

Data Science & Artificial Intelligence



Python For Data Science

Classes & Modules

Lecture No.- 02

By- Satya sir



Recap of Previous Lecture



- Set methods/operations

- Dictionaries

- Functions

- Recursion



Topics to be Covered



- Method overloading
- method overriding
- Constructor
- Modules
 - math
 - datetime
 - numpy



Topic : Modules – datetime, math, numpy

Method overloading :

- The ability to define two or more methods with same name but difference in the arguments, is called [with in the same class]

Method overloading.

Ex:

```
class A:
```

```
    def x(i, j):
```

```
        Print(i + j)
```

```
    def x(x, y, z):
```

```
        Print(x * y * z)
```

```
ob = A()
```

```
ob.x(5, 7) # 12
```

```
ob.x(4, 5, 6) # 120
```




Topic : Modules – datetime, math, numpy

Method-Overriding : The ability to define method with same Name and Signature (arguments) in Parent class and child class is known as method-overriding.

Ex:

```

class A:
    def M1():
        Print('Hello')

class B(A):
    def M2():
        Print('Bye')
    def M1():
        Print('Hau')

ob = B()
ob.M1() # Hau
ob.M2() # Bye
    
```

← override

Method

- defined within class
- Object • method (arguments)
- At a time only one object can be used with method

x.y.z.M1()

Invalid

x.M1()
 y.M1()
 z.M1()

Valid

function

- defined outside class
- function (objects)
- Multiple objects can be arguments for a function.

Ex: function(x, y, z)

x = 4

y = 2.7

z = 'GATE'

def f(a, b, c):

Print(a, b, c)

f(x, y, z)

 ↙ object of class int

 ↘ object of class str



Topic : Modules – datetime, math, numpy

Super() Statement : To access overridden members of Super class.

class A :

def M1() :
 Print('Hello')

class B(A) :

def M1() :
 Super().M1()
 Print('Hai')

def M2() :
 Print('Bye')

ob = B()

ob.M1() # Hello # Super(B, ob).M1()
 # Hai

ob.M2() # Bye



Topic : Modules – datetime, math, numpy

Constructor : It is also a method, invoked / called / executed while Instance / object creation, automatically.

↳ So, It is called Initialization method.

Syntax : `def __init__(self):`
 `# statements(s)`

Ex:

```
class A:  
    result = 0  
    def __init__(self, Var1, Var2):  
        self.Var1 = Var1  
        self.Var2 = Var2  
    def display():  
        result = self.Var1 + self.Var2  
        print(result)
```

`ob = A(4, 5)`
`ob.display()`
9



Topic : Modules – datetime, math, numpy

args, kwargs \Rightarrow Both are Used to Pass Variable No. of arguments to a function/method.

args \rightarrow Non-keyword arguments
kwargs \rightarrow keyword arguments

Ex:

```
def f1(*args):  
    for i in args:  
        Print(i)
```

`f1(7, 5, 2)`

`f1("GATE", "EXAM")`

`f1(3, 5.2, True, False)`

Ex:

```
def f(**kwargs):  
    for key, value in kwargs.items():  
        Print('%.s = %.s', %. (key, value))
```

`f({'a': 3, 'b': 4, 'c': 5})`

o/p: `a=3`

`b=4`

`c=5`

Access Modifiers

- 1) Public : By default Every member is Public.
- 2) Protected : `_` (single underscore) Prefix, it is Protected member. It can be accessed in only it's inherited classes.
- 3) Private : - Any member with `__` (double underscore) Prefix is Private member.

`a=10 #Public` - Private members are accessible only within respective class by its own object only.

```
class A:
    _b=20 #Protected
    __c=30 #Private
    def M():
        print(a, _b, __c, 'In class A')
        # 10 20 30 in class A
```

```
class B(A):
    def M1():
        print(a) # 10
        print(Super()._b) # 20
        print(Super().__c)
        # Error
```

```
class C:
    def M2():
        print(a) # 10
        print(_b) # Error
        print(__c) # Error

ob=B()
print(ob._b) # Valid
print(ob.__c) # Invalid
```


Module :

- Any or Every Program in Python is called a Module.
- 2 Types of modules :
 - 1) Predefined modules : datetime, math, os, sys, Collection, numpy - - -
 - 2) User defined modules : Every Python Program saved as .py extension.
- To Use/avail services of one module in another, import keyword is used.

import ModuleName

(OR) import Package * (All modules in Package)

(OR) from module import Class

Ex: from math import cos



Topic : Modules – datetime, math, numpy

`Print (strftime (%directive))`



Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513



Topic : Modules – datetime, math, numpy

Ex: `import datetime`
`ob = datetime.now()`
`Print(ob.strftime("%A"))`
o/p: Tuesday



Directive	Description	Example
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%C	Century	20
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%
%G	ISO 8601 year	2018
%u	ISO 8601 weekday (1-7)	1
%V	ISO 8601 weeknumber (01-53)	01



Topic : Modules – datetime, math, numpy

Method	Description
<u>math.acos()</u>	Returns the arc cosine of a number
<u>math.acosh()</u>	Returns the inverse hyperbolic cosine of a number
<u>math.asin()</u>	Returns the arc sine of a number
<u>math.asinh()</u>	Returns the inverse hyperbolic sine of a number
<u>math.atan()</u>	Returns the arc tangent of a number in radians
<u>math.atan2()</u>	Returns the arc tangent of y/x in radians
<u>math.atanh()</u>	Returns the inverse hyperbolic tangent of a number
<u>math.ceil()</u>	Rounds a number up to the nearest integer
<u>math.comb()</u>	Returns the number of ways to choose k items from n items without repetition and order
<u>math.copysign()</u>	Returns a float consisting of the value of the first parameter and the sign of the second parameter
<u>math.cos()</u>	Returns the cosine of a number
<u>math.cosh()</u>	Returns the hyperbolic cosine of a number
<u>math.degrees()</u>	Converts an angle from radians to degrees
<u>math.dist()</u>	Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point
<u>math.erf()</u>	Returns the error function of a number
<u>math.erfc()</u>	Returns the complementary error function of a number



Topic : Modules – datetime, math, numpy

Method	Description
<u>math.exp()</u>	Returns E raised to the power of x
<u>math.expm1()</u>	Returns $E^x - 1$
<u>math.fabs()</u>	Returns the absolute value of a number
<u>math.factorial()</u>	Returns the factorial of a number
<u>math.floor()</u>	Rounds a number down to the nearest integer
<u>math.fmod()</u>	Returns the remainder of x/y
<u>math.frexp()</u>	Returns the mantissa and the exponent, of a specified number
<u>math.fsum()</u>	Returns the sum of all items in any iterable (tuples, arrays, lists, etc.)
<u>math.gamma()</u>	Returns the gamma function at x
<u>math.gcd()</u>	Returns the greatest common divisor of two integers
<u>math.hypot()</u>	Returns the Euclidean norm
<u>math.isclose()</u>	Checks whether two values are close to each other, or not
<u>math.isfinite()</u>	Checks whether a number is finite or not



Topic : Modules – datetime, math, numpy

Method	Description
<u>math.isinf()</u>	Checks whether a number is infinite or not
<u>math.isnan()</u>	Checks whether a value is NaN (not a number) or not
<u>math.isqrt()</u>	Rounds a square root number downwards to the nearest integer
<u>math.ldexp()</u>	Returns the inverse of <u>math.frexp()</u> which is $x * (2^{**}i)$ of the given numbers x and i
<u>math.lgamma()</u>	Returns the log gamma value of x
<u>math.log()</u>	Returns the natural logarithm of a number, or the logarithm of number to base
<u>math.log10()</u>	Returns the base-10 logarithm of x
<u>math.log1p()</u>	Returns the natural logarithm of 1+x
<u>math.log2()</u>	Returns the base-2 logarithm of x
<u>math.perm()</u>	Returns the number of ways to choose k items from n items with order and without repetition
<u>math.pow()</u>	Returns the value of x to the power of y
<u>math.prod()</u>	Returns the product of all the elements in an iterable
<u>math.radians()</u>	Converts a degree value into radians



Topic : Modules – datetime, math, numpy

Method	Description
<u>math.isinf()</u>	Checks whether a number is infinite or not
<u>math.isnan()</u>	Checks whether a value is NaN (not a number) or not
<u>math.isqrt()</u>	Rounds a square root number downwards to the nearest integer
<u>math.ldexp()</u>	Returns the inverse of <u>math.frexp()</u> which is $x * (2^{**}i)$ of the given numbers x and i
<u>math.lgamma()</u>	Returns the log gamma value of x
<u>math.log()</u>	Returns the natural logarithm of a number, or the logarithm of number to base
<u>math.log10()</u>	Returns the base-10 logarithm of x
<u>math.log1p()</u>	Returns the natural logarithm of 1+x
<u>math.log2()</u>	Returns the base-2 logarithm of x
<u>math.perm()</u>	Returns the number of ways to choose k items from n items with order and without repetition
<u>math.pow()</u>	Returns the value of x to the power of y
<u>math.prod()</u>	Returns the product of all the elements in an iterable
<u>math.radians()</u>	Converts a degree value into radians



Topic : Modules – datetime, math, numpy

Constant	Description
<u>math.e</u>	Returns Euler's number (2.7182...)
<u>math.inf</u>	Returns a floating-point positive infinity
<u>math.nan</u>	Returns a floating-point NaN (Not a Number) value
<u>math.pi</u>	Returns PI (3.1415...)
<u>math.tau</u>	Returns tau (6.2831...)



Topic : Modules – datetime, math, numpy

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.



2 mins Summary



- Method overloading
- method overriding
- Constructors
- Access Modifiers
- modules - datetime
 - math*, numpy* (please refer the document)

To be contd... 😊



THANK - YOU