Introduction on

Python Fundamental Programming II

Kamal

AptComputingAcademy

Contents

Classes

Modules

Packages

File Handling

Map, reduce, list, filter

Handling excel/csv files

Exception handling

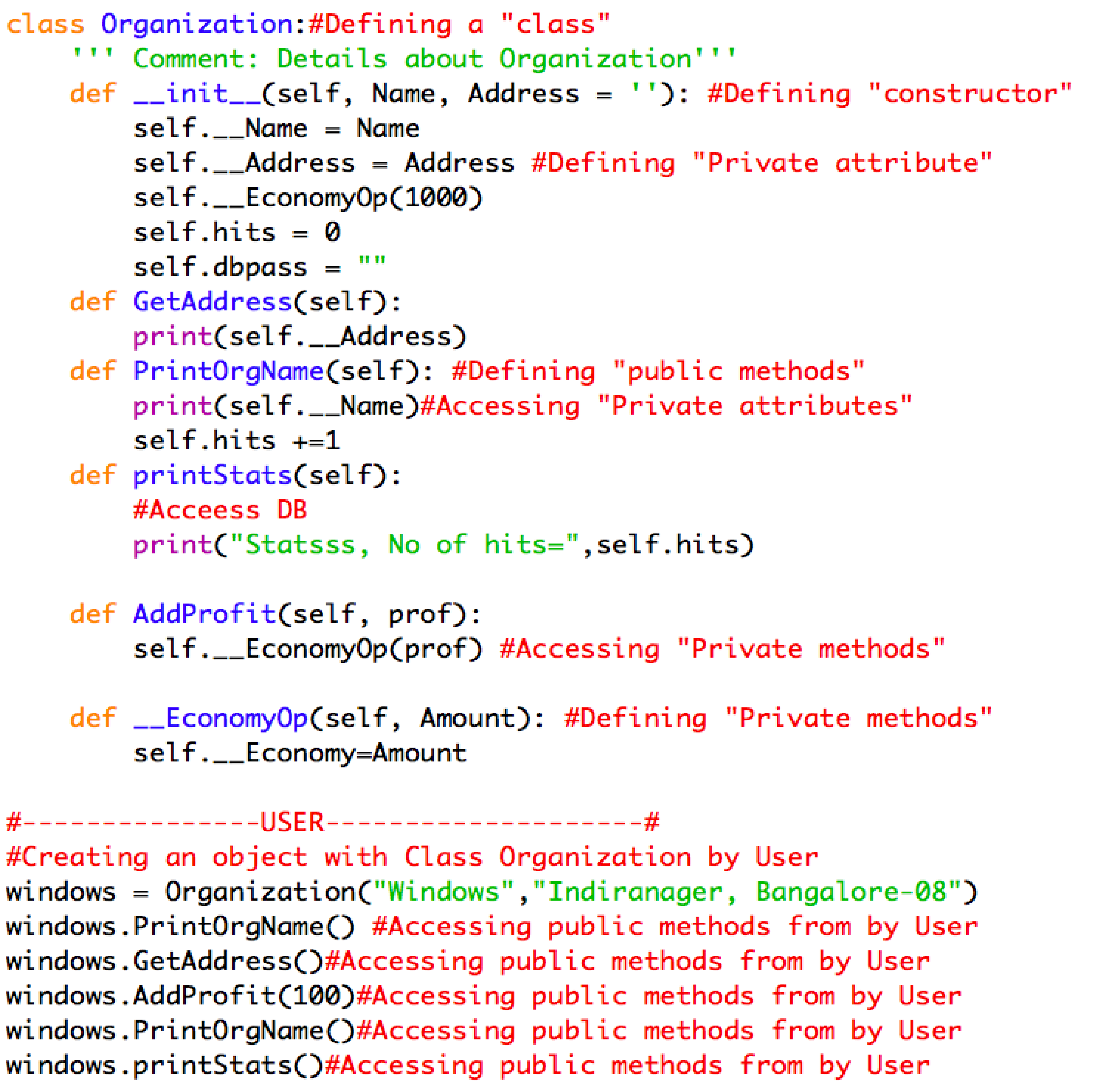
Standard modules:

- OS
- Sys
- Math
- Numpy
- Json

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Basic OOPs concepts**
  - Classes and objects
    - Creating/Accessing attributes (Public)
    - Creating/Accessing methods (Public)
    - Creating constructors
  - Encapsulation
    - Creating attributes (private)
    - Creating/Accessing methods (private)

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
    - OS
    - Sys
    - Math
    - Numpy
    - Json



```
class Organization:#Defining a "class"
    ''' Comment: Details about Organization'''
    def __init__(self, Name, Address = ''): #Defining "constructor"
        self.__Name = Name
        self.__Address = Address #Defining "Private attribute"
        self.__EconomyOp(1000)
        self.hits = 0
        self.dbpass = ""
    def GetAddress(self):
        print(self.__Address)
    def PrintOrgName(self): #Defining "public methods"
        print(self.__Name)#Accessing "Private attributes"
        self.hits +=1
    def printStats(self):
        #Acceess DB
        print("Statsss, No of hits=",self.hits)

    def AddProfit(self, prof):
        self.__EconomyOp(prof) #Accessing "Private methods"

    def __EconomyOp(self, Amount): #Defining "Private methods"
        self.__Economy=Amount

#--------------USER-------------------#
#Creating an object with Class Organization by User
windows = Organization("Windows","Indiranager, Bangalore-08")
windows.PrintOrgName() #Accessing public methods from by User
windows.GetAddress()#Accessing public methods from by User
windows.AddProfit(100)#Accessing public methods from by User
windows.PrintOrgName()#Accessing public methods from by User
windows.printStats()#Accessing public methods from by User
```

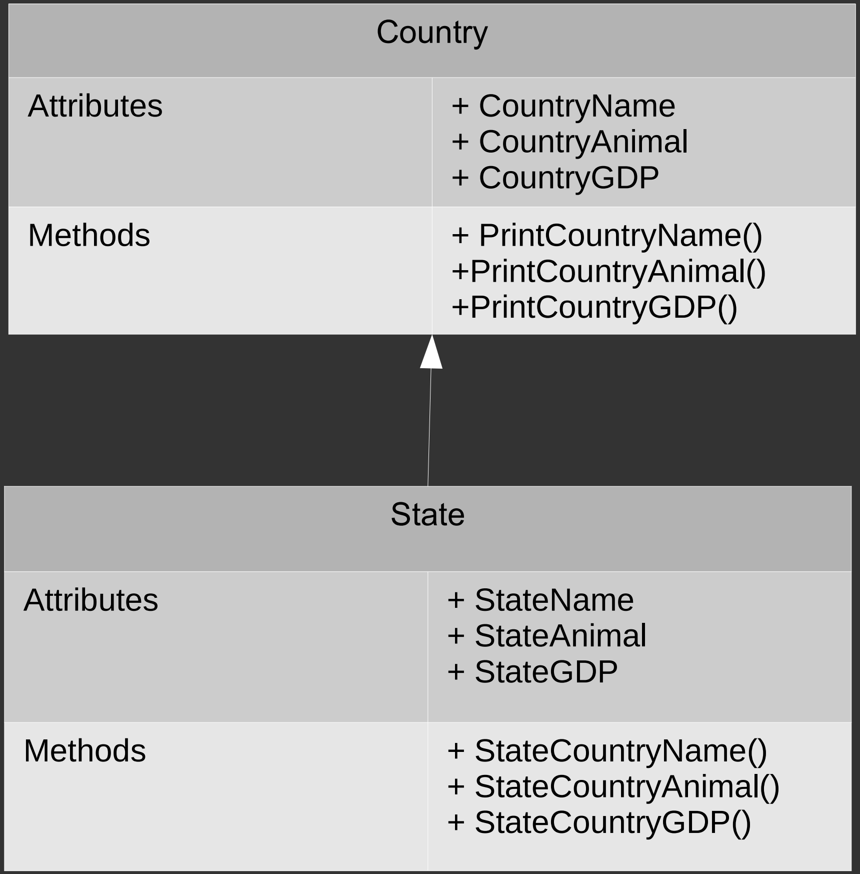
- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Basic OOPs concepts:

- Inheritance
  - Simple Example of inheritance
  - Multiple
  - Multi-level
    - .__mro__, .issubclass(obj1, class)

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- Inheritance
  - Simple Example of inheritance
  - Multiple
  - Multi-level



Country	
Attributes	+ CountryName + CountryAnimal + CountryGDP

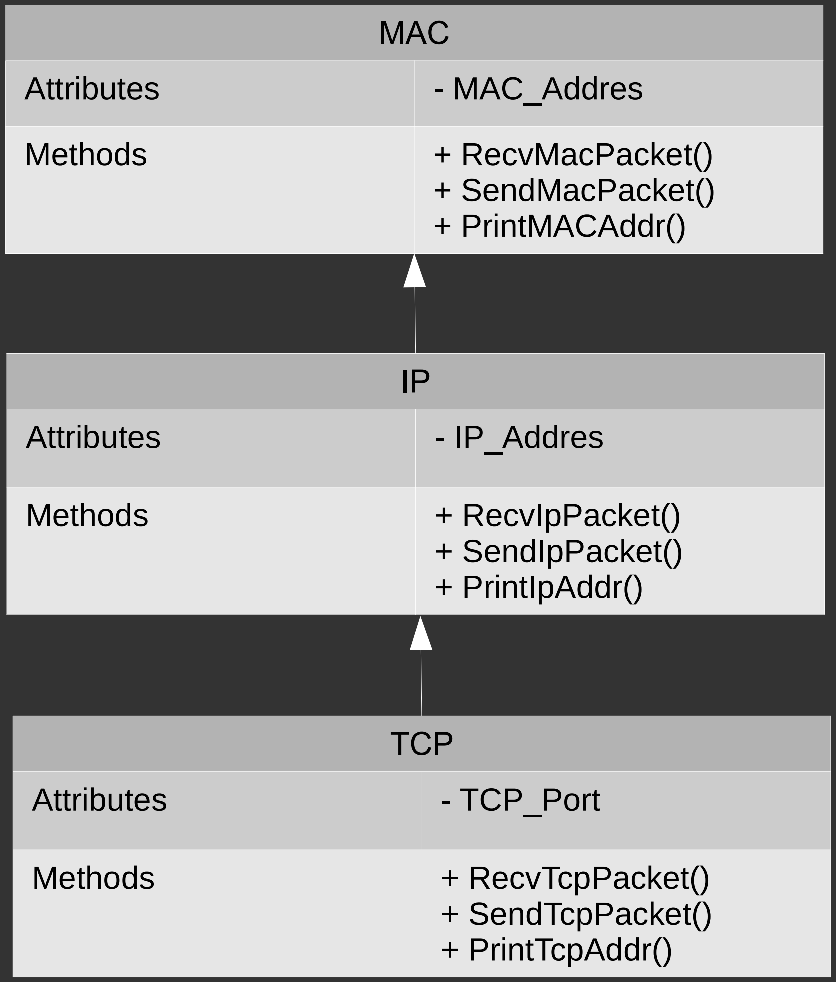
Methods	+ PrintCountryName()
	+PrintCountryAnimal()
	+PrintCountryGDP()

State	
Attributes	+ StateName + StateAnimal + StateGDP

Methods	+ StateCountryName()
	+ StateCountryAnimal()
	+ StateCountryGDP()

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- Inheritance
  - Simple Example of inheritance
  - Multiple
  - Multi-level



MAC	
Attributes	- MAC_Addres

Methods	+ RecvMacPacket()
	+ SendMacPacket()
	+ PrintMACAddr()

IP	
Attributes	- IP_Addres

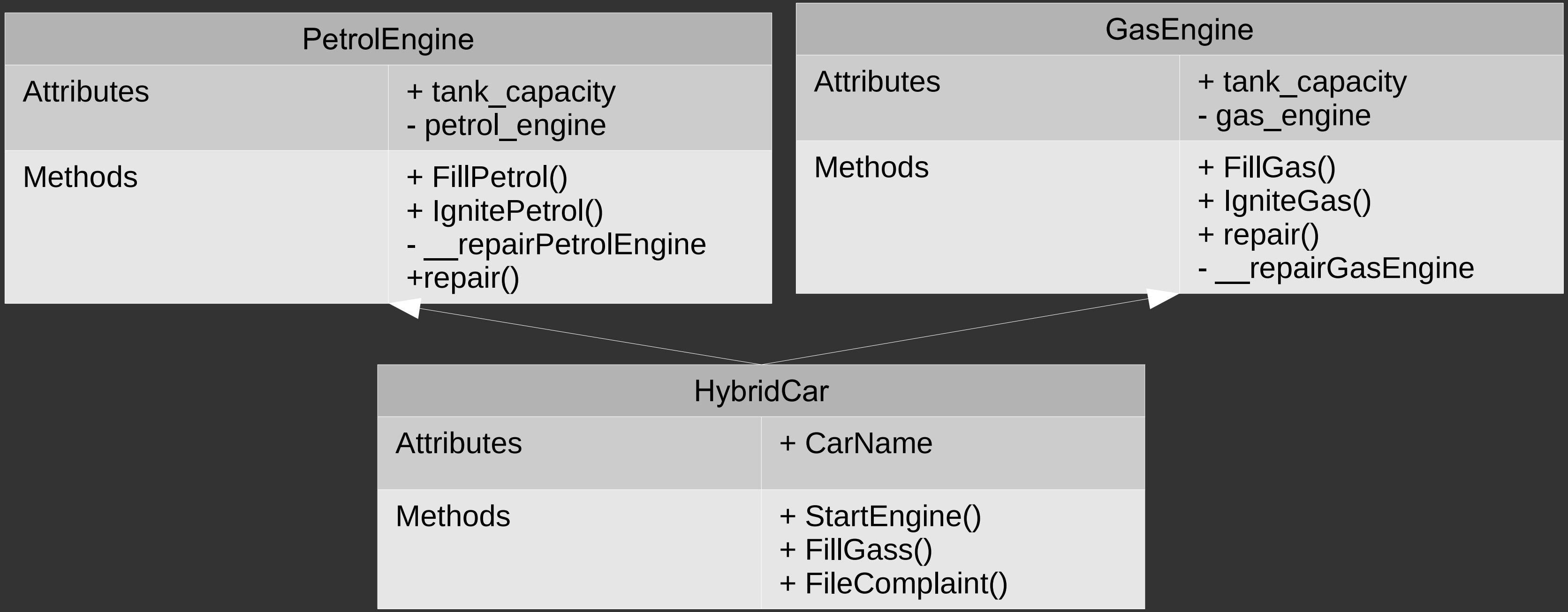
Methods	+ RecvIpPacket()
	+ SendIpPacket()
	+ PrintIpAddr()

TCP	
Attributes	- TCP_Port

Methods	+ RecvTcpPacket()
	+ SendTcpPacket()
	+ PrintTcpAddr()

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- Inheritance
  - Simple Example of inheritance
  - Multiple
  - Multi-level



PetrolEngine	
Attributes	+ tank_capacity - petrol_engine

Methods	+ FillPetrol()
	+ IgnitePetrol()
	- __repairPetrolEngine
	+repair()

GasEngine	
Attributes	+ tank_capacity - gas_engine

Methods	+ FillGas()
	+ IgniteGas()
	+ repair()
	- __repairGasEngine

HybridCar	
Attributes	+ CarName

Methods	+ StartEngine()
	+ FillGass()
	+ FileComplaint()

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Advanced OOPs concepts

- Class and instance Variables
  - Class variables: Can be accessed using Class and instance
  - Instance variables: Can be accessed using only instances

```
Example: Creating and setting class variable
hybryd_car.CompanyName = "Honda" #---> hybryd_car is a class

self.__class__.CompanyName = "Scoda" #--> Using instance
method

Example: Creating and setting instance variable
mycar.color = "Red" #---> mycar is an instance
```

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Advanced OOPs concepts

Class methods:

- **To access class variables.**
- **Can not access instance variables**

```
@classmethod
def setCompanyName(cls, Name):
        cls.CompanyName = Name
```

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Advanced OOPs concepts**
  - **Static methods:**
    - **Can not access/modify instance/class attributes/methods**

```
@staticmethod
def static_method(i):
        print("Value of i = ", empnum)
```

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Comparison:

	Instance variable/ method	Class variable	
Instance method	Yes	Yes	
Class method	No	Yes	
Static method	No	No	

- **Classes**
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Operator overloading:

Operator	Expression	Internally
Addition	p1 + p2	p1.__add__(p2)
Subtraction	p1 - p2	p1.__sub__(p2)
Multiplication	p1 * p2	p1.__mul__(p2)
Power	p1 ** p2	p1.__pow__(p2)
Division	p1 / p2	p1.__truediv__(p2)
Floor Division	p1 // p2	p1.__floordiv__(p2)
Remainder (modulo)	p1 % p2	p1.__mod__(p2)
Bitwise Left Shift	p1 << p2	p1.__lshift__(p2)
Bitwise Right Shift	p1 >> p2	p1.__rshift__(p2)
Bitwise AND	p1 & p2	p1.__and__(p2)
Bitwise OR	p1 | p2	p1.__or__(p2)
Bitwise XOR	p1 ^ p2	p1.__xor__(p2)
Bitwise NOT	~p1	p1.__invert__()

- Classes
- **Modules**
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Modules: (Introduction)

- **Sets of functions, classes, variables**
- **Other programs can use modules by importing**
- **Module is a simple ".py" file**

1) How to use existing modules/standard modules?

2) How to create Own modules?

3) How to import Own modules and use?

4) How to make own modules available for import from any directory

- Classes
- **Modules**
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Modules:

1) How to use existing modules/standard modules?

```
import os #Importing OS module
import sys #Importing sys module
```

2) How to create Own modules?

```
A simple python file
```

- Classes
- **Modules**
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Modules:**

3) How to import Own modules and use?

Way to import	Comment
`import cars`	Importing “cars” **module**
`import cars as mycars`	Importing “cars” module as **mycars**
`from cars import version`	Importing a **variable** from module “cars”
`from cars import carclass`	Importing a **class** from module “cars”
`from cars import CreateCarObjects`	Importing a **method** from module “cars”
`from cars import SampleCar`	Importing an **object** from module “cars”

- Classes
- **Modules**
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Modules:**

3) How to import Own modules and use? ....cont

Way to import	Comment
from math **import** sin,cos	Lifting some symbols out from the module and making them available locally
from math **import** *	Import everything from module as local

- Classes
- **Modules**
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

➢ Modules:

4) How to make own modules available for import from any directory

- **Add path in environment variable “PYTHONPATH” in ~/.bash_profile**

```
export
PYTHONPATH=/Users/kamalmukiri/Documents/AptComputingAc
ademy/Python/Classes/Classes/3.\ Fundamental\
Programming\ II/modules/automobile
```

- **Add path in PyCharm**

https://stackoverflow.com/questions/17198319/how-to-configure-custom-pythonpath-with-vm-and-pycharm

- Classes
- **Modules**
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Modules:

Good practices:

- File names have to follow the rules
- Comments makes life easy
- Avoid non-ASCII chars
- Module names/Packages should be named with lower case
- Don't user names which is similar to standard modules
- Reload works.... Try to avoid
- Put conditions to run (Like check module name as __main__ to run as only main program)

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Introduction:

- **For large collection of code, it is desired to keep the modules in hierarchy.**

```
mycompany/
    automobile/
        cars.py
        bikes.py
    humanresource/
        employee.py
        contract.py
    web/
        web_news.py
        web_adv.py
```

```
Mycompany/
    __init__.py
    automobile/
        __init__.py
        cars.py
        bikes.py
    humanresource/
        __init__.py
        employee.py
        contract.py
    web/
        __init__.py
        web_news.py
        web_adv.py
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Importing Package:

```
from mymodules.automobile.cars import myversion
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Setting paths in ENV/PyCharm is same as modules.. :):)**

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- Classes
- Modules
- Packages
- **File Handling**
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

File Open:

Syntax:

```
file object = open(file_name [, access_mode][, buffering])

file_name          -->   A file name
access_mode        -->
```

Option	Read	Write	Append/ Overwrite	Create
r	Yes	No	NA	No
r+	Yes	Yes	Over write	No
w	No	Yes	Over write	Yes
w+	Yes	Yes	Over write	Yes
a	No	Yes	Append	Yes
a+	Yes	Yes	Append	Yes

```
Buffering          -->  10000 bytes
```

- Classes
- Modules
- Packages
- **File Handling**
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **File read**
  - **Read, readline, readlines**
- **File seek, tell**
- **File write**
- **File close**
- **Attributes**
  - **Name, closed, mode**

- Classes
- Modules
- Packages
- **File Handling**
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- File open
  - Modes:
    - Read, write,append
    - Create
- File read
  - Read, readline, readlines
- File seek, tell
- File write
- File close
- Attributes
  - Name, closed, mode

- Classes
- Modules
- Packages
- **File Handling**
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Options:

- Buffering (buffering)
  - Buffering = 0 ---> Switch off (Only binary files)
  - Buffering = 1 ---> Line buffering (Only for Txt files)
  - Buffering > 1 --> Setting buffer size
  - Buffering = -1 --> Default (set by OS)
- NewLine (newline) = ‘\n’ ‘\r’ ‘\rn’
  - Notes: \n ---> Unix

    \r ----> Old Unix

    \r\n ---> Windows
- Encoding (encoding)
  - https://docs.python.org/2.4/lib/standard-encodings.html

- Classes
- Modules
- Packages
- **File Handling**
- Map, reduce, list, filter
- Handling excel/csv files
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Methods:**
  - readable()
  - writeable()
  - fileno()
  - flush()
  - isatty()
  - truncate()
    - Ex: file.truncate(100) ---> Truncate the file size to 100 bytes

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Introduction:

- **Do not want to terminate the program, there should be softer way of dealing with errors.**
- **Need to crate own exception classes**
- **What is ERROR?**
- **What is EXCEPTION?**

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Syntax error Vs Exception

- How to RAISE EXCEPTION?
  - List of Standard Exception:
  - https://docs.python.org/3/library/exceptions.html
- How to ASSERT?

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

```
try:
    --------Statement1
except (ErrorName1, ErrorName2):
    --------Statement2
else:
    --------Statement3
finally:
    --------Statement4
```

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Write own exceptions sub class:

```
class MyException(Exception):
    def __init__(self, *arg, **kwargs):
          Exception.__init__(self, *arg, **kwargs)
```

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Best practices:

- **Clean all the resources in “finally” block.**
- **Use Context managers (like ‘with’)**
- **Use Decorators**
- **Use locks**

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Making Context manager:

```
class MyFile():
    def __init__(self, file, mode, buffering=1024*1024,
encoding='utf-8'):
        self.file = file
        self.mode = mode
        self.buffering = buffering
        self.encoding = encoding
    def __enter__(self):
        self.fd = open(self.file, mode = self.mode,buffering
= self.buffering, encoding = self.encoding)
        return self.fd
    def __exit__(self, *args):
        self.fd.close()
        print("Closed the open file")

with MyFile("Kamal.txt", "w+", 1024*1024) as fd:
    fd.write("Welcome")
```

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Making Context manager:

- **Use of __exit__:**
  - **Useful in case of exceptions raised in __enter__ or (by user or by python)**

```
def __exit__(self, exception_type, exception_value, traceback):
```

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Decorators:

- **Decorator is a function that takes another function and extends the behaviour.**

```
def decorator_fun(fun):
    addings = "pepper, coriander"
    def wraper():
        fun()
        print("Added "+addings)
        print("Done")
    return wraper
@decorator_fun
def make_chicken_curry():
    print("Boiled chicken")
```

- Classes
- Modules
- Packages
- File Handling
- Map, reduce, list, filter
- Handling excel/csv files
- **Exception handling**
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Using Locks:

- **Locks are used to use common resources to access modify in synchronous.**

```
Syntax:
import threading
lock = threading.Lock()

lock.acquire() ---> To lock

lock.release() ---> To unlock
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- Lambda function (an anonymous function without name):
  - Uses of lambda functions:
    - Simple to write
    - Boosts performance together with map

```
Syntax:
lambda arguments: expression
Ex:
double = lambda x: x * 2
add = lambda x,y: x+y


Rules:
1) There can be number of arguments, expression
should be one
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter,lambda**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- map(fun, itr)
  - **Returns a list of the results after applying the given function to each item of a given itr (list/tuple).**

```
Syntax:
    map(fun, iter)

Example:
output = list(map(lambda x,y:x+y, mylist1, mylist2))
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

filter(fun, itr)

Filter out all the elements of a sequence "mylist1", for which the function *"lambda"* returns True/1.

```
Example:
output = list(filter(lambda x: x<1000, mylist1))
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Reduce:**
  - **At first step, first two elements of sequence are picked and the result is obtained.**
  - **Next step is to apply the same function to the previously attained result and the number just succeeding the second element and the result is again stored.**
  - **This process continues till no more elements are left in the container.**
  - **The final returned result is returned and printed on console**

```
Syntax:
    reduce(fun,seq)
Example:
import functools
lis = [ 1 , 3, 5, 6, 2, ]
print(functools.reduce(lambda a,b : a if a > b else
b,lis))
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Why should we use map?

```
Example 01:
from multiprocessing import Pool

def f(x):
    os.getpid()
    return x*x

if __name__ == '__main__':
    p = Pool(5)
    print(p.map(f, [1, 2, 3]))


Example 02:
map(str, range(10**100))
[str(n) for n in range(10**100)]
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Iterators:

- **What is it?**
  - **Objects that allow you to traverse through all the elements of a collection**
- **Existing Example:**
  - **range(10)**
- **How to write our own?**
  - **Create a class and define __next__ and __iter__ as member functions**

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Iterators:

Example:

```
#Creating class:
class myIter:
    def __init__(self,min_num, max_num=0, interval =1):
        if max_num == 0:
            self.min = 0
            self.max = min_num
        else:
            self.max = max_num
            self.min = min_num
            self.inter = interval
    def __iter__(self):
        return self
    def __next__(self):
        self.min += 1
        return self.min-1

#Using myIter:
obj = myIter(10)
print(next(obj))
print(next(obj))
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Containers:**
  - **What is a container?**
    - **Objects that hold data values. They support membership tests, which means you can check if a value exists in the container**
  - **How to create a container?**
    - **No special code is required.**

```
#Using myIter:
obj = myIter(10)
print(next(obj))
print(next(obj))

if 3 in obj:
    print("Yes....")
```

- Classes
- Modules
- **Packages**
- File Handling
- **Map, reduce, list, filter**
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Generators:**
  - **What is a Generator?**
    - **Behaviour of “yield”**

      1) **Generator is a fun which uses benefit of yield**

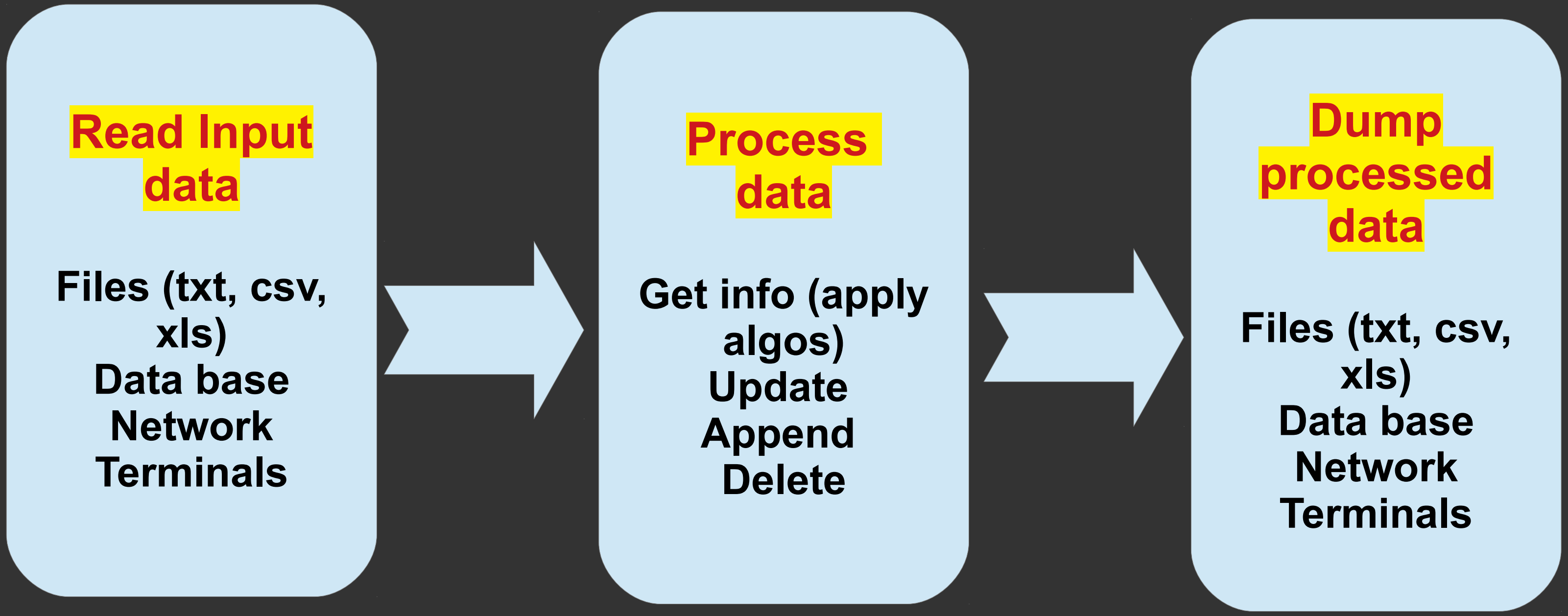
      2) **Generator can be an expression**
  - **How to write a Generator?**

```
Example Function:
def myGen(num):
    i = 0
    while i < num:
        yield i
        i += 1
```

```
Example Expression:
squares = (x * x for x in range(1,10))
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

===========DATA PROCESSING=========



Read Input data

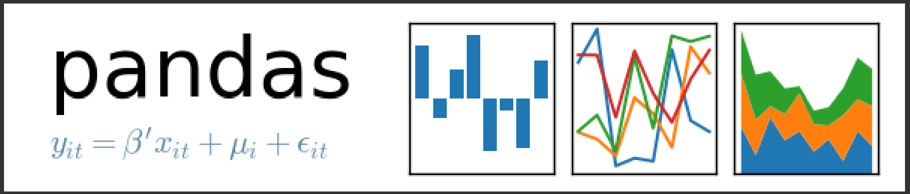
Files (txt, csv, xls)
Data base
Network
Terminals

Process data

Get info (apply algos)
Update
Append
Delete

Dump processed data

Files (txt, csv, xls)
Data base
Network
Terminals



- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

===========DATA PROCESSING=========

(Examples)

1) Online purchases

2) Agriculture

3) Power grid

4) Retails shops

5) Manufacturing units

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

- **Data frames:**
  - **What is Data frame?**
    - A simple table of data having two dimensions.
      - Multiple rows and columns
      - Each row represents a sample of data
      - The data in same column has same data type
      - Avoids missing values
  - **How to create Data frame?**

    1) Manual

    2) Loading files (csv/xlxs/SQL data base)

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Pandas Data Frame

- **How to create Data Frame (Manually)?**
- **How to traverse each Cell in data frame?**
-

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Hands on Data frames:

1) Creating data frame manually

```
import os
import pandas as pd #Importing pandas module
data1 = {’Name’:["apple","samsung","LG","huawei","HTC"],
’Rank’:[1,2,3,4,5]}
#1) Creating a data frame
df = pd.DataFrame(data=data1)
df = pd.DateFrame([[“Apple”,”Samsung”],
[1,2]],columns=[“Name”,”Rank”]

#2) Getting details: size
print(df.size)

#3) Getting number of dimensions
print(df.ndim)

#4) Getting data types of each column
print(df.dtypes)

#5) Getting the entire column of "Name"
print("Names:",df.Name)
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Hands on Data frames:

1) Creating data frame manually

```
import os
import pandas as pd #Importing pandas module
data1 = {’Name’:["apple","samsung","LG","huawei","HTC"],
’Rank’:[1,2,3,4,5]}
#1) Creating a data frame
df = pd.DataFrame(data=data1)
df = pd.DateFrame([[“Apple”,”Samsung”],
[1,2]],columns=[“Name”,”Rank”]

#2) Getting details: size
print(df.size)

#3) Getting number of dimensions
print(df.ndim)

#4) Getting data types of each column
print(df.dtypes)

#5) Getting the entire column of "Name"
print("Names:",df.Name)
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

Hands on Data frames:

2) Updating data frame manually

```
#6) Appending another table to df
df2 = pd.DataFrame({"Name":["oppo","micromax"],  "Rank":
[11,12]}, index=['x','y'])
df = df.append(df2)
print(df)

#7) iloc: All indexing workds
df.iloc[0,1] = 10
df.iloc[:]
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

CSV: Comma Separated Values:

Using data frame return by “read_csv”

```
import pandas as pd

Data = pd.read_csv(“file”,[options])

Data.shape()
Data.head()
Data.iloc[x,y]
Data.Name[0]
Data.Name
Data[Data.Name = ‘kamal’]
Data[‘Field’], Data.Field
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

CSV: Comma Separated Values:

Read:

```
import pandas as pd

Data = pd.read_csv(“file”,[options])
    Options:
        sep = “:”
        encoding = “utf-8”
        names = [“Name”,”Phone”]
        na_values = [“NA”,-1,”not”]
        nrows = 10

More options:
https://pandas.pydata.org/pandas-docs/version/0.21/generated
/pandas.read_csv.html
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

CSV: Comma Separated Values:

Write:

```
import pandas as pd

Data = pd.read_csv(“file”,[options])
.
.
.
Data.to_csv(“new_name”, [options])
    Options:
        index = False/True
        columns = [“col1”,”col3”]
        header = True/False
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Numpy
  - Math
  - **Json**

JSON module in Python:

- Convert from JSON to Python and Python to JSON format

```
import json
x =  '{ "name":"John", "age":30, "city":"New York"}'
y = json.loads(x)   -----> Converting string to dict

Z = json.dumps(y)   -----> Converting dict to string
```

- Json.dumps():
  - intend = True
  - sort_keys = True

Python	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Numpy
  - **Math**
  - Json

Math module in Python:

- It provides mathematical functions which are defined by C standards.
- **Operations:**
  - ceil(), floor(), factorial(), fabs()
- **Logarithmic and exponential:**
  - exp(x), log(x,y), log2(), log10(), pow(x,y),sqrt()
- **Trigonometric functions:**
  - sin(), cos(), tan(), asin(), singh(), cosh(),....
- **Angular functions:**
  - math.degrees(), math.radians()
- **Constants:**
  - **math.pi, math.e, math.tau, math.nan**

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Numpy
  - **Math**
  - Json

NumPy:

- To create multidimensional arrays and perform operations.
-

```
import numpy as np
#1) Creating ndarray of 10 elements
array = np.arange(10)

#2) Reshape to 5x2
array1 = array.reshape(5,2)
print(array1.size)
print(array1.ndim)
print(array1.itemsize)
print(array1.dtype.name)

#3) Creating ndarray by existing array
a = np.array([2,3,4])
c = np.array( [ [1,2], [3,4] ], dtype=complex)
```

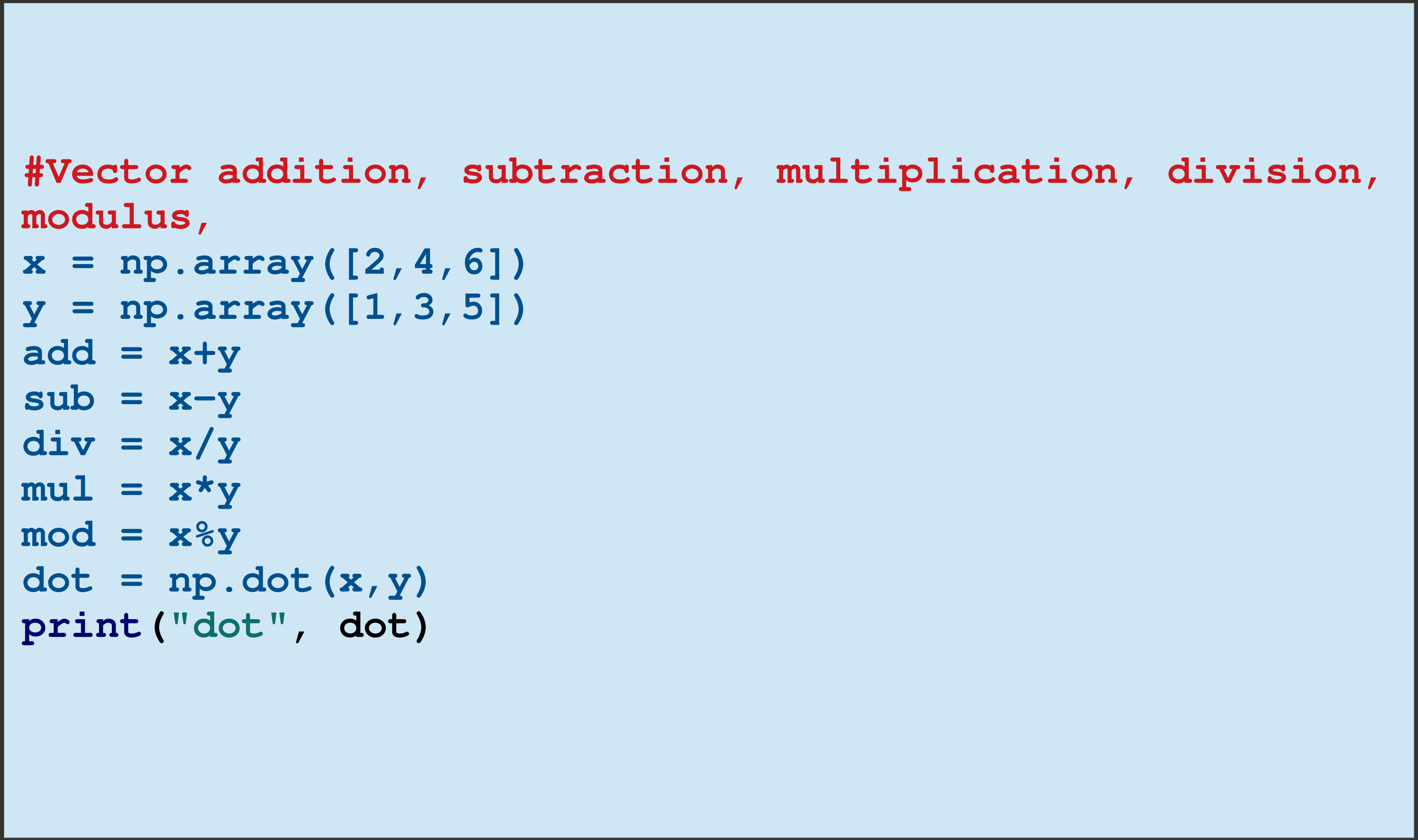
- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Numpy
  - **Math**
  - Json

NumPy:

- To create multidimensional arrays and perform operations.
-
```
#4) Creating zeros/ ones array
zarray = np.zeros((3,4))
onearray = np.ones((2,3))

#5) Creating sequence
seq = np.arange( 0, 2, 0.3 )
seq = np.linspace(0,90,20)
print(seq)
print(np.sin(seq))

def f(x,y):
    return x+y

#6) Creating sequences using "fromfunction"
array2 = np.fromfunction(f,(5,4),dtype=int)
print(array2)
#Indexing is common
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Numpy
  - **Math**
  - Json

NumPy:

- Vector operations
- 

```
#Vector addition, subtraction, multiplication, division, modulus,
x = np.array([2,4,6])
y = np.array([1,3,5])
add = x+y
sub = x-y
div = x/y
mul = x*y
mod = x%y
dot = np.dot(x,y)
print("dot", dot)
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Numpy
  - **Math**
  - Json

NumPy:

- Matrix operations
-

```
#Matrix operations
x = np.matrix( ((2,3), (3, 5)) )
y = np.matrix( ((1,2), (5, -1)) )
print(x+y)
print(x*y)
```

- Classes
- Modules
- **Packages**
- File Handling
- Map, reduce, list, filter
- **Handling excel/csv files**
- Exception handling
- Standard modules:
  - OS
  - Sys
  - Math
  - Numpy
  - Json

XLS:

```
import pandas as pd

Data = pd.read_xls(“file”,[options])

Data.shape()
Data.head()
Data.iloc[x,y]
Data.Name[0]
Data.Name
Data[Data.Name = ‘kamal’]
Data[‘Field’], Data.Field
```

Thank You……