Introduction to PYTHON

Module 1 / Lecture-5

By: Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,

GLA University, Mathura

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html

abs(x)

- Returns the absolute value of a number

```
>>>a = -10
>>>abs(a)
10
>>>b = 2+3j
>>>abs(b)
3.6055512754639896
```

all(*iterable*)

- Return True
  - if all elements of the *iterable* are true
  - if the iterable is empty

>>>all([1,2,3])

True

>>>all([])

True

any(*iterable*)

- Return **True**
  - if **any element** of the *iterable* is **True**.
- Return **False**.
  - If the iterable is **empty**,

```
>>>any([1, 2, 3])
True
>>>any([])
False
```

bin(*x*)

- Convert an integer number to a binary string prefixed with “0b”.

```
>>>bin(255)
'0b11111111'
```

chr(*i*)

- Return the string representing a character whose Unicode code point is the integer *i*.

>>>chr(97)

'a‘

>>>chr(65)

‘A’

complex([*real*[, *imag*]]**)**

- Return a complex number with the value *real* + *imag**1j or convert a string or number to a complex number

>>>complex(2,3)
(2+3j)

>>>complex()
0j

>>>complex(5)
5+0j

dict(*iterable, **kwarg***)**

- Create a new dictionary.

```
>>>dict(a=1, b=2, c=3, d=4)
{'a': 1, 'b': 2, 'c': 3, 'd': 4}

>>>dict([ ('a', 1), ('b', 2), ('c',3) ] )
{'a': 1, 'b': 2, 'c': 3}

>>>dict({1:11, 2:22, 3:33}, x = 44, y = 55, z = 66)
{1: 11, 2: 22, 3: 33, 'x': 44, 'y': 55, 'z': 66}
```

enumerate(*iterable*, *start=0***)**

- Return an enumerate object.
- *iterable* must be a sequence, an iterator, or some other object which supports iteration.

```
>>>b=enumerate([10,20,30], start=1)
>>>list(b)
[(1, 10), (2, 20), (3, 30)]
```

float(*x*)

- Return a floating point number constructed from a number or string *x*.

```
>>>float('11.2')
11.2
>>>float(11)
11.0
```

hex(*x*)

- Convert an integer number to a lowercase hexadecimal string prefixed with “0x”.

>>>hex(10)

‘0xa’

id(*object*)

- Return the “identity” of an object.
- This is an integer which is guaranteed to be
  - unique and
  - constant

  for this object during its lifetime.

>>>a = 10

>>>b = 10

>>>id(a)

1467373440

>>>id(b)

1467373440

int(x**)**

- Return an integer object constructed from a number or string *x*, or return 0 if no arguments are given.

>>>**int(**‘12’**)**

12

isinstance(*object*, *classinfo***)**

- Return True if the *object* argument is an instance of the *classinfo* argument.

>>>isinstance(12, int**)**

True

len(*s*)

- Return the length (the number of items) of an object.
- **S** may be
  - a sequence
    - String
    - tuple
    - list
    - range
  - a collection
    - dictionary
    - set

max(*iterable***)**

- Return the largest item in an iterable or the largest of two or more arguments.

```
>>>max(1, 22, 4, 55, 777)
777
```

min(*iterable***)**

- Return the smallest item in an iterable or the smallest of two or more arguments.

```
>>>min(1, 22, 4, 55, 777)
1
```

oct(*x*)

- Convert an integer number to an octal string prefixed with “0o”.

```
>>>oct(8)
'0o10'

>>> oct(-56)
'-0o70'
```

ord(*c*)

- Given a string representing one Unicode character, return an integer representing the Unicode code point of that character.

```
>>>ord('a')
97
```

pow(*base*, *exp*[, *mod*]**)**

- Return *base* to the power *exp*;
- if *mod* is present,
- **pow(base, exp) % mod)**

- **pow(base, exp)** is equivalent to **base**exp**

```
>>>pow(2, 4, 3)
1

>>>pow(2,4)
16
```

range(*start*, *stop*[, *step*]**)**

- Rather than being a function, range is actually an immutable sequence type

>>>list(range(1,11))

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>>list(range(1,11,2))

[1, 3, 5, 7, 9]

reversed(*seq*)

- Return a reverse iterator.

```
>>>list(reversed([1, 2, 3, 40, 5]))
[5, 40, 3, 2, 1]
```

sorted(*iterable*, *key=None*, *reverse=False***)**

- Return a **new sorted list** from the items in *iterable*.
- Has two optional arguments which must be specified as keyword arguments.
  - *key* specifies a function of one argument that is used to extract a comparison key from each element in *iterable*.
  - *reverse* is a boolean value.

```
>>>sorted([1, 20, 3, 40, 5])
[1, 3, 5, 20, 40]

>>>sorted(["aa", "aaa","aaaa","b"], key = len)
['b', 'aa', 'aaa', 'aaaa']

>>>sorted(["aa", "aaa","aaaa","b"])
['aa', 'aaa', 'aaaa', 'b']
```

type(*object*)

- Return the type of an *object*

```
>>>a=10
>>>type(a)
int
```

zip(*iterables***)**

- Make an iterator that aggregates elements from each of the iterables.

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]

>>> zipped = zip(x, y)

>>> list(zipped)
[(1, 4), (2, 5), (3, 6)]
```

Multiple input from user

```
a, b, c = input("enter three values ").split()
enter three values 11 22 33

a,b,c=[int(a), int(b),int(c)]

print(a+b+c)
66
```

Input the values from user in a list

```
L=[int(x) for x in input().split()]
1 33  555  56

print(L)
[1, 33, 555, 56]
```

```
L = []
for i in range(5):
    a = int(input("enter a number. "))
    L.append(a)
```

For more functions or detail visit

https://docs.python.org/3/library/functions.html

Or

>>>**help(**function_name**)**