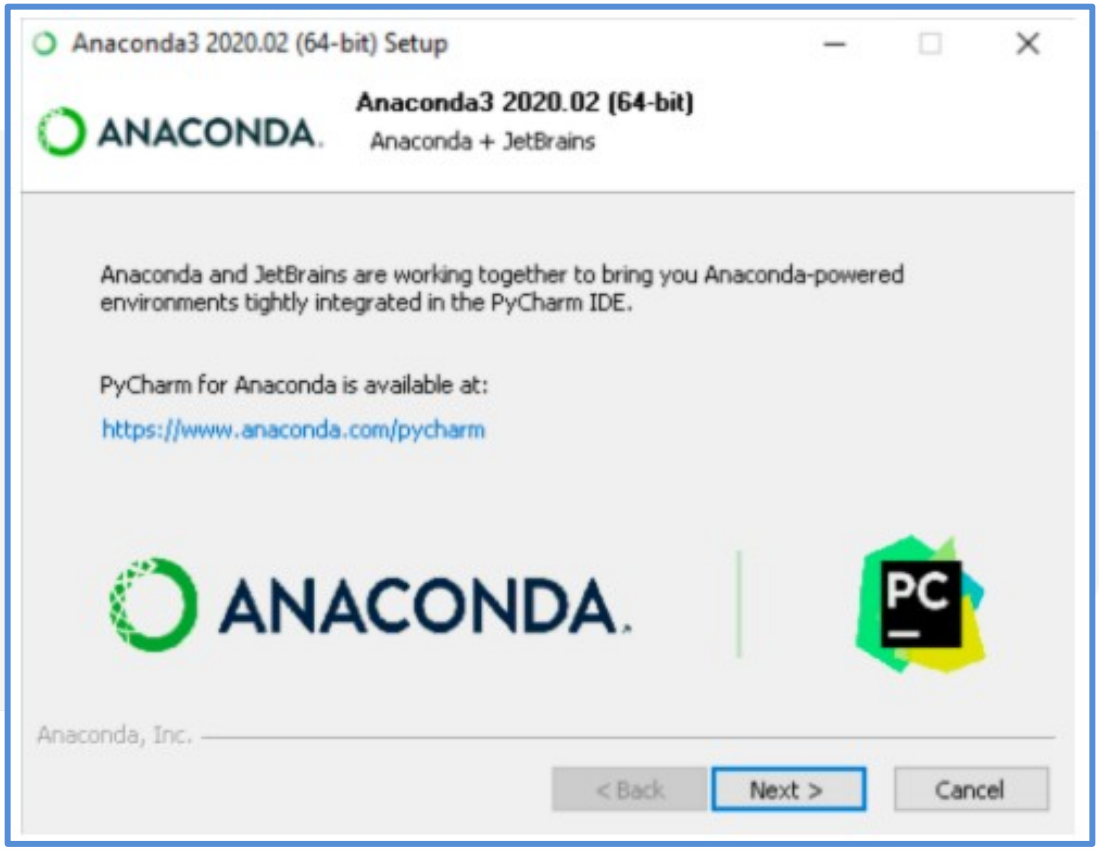


Image Source : https://docs.anaconda.com/

Cont...


Anaconda3 2020.02 (64-bit) Setup
ANACONDA.
Advanced Installation Options
Customize how Anaconda integrates with Windows
Advanced Options
Add Anaconda3 to my PATH environment variable
Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.
Register Anaconda3 as my default Python 3.7
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.
Anaconda, Inc.
< Back
Install
Cancel

Cont...


Anaconda3 2020.02 (64-bit) Setup
ANACONDA.
Anaconda3 2020.02 (64-bit)
Anaconda + JetBrains
Anaconda and JetBrains are working together to bring you Anaconda-powered environments tightly integrated in the PyCharm IDE.
PyCharm for Anaconda is available at:
https://www.anaconda.com/pycharm
ANACONDA.
PC
Anaconda, Inc.
< Back
Next >
Cancel

Cont...


Anaconda3 2020.02 (64-bit) Setup
Completing Anaconda3 2020.02 (64-bit) Setup
Thank you for installing Anaconda Individual Edition.
Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.
Anaconda Individual Edition Tutorial
Learn More About Anaconda
ANACONDA.
< Back
Finish
Cancel

About Jupyter

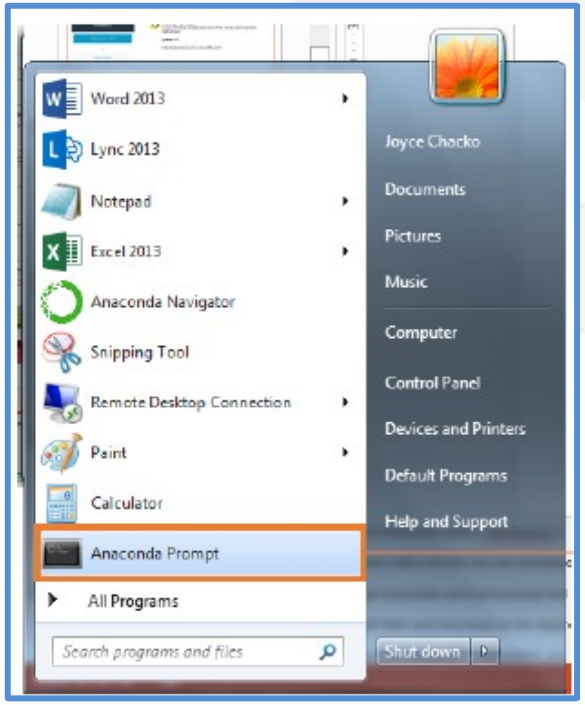
- Jupyter is open source tool which also support python.
- It's web based interactive tool.
- Which provide web based interface for Data Science and Scientific Computing.
- It's client server based app which allow to editing and running document or code via web browser. It also runs on local desktop without internet.
- If you install it on remote server than one can access it through internet in organization.

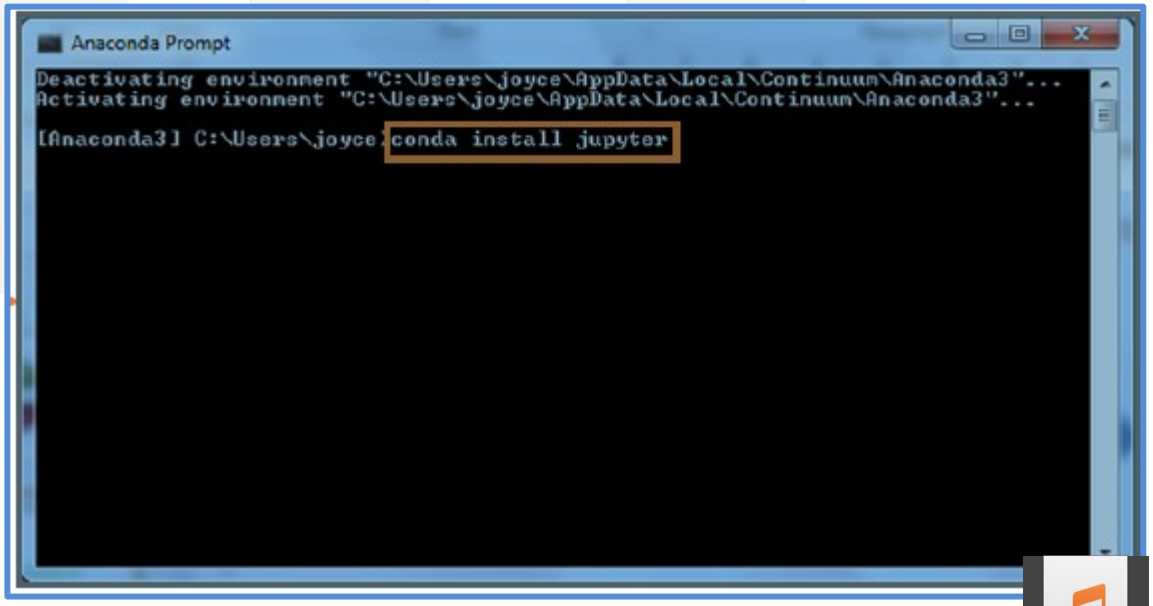
Few advantages of jupyter

Image Source : Simplilearn

Cont...

Once you install anaconda on machine than using anaconda cmd prompt one can install jupyter.

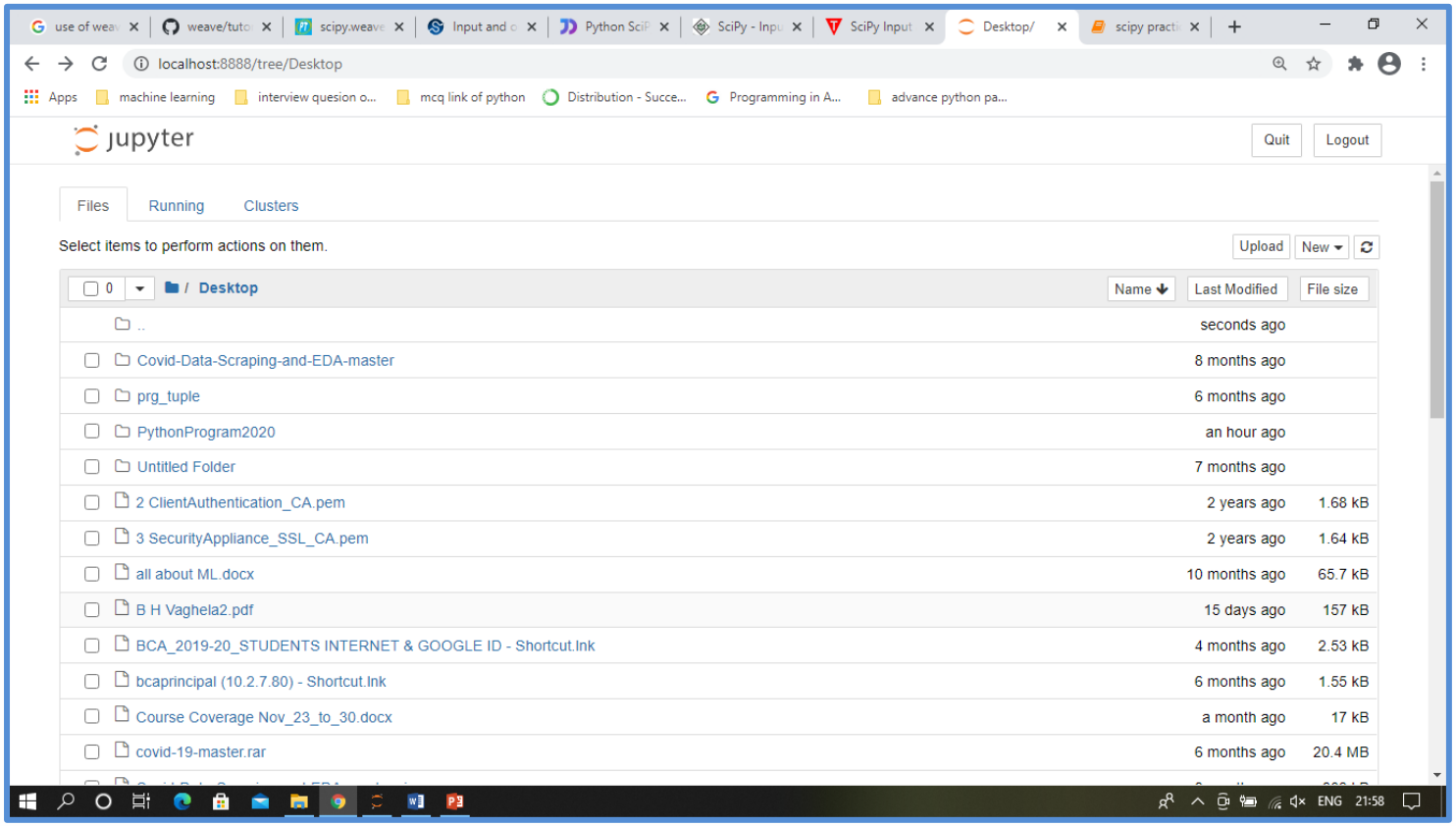




Parul®
University

DIGITAL LEARNING CONTENT

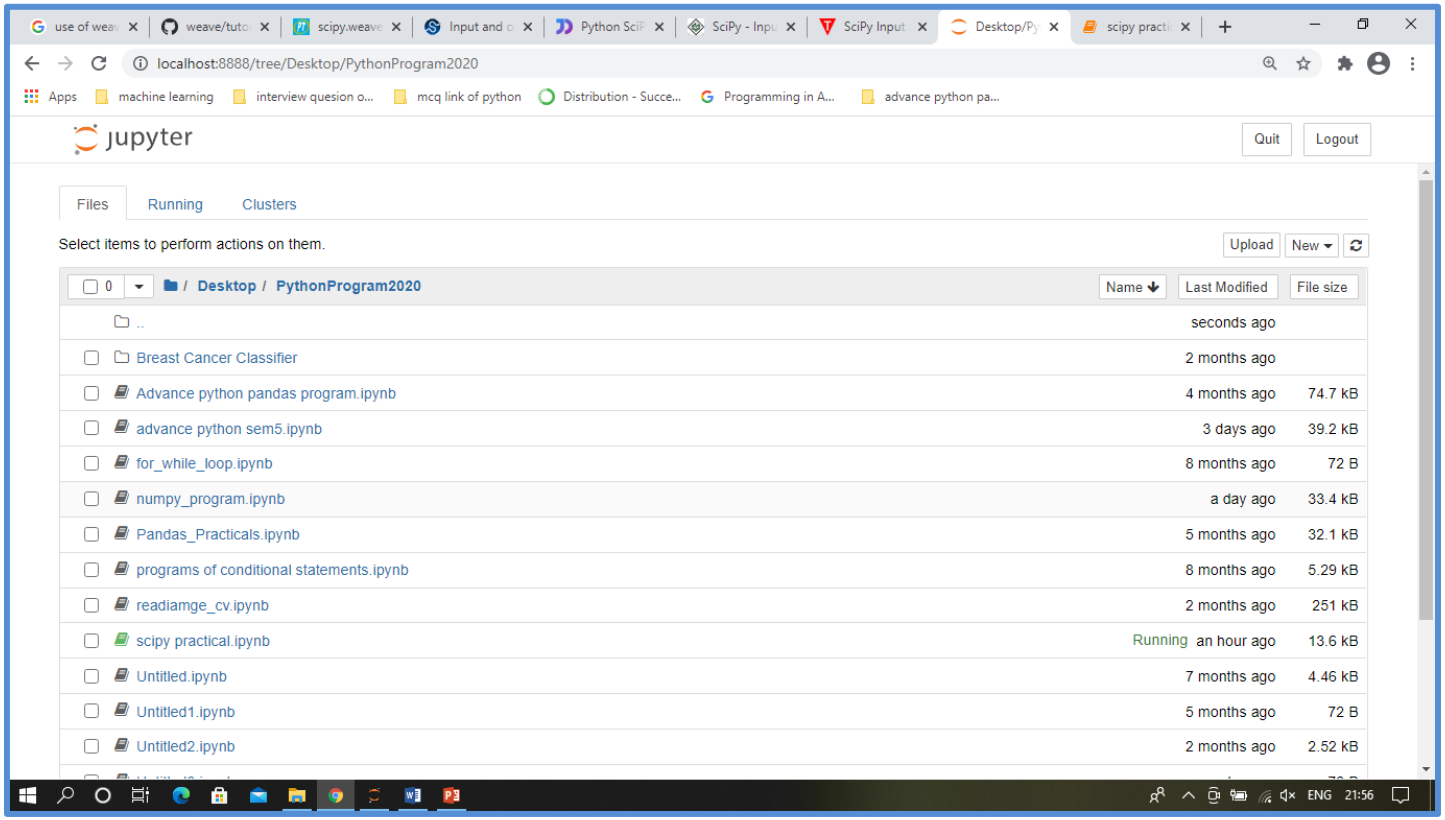
Lunching window of jupyter



Parul®
University

DIGITAL LEARNING CONTENT

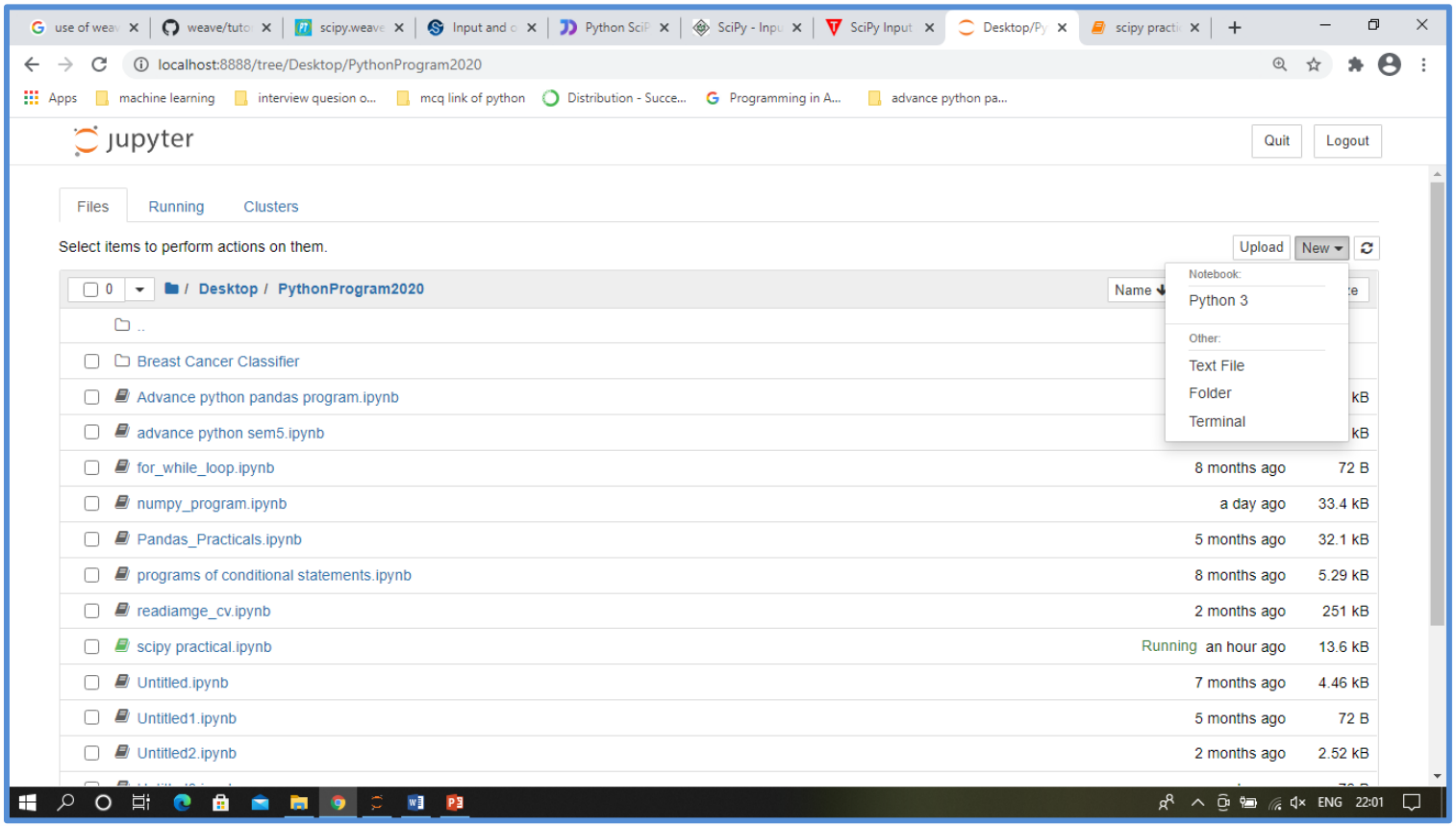
Cont...



Parul®
University

DIGITAL LEARNING CONTENT

Cont...



Cont...

Basic Python

localhost:8888/notebooks/Basic%20Python/Basic%20Python.ipynb

jupyter Basic Python Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

```
In [1]: import sys                                   <- Import sys module
In [2]: print(sys.version)                          <- Print sys version
        2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Feb 16 2016, 09:58:36) [MSC v.1500 64 bit (AMD64)]
In [3]: import platform                             <- Import platform library
In [4]: platform.python_version()                   <- View python version
Out[4]: '2.7.11'
In [5]: # A Hello world example                     <- Comment line
        print('hello world')
        hello world                                 <- Test string
In [6]: 3+5                                         <- Test number operation
Out[6]: 8
In [7]: 8*4
Out[7]: 32
```

Python DataTypes

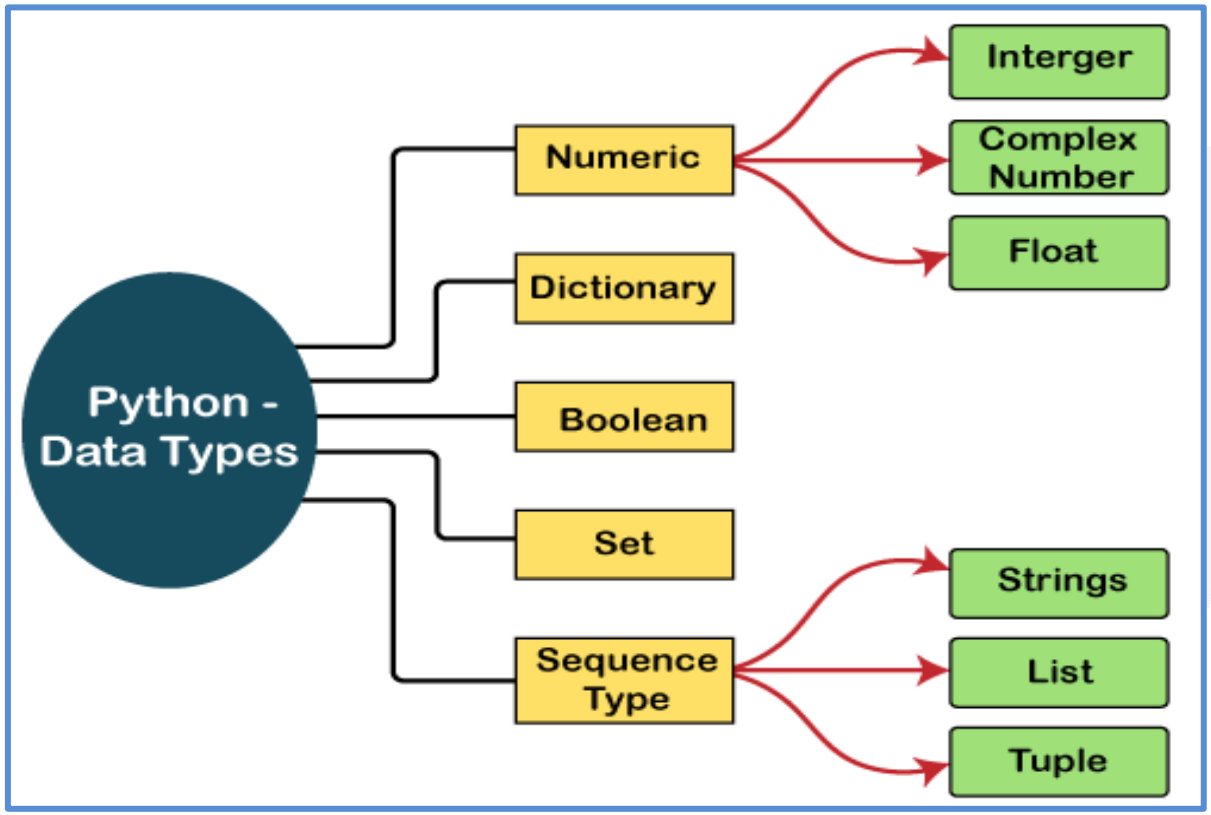


Image Source : Javatpoint

Variable, Value, Expression and Assignment

- A **value** is one of the basic things a program works with, like a letter or a number.
- The values we have seen so far are 1, 2, and 'Hello, World!'.
- These values belong to different types: 2 is an integer, and 'Hello, World!' is a string, 0.4 is a float etc.
- If you are not sure what type a value has, the interpreter can tell you the type of value using type() method.
- When you type a large integer, you might be tempted to use commas between groups of three digits, as in 1,000,000. This is not a legal integer in Python.

```
a=1,000,000
>>> print(a)
(1, 0, 0)
```

Cont...

- Variable is a name which is used to refer memory location. Variable also known as identifier and used to hold value.
- A variable is a name that refers to a value.
- An assignment statement creates new variables and gives them values.

```
>>>number=105467
>>>name="John"
```

- If you type an integer with a leading zero, you might get a confusing error:

```
>>> zipcode = 02492
```

Syntax Error: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers

Cont...

- **An expression** is a combination of values, variables, and operators.
- Is evaluate down some result after the execution in single value either decimal, float or string.
- Lets see some of the example of expression in python.

```
num1 + num2
>>>print(“hello”+”world”)
a*b+(c-d)/8
print(45.90+”hello”)
```

Rules to Declare Variable

- It can be group of letters and number.
- It should not start with number.
- Use underscore (_) as a special character to join two word like student_name, first_name like.
- Identifier/ Variable name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Variable/ Identifier name must not be any reserved keyword of python.
- As python is case sensitive language so variable name “myname” and “Myname” is differ.
- Some valid variable name as below.

- **e.g _name_, _myname123, my_name, hello123, _123name, __in__ all are valid**

Cont...

The variable refers to the memory location of the assigned value.

```
In [1]: x = 3
        type(x)

Out[1]: int
```

The variable appears on the left, while the value appears on the right.

```
In [2]: y = 2.1
        type(y)

Out[2]: float
```

The data type of the assigned value and the variable is the same.

```
In [3]: z = 'test'
        type(z)

Out[3]: str
```

Image Source : Simplilearn

Variable and assignment cont...

```
In [44]: first_string_variable = 'test'
         first_integer_variable = 123
```

Assignment

```
In [45]: print (first_string_variable)
         print (first_integer_variable)
```

```
test
123
```

Variable data value

```
In [47]: print type (first_string_variable)
         print type (first_integer_variable)
```

```
<type 'str'>
<type 'int'>
```

Data type of the object

Multiple Assignment

```
In [48]: number_example
```

Access variable without assignment

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-48-a856f233ae98> in <module>()
----> 1 number_example

NameError: name 'number_example' is not defined
```

```
In [49]: number_example = 2
         number_example
Out[49]: 2
```

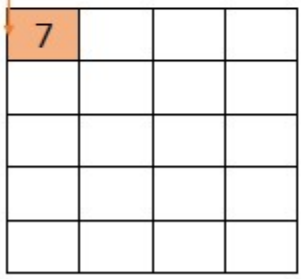
Access variable after assignment

```
In [54]: integer_x, integer_y = 5,22
In [55]: integer_x
Out[55]: 5
In [56]: integer_y
Out[56]: 22
```

Multiple assignments

Assignment and Reference

```
x = 7
```



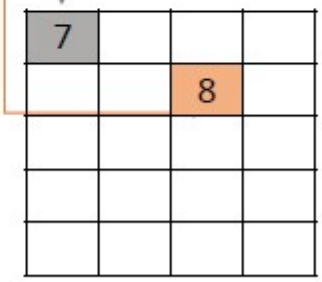
Memory location

Ref: <address 12>

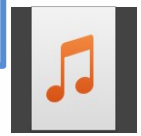
```
x = 7
x = x + 1

print(x)
```

```
8
```

Garbage collected



Memory location



Basic DataType – (int, float and string)

```
In [1]: #Basic Data Types Int, float and string
        num = 100
        print(num)
        num2 = 10.20
        print(num2)

        100
        10.2

In [4]: type(num)
Out[4]: int

In [5]: type(num2)
Out[5]: float

In [6]: type(num + num2)
Out[6]: float

In [7]: num+num2
Out[7]: 110.2
```

- Here num and num2 are variable which assign integer and float value simultaneously.
- You can also check data type of variable or expression by using type() method.
- Here in the given example you can see num is of type integer and num2 is of type float.
- Addition of num and num2 is of type float.

Cont...

In [8]:
```
#in python one can decalcre string using  "" (double quote)
#using '' (single quote)
#using """""" (triple double quote) or '''''' (triple single quote)
str1 = 'hello world'
str2 = "hello world"
str3 = '''hello
                world'''
str4 = """Good
           Morning"""
```

In [9]:
```
print(str1)
print(str2)
print(str3)
print(str4)
```

```
hello world
hello world
hello
                world
Good
           Morning
```

None and Boolean Value

```
In [10]: var1 = None
         var1 is None

Out[10]: True

In [11]: var1 = 10
         var1 is None

Out[11]: False

In [16]: var1 is int

Out[16]: False
```

Assign value none

Check data is none or not and gives Boolean value as True or False

Parul®
University

DIGITAL LEARNING CONTENT

Type Casting

```
In [17]: #Type Casting
         num1 = 12
         float(num1)

Out[17]: 12.0

In [18]: num2 = 134.23
         int(num2)

Out[18]: 134

In [19]: num3 = 14
         str(num3)

Out[19]: '14'
```

Type Casting mean to convert data into other data type using int(), float() or str() function.

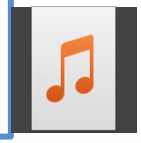
Parul®
University

DIGITAL LEARNING CONTENT

Data Structure - Tuple

```
In [21]: #creating tuple
         data = (10,'hello',23.50,100,'xyz',"good")
         data1 = tuple()  #creating empty tuple
         #as tuple is immutable so there is no meaning to create empty tuple
         print(data1)

         ()

In [22]: print(data)

         (10, 'hello', 23.5, 100, 'xyz', 'good')

In [23]: data[2]

Out[23]: 23.5

In [24]: data[4] = 'hi' #try to modify tuple

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-24-a1e9a3700142> in <module>
         ----> 1 data[4] = 'hi' #try to modify tuple

         TypeError: 'tuple' object does not support item assignment
```

It gives an error because tuple is immutable



Accessing Tuple elements using index value

```
In [7]: #access tuple element by passing index value
        #one can pass positive value as well as negative value as an index
        #positive index value start by 0 (left to right)
        #negative index value start by -1 (right to left)
        data = (10, 'hello', 23.5, 100, 'xyz', 'good',45.23,'USA')
        data[2]

Out[7]: 23.5

In [8]: data[-3]

Out[8]: 'good'
```

- You can also access tuple element using index value. Index value must be positive or negative.
  Negative index value start with -1 from right to left. And positive index value start with 0 from left to right.

Parul®
University

DIGITAL LEARNING CONTENT

Slicing in Tuple

```
In [1]: first_tuple = (12, 'Jack', 45.6, 'new', (3,2), 'test')
```

Tuple

```
In [4]: #Creating a subset/slice of the tuple
        #Specify the indices of the elements, separated by a colon
        #The first index is inclusive; the second index is exclusive
        first_tuple[1:4]
```

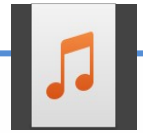
```
Out[4]: ('Jack', 45.6, 'new')
```

Count starts with the first index,
but stops before the second index

```
In [5]: #You can use negative indices as well to slice a tuple
        #Count from the right, starting from -1, to specify the correct index
        first_tuple[1: -1]
```

```
Out[5]: ('Jack', 45.6, 'new', (3, 2))
```

Even for negative indices, the count
stops before the second index



Parul® University

DIGITAL LEARNING CONTENT

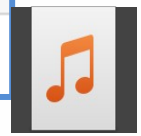
Data Structure - List

```
In [9]: #list is collection of different data in sequenctial order
        #one can also create empty list using list() function
        #list is mutable data structure so one can update it later on
        mylist = [10,200,45.3,"hello","good morning",78.34,'a',"welcome"]
        mylist

Out[9]: [10, 200, 45.3, 'hello', 'good morning', 78.34, 'a', 'welcome']

In [10]: #Accessing element by passing index value
         mylist[4]

Out[10]: 'good morning'

In [11]: mylist[-2]

Out[11]: 'a'

In [12]: #append elemnt at last using append() method of list
         mylist.append(100)
         mylist

Out[12]: [10, 200, 45.3, 'hello', 'good morning', 78.34, 'a', 'welcome', 100]
```

- List is one dimensional order sequence. It's a mutable so one can change its data in future also by passing index value.
- List is collection of mix data type value so we can also call it heterogeneo

Slicing and other list operation

```
In [50]: #removing element using pop() method and remove() method
         #in pop()method have to pass index value
         #in remove() method pass which elements value wants to remove
         mylist.remove('hello')
         mylist

Out[50]: [10, 200, 45.3, 'good morning', 78.34, 'a', 'welcome', 100]

In [51]: mylist.remove(200)
         mylist

Out[51]: [10, 45.3, 'good morning', 78.34, 'a', 'welcome', 100]

In [52]: #for removing element using pop() have to pass index value
         mylist.pop(2)

Out[52]: 'good morning'

In [53]: mylist

Out[53]: [10, 45.3, 78.34, 'a', 'welcome', 100]
```



Cont...

In [81]:
```
#one can insert element in between the list also
#for that insert() method is use
#have to pass index value (location where wants to insert new element)
mylist.insert(3,"hello")
print(mylist)
```

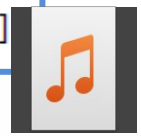
```
[10, 45.3, 78.34, 'hello', 'a', 'welcome', 100]
```

In [82]:
```
#one can also insert list using insert method
mylist.insert(1,[12,23,67,100])
print(mylist)
```

```
[10, [12, 23, 67, 100], 45.3, 78.34, 'hello', 'a', 'welcome', 100]
```

In [83]:
```
#one can also extend list using one more list
mylist.extend(['a','b',100])
print(mylist)
```

```
[10, [12, 23, 67, 100], 45.3, 78.34, 'hello', 'a', 'welcome', 100, 'a', 'b', 100]
```



Cont...

```
In [85]: print(mylist)

         [10, [12, 23, 67, 100], 45.3, 78.34, 'hello', 'a', 'welcome', 100, 'a', 'b', 100]

In [87]: #slicing operation in list
         #for slicing we have to pass starting index and stoping (end) index value
         #it will not going to cosider end index value
         mylist[2:6]

Out[87]: [45.3, 78.34, 'hello', 'a']

In [88]: mylist[2:-2]

Out[88]: [45.3, 78.34, 'hello', 'a', 'welcome', 100, 'a']

In [89]: mylist[4:-1]

Out[89]: ['hello', 'a', 'welcome', 100, 'a', 'b']
```

Data Structure - Dictionary

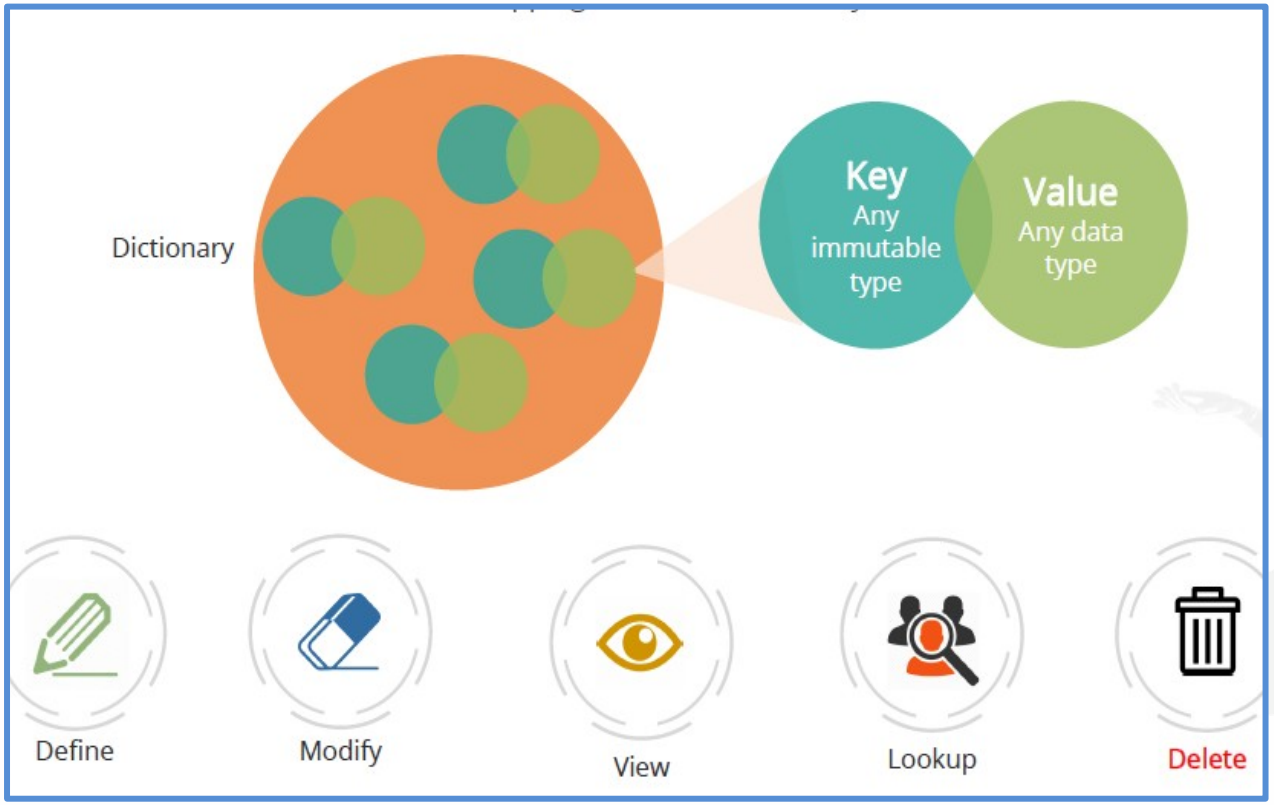


Image Source : Simplilearn

- Dictionary is a collection of pair data in key and its value.
- Key is immutable but value is mutable

Cont...

```
In [90]: #creating dictionay
         #one can also view dictionary, key or only value
         #its enclose withing {} bracket
         #one can also declare empty dict using dict() function
         dict1 = dict()
         print(dict1)

         {}

In [91]: #dictionary is combination of key : value
         #so data of dictionary always in pair
         #as a key one can declare any number or string or alphabats
         dict2 = {'A' : 65, 'B' : 66, 'hello' : 'hi',1:'one',2:'two'}
         dict2

Out[91]: {'A': 65, 'B': 66, 'hello': 'hi', 1: 'one', 2: 'two'}

In [92]: dict2.keys()

Out[92]: dict_keys(['A', 'B', 'hello', 1, 2])

In [93]: dict2.values()

Out[93]: dict_values([65, 66, 'hi', 'one', 'two'])
```

Parul®
University

Modify Dictionary

```
In [94]: #in dictionay data is modify by passing its key value
         #if key is not present than it will going add it.
         #if key is not present than it not fire an error
         dict2

Out[94]: {'A': 65, 'B': 66, 'hello': 'hi', 1: 'one', 2: 'two'}

In [96]: dict2['hello'] = 'good morning'
         dict2

Out[96]: {'A': 65, 'B': 66, 'hello': 'good morning', 1: 'one', 2: 'two'}

In [98]: dict2.update({'B' : 70})
         dict2

Out[98]: {'A': 65, 'B': 70, 'hello': 'good morning', 1: 'one', 2: 'two'}

In [99]: dict2[3] = 'three'
         dict2

Out[99]: {'A': 65, 'B': 70, 'hello': 'good morning', 1: 'one', 2: 'two', 3: 'three'}
```

Data Structure - Set

```
In [103]: #set is collection of unique data
          #using set() function one can create empty set
          #element of set is enclose withing {} braches
          set1 =  set() #empty set
          type(set1)

Out[103]: set

In [104]: set2 = {} #empty set
          type(set2)

Out[104]: dict

In [109]: set1 = {1,4,1,2,49,2,'hello','hi',100,'Hello'}
          set1 #duplicate element is going to removed

Out[109]: {1, 100, 2, 4, 49, 'Hello', 'hello', 'hi'}
```

Set Operation – Union Operation

```
In [109]: set1 = {1,4,1,2,49,2,'hello','hi',100,'Hello'}
          set1 #duplicate element is going to removed

Out[109]: {1, 100, 2, 4, 49, 'Hello', 'hello', 'hi'}

In [110]: set2 = {5,4,20,10,'Good',"Morning",100}
          set2

Out[110]: {10, 100, 20, 4, 5, 'Good', 'Morning'}

In [111]: #union operation perform using | or union() method
          set1 | set2

Out[111]: {1, 10, 100, 2, 20, 4, 49, 5, 'Good', 'Hello', 'Morning', 'hello', 'hi'}

In [112]: set1.union(set2)

Out[112]: {1, 10, 100, 2, 20, 4, 49, 5, 'Good', 'Hello', 'Morning', 'hello', 'hi'}
```

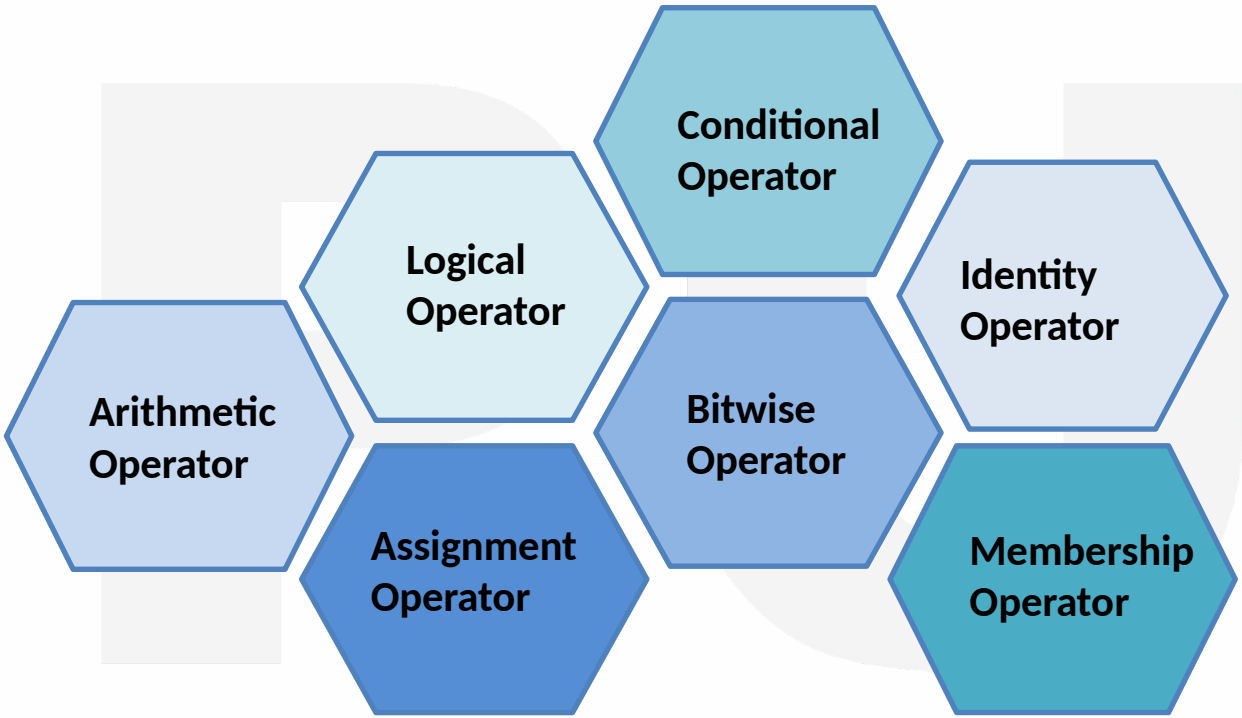
Set operation - intersection

```
In [113]: set1
Out[113]: {1, 100, 2, 4, 49, 'Hello', 'hello', 'hi'}

In [114]: set2
Out[114]: {10, 100, 20, 4, 5, 'Good', 'Morning'}

In [115]: #intersection operation
          #it gives common element from two sets
          #intersection operation perform using & and intersection() method
          set1 & set2
Out[115]: {4, 100}

In [116]: set1.intersection(set2)
Out[116]: {4, 100}

In [117]: set2.intersection(set1)
Out[117]: {4, 100}
```

Parul® University

DIGITAL LEARNING CONTENT

Operators in Python



Membership operator in and not in

```
In [121]: print(mylist)
          [10, [12, 23, 67, 100], 45.3, 78.34, 'hello', 'a', 'welcome', 100, 'a', 'b', 100]

In [122]: 100 in mylist   #check and gives boolean value True or False
Out[122]: True

In [123]: str1 = "wel come to Parul University"
          'come' in str1
Out[123]: True

In [124]: 'P' in str1
Out[124]: True

In [125]: 'Z' in str1
Out[125]: False

In [126]: 'Z' not in str1
Out[126]: True
```

Basic Operator +

```
In [127]: #basic operator + is use to perform the concatenation operation
          #on tuple, string, list other data structure
          str1 = 'hello'
          str2 = 'world'
          str1 + ' ' + str2

Out[127]: 'hello world'

In [129]: list1 = [1,2,7]
          list2 = ['a','b',100,'hello']
          list1 + list2

Out[129]: [1, 2, 7, 'a', 'b', 100, 'hello']

In [130]: tuple1 = (3,7,'hello')
          tuple2 = (10.34,'morning')
          tuple1 + tuple2

Out[130]: (3, 7, 'hello', 10.34, 'morning')
```

- But it is not possible for different data structure

Basic Operator *

```
In [133]: #basic operator *
          #this operator not muliply the data its use for repeat the element
          tuple1

Out[133]: (3, 7, 'hello')

In [134]: tuple1 * 3 #repeat 3 times

Out[134]: (3, 7, 'hello', 3, 7, 'hello', 3, 7, 'hello')

In [135]: list1 * 2

Out[135]: [1, 2, 7, 1, 2, 7]

In [136]: str1 * 5

Out[136]: 'hellohellohellohellohello'
```

Function

Why Function?

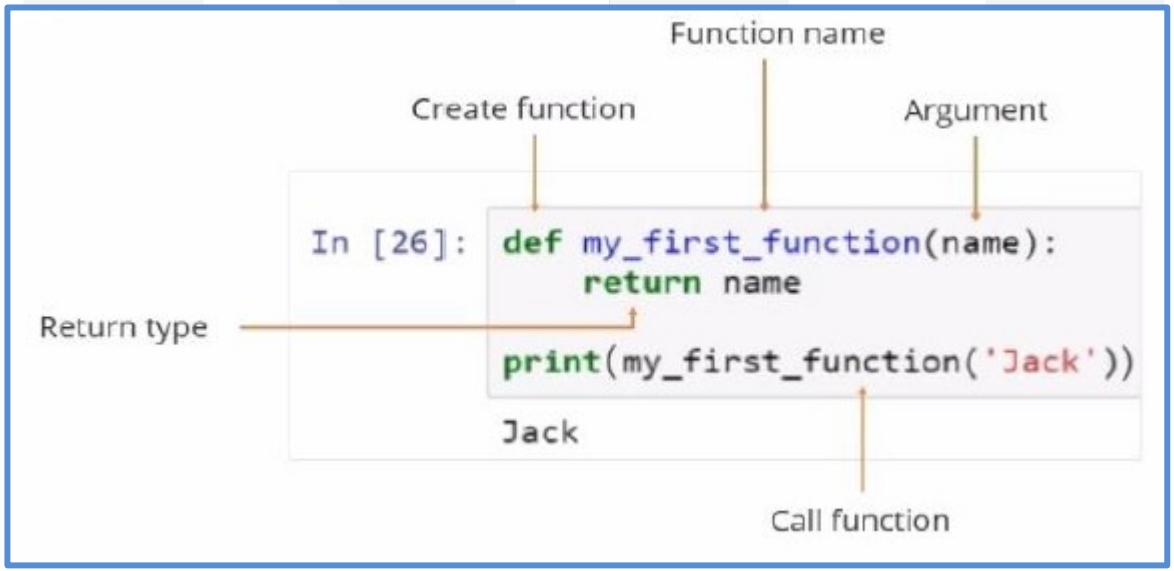
- We can avoid rewriting same logic/code again and again in a program.
- We can call python functions any number of times in a program and from any place in a program.
- We can trail a large python program without any difficulty when it is divided into multiple functions.
- Reusability is the main achievement of python functions.
- Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Syntax

```
def <function name> ( arg1, arg2...):
        //function executable statements
          return statement if any
```

Cont...

- Function declaration start with def keyword.
- Function should always return value.
- If function is not returning any value than its NONE
- Like java overloading of function is not possible in python



Cont...

```
In [137]: #using function one can retrun single value,
          #multiple value or no value
          def add_number(n1,n2):
              return n1+n2
          num1 = 12
          num2 = 100
          add_number(num1,num2) #calling of function

Out[137]: 112

In [138]: #storing answer in variable by calling function
          ans = add_number(num1,num2)
          print(ans)

          112
```

Function return multiple value

In [139]:
```
#return multiple value using retrun statement
def student_profile():
    age=21
    name="Kavya Patel"
    add = "Surat"
    B_group = "AB+"
    return age,name,add,B_group
#calling function
std_age,std_name,address,blood_group = student_profile()
print(std_age,std_name,address,blood_group)
```

```
21 Kavya Patel Surat AB+
```

In Built sequence function

enumerate
Indexes data to keep track of indices and corresponding data mapping

sorted
Returns the new sorted list for the given sequence

reversed
Iterates the data in reverse order

Zip
Creates lists of tuples by pairing up elements of lists, tuples, or other sequence

Image Source : Simplilearn

Parul®
University

DIGITAL LEARNING CONTENT

Built-in Sequence Functions: enumerate

In [140]:
```
state_india = ['Gujarat','Mharastra','Madhyapradesh','Aandra-Pradesh']
print(state_india)
```

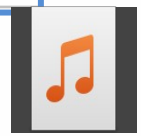
```
['Gujarat', 'Mharastra', 'Madhyapradesh', 'Aandra-Pradesh']
```

In [142]:
```
for index_val,name in enumerate (state_india):
    print(index_val,name)
```

```
0 Gujarat
1 Mharastra
2 Madhyapradesh
3 Aandra-Pradesh
```

In [144]:
```
#create dictionay from list using enumerate function
dict_state = dict((index_val,name) for index_val,name in enumerate(state_india))
print(dict_state)
```

```
{0: 'Gujarat', 1: 'Mharastra', 2: 'Madhyapradesh', 3: 'Aandra-Pradesh'}
```



Built in sequence function – sort() & sorted()

```
In [145]: new_list = [20,34,10,56,23,9,-1,34,11]
          new_list.sort()
          print(new_list)

          [-1, 9, 10, 11, 20, 23, 34, 34, 56]

In [146]: sorted([45,20,-1,0,45,89,2,56,34])
Out[146]: [-1, 0, 2, 20, 34, 45, 45, 56, 89]

In [148]: sorted("good morning")
Out[148]: [' ', 'd', 'g', 'g', 'i', 'm', 'n', 'n', 'o', 'o', 'o', 'r']

In [152]: tup=sorted((23,12,0,-5,35,21,45,200))
          print(tup)
          type(tup)

          [-5, 0, 12, 21, 23, 35, 45, 200]
Out[152]: list
```

Parul®
University

Built in sequence function – reversed() and zip()

```
In [166]: mylist2 = range(20)
          print(mylist2)

          range(0, 20)

In [167]: list(reversed(mylist2))

Out[167]: [19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

In [170]: sub1 = ['English','Gujarati','Hindi','Sanskrit']
          sub2 = ['one','two','three','four']
          tot_subject = zip(sub1,sub2)
          print(tot_subject)
          type(tot_subject)

          <zip object at 0x000001971A6771C0>

Out[170]: zip
```

Parul®
University

DIGITAL LEARNING CONTENT

Control statements – if, if else and if elif else

In [2]:
```
#if statement is use to check condition
#if condition is True than inner block is going to execute
#other wise branch (else) block is going to execute
#it also use for control the loop or execution of program based on condition
num=12
if(num>0):
    print("Positive number")
else:
    print("Negative Number")
```

```
Positive number
```

Multiple if statement

In [3]:
```
#more than one if statement
#condition is going to check in sequece
#execute all the if statement where condition become true
#follow top to bottom approch
if(num>0):
    print("Even number")
if(num%2==0):
    print("Even number")
if(num%3==0):
    print("number divisible by 3")
if(num%5==0):
    print("number divisible by 5")
else:
    print("number is not valid")
```

```
Even number
Even number
number divisible by 3
number is not valid
```

If elif elif ... else statement

In [5]:
```
#if elif statment
#is also use to check more than one condition
#it will stop the execution once conditoin become true
marks = 85
if(marks>85 and marks<=100):
    print("Higher Distinction")
elif(marks>75 and marks<=85):
    print("Distinction")
elif(marks>65 and marks<=75):
    print("First Class")
elif(marks>55 and marks<=65):
    print("Second Class")
elif(marks>45 and marks<=55):
    print("Pass Class")
else:
    print("Fail")
```

```
Distinction
```

Control Flow loop statement

In [10]:
```
#contol flow loop statment
#for loop
#print all the element in sequecne
city_list = ['Bombay','Vaodara','Surat','Nadiyad',None,'Navsari','Ahemdabad']
for i in city_list:
    print(i)
```

```
Bombay
Vaodara
Surat
Nadiyad
None
Navsari
Ahemdabad
```

In [11]:
```
for i in range(0,3):
    print(city_list[i])
```

```
Bombay
Vaodara
Surat
```

Parul®
University

DIGITAL LEARNING CONTENT

Control flow statement – break and continue

In [15]:
```
print(city_list)
for i in city_list:
    if(i is None):
        break
    print(i)
```

```
['Bombay', 'Vaodara', 'Surat', 'Nadiyad', None, 'Navsari', 'Ahemdabad']
Bombay
Vaodara
Surat
Nadiyad
```

In [14]:
```
for i in city_list:
    if(i is None):
        continue
    print(i)
```

```
Bombay
Vaodara
Surat
Nadiyad
Navsari
Ahemdabad
```

Parul® University

DIGITAL LEARNING CONTENT

Control Flow while statement

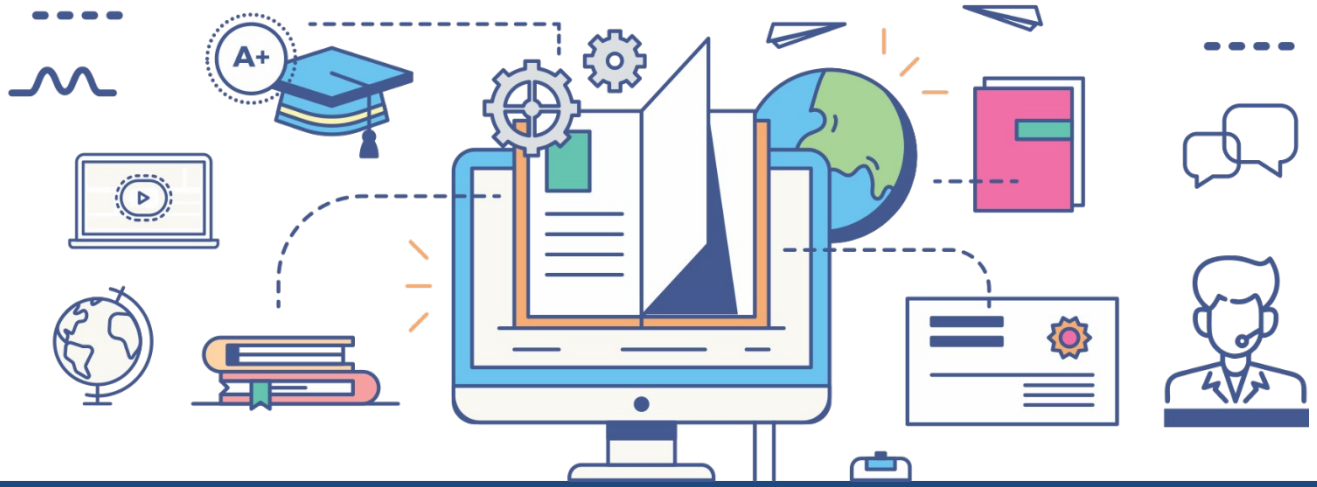
In [17]:
```
#while loop execute the block of code untill conditon become true
#stop execution once condition become false
num = 1
while(num<=10):
    print(num)
    num+=2
```

```
1
3
5
7
9
```

In [21]:
```
i=0
while(city_list[i] is not None):
    print(city_list[i])
    i+=1
```

```
Bombay
Vaodara
Surat
Nadiyad
```

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in