

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

## SECTION 1: GETTING STARTED

### 1 | How to Install Python

(4:46m)

[Welcome to Python.org](https://www.python.org) -> Downloads -> Download python 3 version.



```
C:\Users\HP>python
Python 3.6.8
Type "help", "copyright", "credits" or
"license" for more information.
>>> ^Z
```

```
C:\Users\HP>python --version
Python 3.6.8
```

After installation is completed, check in cmd.

### 2 | First Basic Program in Python

(7:2m)

Python is an interpreter language, it generates/modifies pyc file(a cache file to avoid checks unless code is changed) whenever there's some code change.

Resource: basics.py

To run in CMD: `C:\Users\HP>python path\to\the\file\basics.py`

### 3 | Python Data Types(Number, String, List, Tuple, Dictionary)

(7:23m)

Immutable -> Number: int, float, long, complex; string: unicode strings; tuple;

Mutable-> list & dictionary.

```
C:\Users\HP>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

Type "help", "copyright", "credits" or "license" for more information.

```
>>> num = 10
>>> type(num)
<class 'int'>
>>> float(num)
10.0
>>> message = "Hello world"
>>> type(message)
<class 'str'>
>>> message = u"Hello world"
>>> type(message)
<class 'str'>
>>> a = 10
>>> a = 1
>>> tup = ('Tom', 'Jerry')
>>> tup
('Tom', 'Jerry')
>>> type(tup)
<class 'tuple'>
>>> tup[0]
'Tom'
>>> tup[1]
'Jerry'
>>> del tup[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
>>> tup[0] = 'Jack'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> list = ['Tom', 'Jerry']
>>> list[0]
'Tom'
>>> list[1]
'Jerry'
>>> list[1] = 'Jack'
>>> list[1]
'Jack'
>>> del list[0]
>>> list
['Jack']
>>> dic1 = {}
>>> dic2 = dict()
>>> type(dic1)
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
<class 'dict'>
>>> dic2["one"] = 1
>>> dic2
{'one': 1}
```

## 4 | Python Operators

(9:39m)

```
C:\Users\HP>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 10
>>> b = 2
>>> #Basic operators
...
>>> a + b
12
>>> a - b
8
>>> a * b
20
>>> a / b
5.0
>>> a // b
5
>>> a / 3
3.3333333333333335
>>> a // 3
3
>>> a % b
0
>>> a ** b
100
>>> #Assignment operators
...
>>> a += b
>>> a
12
>>> a -= b
>>> a
10
>>> a /= b
>>> a
5.0
>>> a // = b
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
>>> a
2.0
>>> a *= b
>>> a
4.0
>>> a %= b
>>> a
0.0
>>> a = 5
>>> a **= b
>>> a
25
>>> #Conditional operators
...
>>> a == b
False
>>> a < b
False
>>> a > b
True
>>> a != b
True
>>> #Logical operators
...
>>> t = True
>>> f = False
>>> t & f
False
>>> t and f
False
>>> t | f
True
>>> t or f
True
>>> #Identity operator
...
>>> a is 25
True
>>> a is not 25
False
>>> #Membership operator
...
>>> list = [1,3,5,7,9]
>>> 1 in list
True
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
>>> 1 not in list
False
```

## 5 | Python Numbers and Methods

(11:39m)

Install IPython: *pip install ipython*

IPython is a python shell that provides all available functions with a value on *TAB* keypress or *?*.

Use *exit* to exit ipython shell

```
C:\Users\HP>ipython
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.3 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: a = 10
```

```
In [2]: type(a)
```

```
Out[2]: int
```

```
In [3]: a = 10.0
```

```
In [4]: type(a)
```

```
Out[4]: float
```

```
In [5]: complex?
```

```
Init signature: complex(self, /, *args, **kwargs)
```

```
Docstring:
```

```
complex(real[, imag]) -> complex number
```

Create a complex number from a real part and an optional imaginary part.

This is equivalent to (real + imag\*1j) where imag defaults to 0.

Type:                   type

Subclasses:

```
In [6]: i = 2
```

```
In [7]: c = complex(a,i)
```

```
In [8]: c
```

```
Out[8]: (10+2j)
```

```
In [9]: c.imag
```

```
Out[9]: 2.0
```

```
In [10]: c.imag
```

```
Out[10]: 2.0
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [11]: c.
```

```
conjugate()  
imag  
real
```

Complex number methods

```
In [11]: float(10)
```

```
Out[11]: 10.0
```

```
In [12]: int(10.0)
```

```
Out[12]: 10
```

```
In [13]: import math
```

```
In [14]: math.trunc(10.01)
```

```
Out[14]: 10
```

```
In [24]: math.
```

acos()	ceil()	erfc()	frexp()	isfinite()	log1p()	sin()
acosh()	copysign()	exp()	fsum()	isinf()	log2()	sinh()
asin()	cos()	expm1()	gamma()	isnan()	modf()	sqrt()
asinh()	cosh()	fabs()	gcd()	ldexp()	nan	tan()
atan()	degrees()	factorial()	hypot()	lgamma()	pi	tanh()
atan2()	e	floor()	inf	log()	pow()	tau
atanh()	erf()	fmod()	isclose()	log10()	radians()	trunc()

```
In [20]: import random
```

```
In [21]: random.choice(['Play', 'Watch TV', 'Sleep'])
```

```
Out[21]: 'Sleep'
```

```
In [22]: random.choice(range(0,100))
```

```
Out[22]: 33
```

```
In [24]: random.
```

betavariate()	gauss()	NV_MAGICCONST	RECIP_BPF	SystemRandom
BPF	getrandbits()	paretovariate()	sample()	triangular()
choice()	getstate()	randint()	seed()	TWOPI
choices()	LOG4	Random	setstate()	uniform()
expovariate()	lognormvariate()	random()	SG_MAGICCONST	vonmisesvariate()
gammavariate()	normalvariate()	randrange()	shuffle()	weibullvariate()

## 6 | Python Strings and Methods

(14:17m)

```
C:\Users\HP>ipython
```

```
In [1]: s = "how are you?"
```

```
In [2]: "how" in s
```

```
Out[2]: True
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [3]: "how" not in s
Out[3]: False

In [5]: l = "*"

In [6]: l*20
Out[6]: '********************'

In [8]: l *=20

In [9]: l
Out[9]: '********************'
```

Escape Characters: \n, \t, \r, \s, etc. Refer official docs.

```
In [10]: p = "new\nline"

In [11]: p
Out[11]: 'new\nline'

In [12]: p.splitlines()
Out[12]: ['new', 'line']
```

Formatting:

```
In [13]: s = "name: %s , age: %d" %("Alice", 10)

In [14]: s
Out[14]: 'name: Alice , age: 10'

In [15]: s = "name is {1}, age is {0}".format(10, "Alice")

In [16]: s
Out[16]: 'name is Alice, age is 10'
```

```
In [17]: s.split()
Out[17]: ['name', 'is', 'Alice,', 'age', 'is', '10']

In [18]: s.split(",")
Out[18]: ['name is Alice', ' age is 10']

In [19]: s.startswith("n")
Out[19]: True
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [21]: s.  
capitalize() encode      format      isalpha      islower      istitle      lower      replace      rpartition  splitlines  title  
casefold      endswith    format_map  isdecimal    isnumeric    isupper     lstrip      rfind       rsplit      startswith  translate  
center        expandtabs  index       isdigit      isprintable  join        maketrans   rindex      rstrip      strip       upper  
count         find        isalnum     isidentifier isspace      ljust       partition   rjust       split       swapcase    zfill
```

## 7 | Python List and Tuple

(11:47m)

List indexing: L->R: 0, 1, .... R->L: -1,-2, ....

```
In [1]: l = ["Tom", 50, "Jerry", 20.5]  
  
In [2]: m = ['Alice', 'apple']  
  
In [3]: l += m  
  
In [4]: l  
Out[4]: ['Tom', 50, 'Jerry', 20.5, 'Alice', 'apple']
```

```
In [5]: len(l)  
Out[5]: 6  
  
In [6]: l.insert(2, '&')  
  
In [7]: l  
Out[7]: ['Tom', 50, '&', 'Jerry', 20.5, 'Alice', 'apple']  
  
In [8]: l.pop(2)  
Out[8]: '&'  
  
In [9]: l  
Out[9]: ['Tom', 50, 'Jerry', 20.5, 'Alice', 'apple']
```

```
In [10]: l.
```

append()	count	insert	reverse
clear	extend	pop	sort
copy	index	remove	

```
In [1]: t = ()  
  
In [2]: t  
Out[2]: ()  
  
In [3]: t = ("tom") #this will define a string  
  
In [4]: t  
Out[4]: 'tom'
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [5]: t = ("tom",) #this will define a tuple
```

```
In [6]: t
```

```
Out[6]: ('tom',)
```

```
In [7]: t = ("sat", "sun")
```

```
In [8]: t = ("fri", "sat", "sun")
```

```
In [9]: t1 = t[1:]
```

```
In [19]: t.
```

```
count() index()
```

```
In [14]: t
```

```
Out[14]: ('tom',)
```

```
In [15]: l = list(t)
```

```
In [16]: l
```

```
Out[16]: ['tom']
```

```
In [17]: t = tuple(l)
```

```
In [18]: t
```

```
Out[18]: ('tom',)
```

## 8 | Slicing in Python

(14:32m)

Slicing can be done on strings, tuples and lists.

```
In [1]: l = [1,3,5,7,11,13]
```

```
In [2]: l[-6::2]
```

```
Out[2]: [1, 5, 11]
```

```
In [3]: l[-1::-2]
```

```
Out[3]: [13, 7, 3]
```

```
In [4]: l[::-1] #to reverse a string
```

```
Out[5]: [13, 11, 7, 5, 3, 1]
```

```
deepcopy
```

```
In [1]: l = [1,2,3,4,5]
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [2]: m = 1

In [3]: m.append(6)

In [4]: 1
Out[4]: [1, 2, 3, 4, 5, 6]

In [5]: m
Out[5]: [1, 2, 3, 4, 5, 6]

In [6]: import copy

In [7]: n = copy.deepcopy(1)

In [8]: n = copy.deepcopy(m)

In [9]: m.append(7)

In [10]: m
Out[10]: [1, 2, 3, 4, 5, 6, 7]

In [11]: n
Out[11]: [1, 2, 3, 4, 5, 6]

In [12]: 1
Out[12]: [1, 2, 3, 4, 5, 6, 7]

In [13]: n.append(8)

In [14]: n
Out[14]: [1, 2, 3, 4, 5, 6, 8]

In [15]: m
Out[15]: [1, 2, 3, 4, 5, 6, 7]
```

## 9 | Python Dictionary

(9:36m)

```
PS C:\Users\HP> ipython

In [1]: d = {"name": "Tom", "animal": "cat"}

In [2]: d
Out[2]: {'name': 'Tom', 'animal': 'cat'}

In [3]: d["name"]
Out[3]: 'Tom'
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [4]: d["age"] #will face error when accessing non-existing keys
-----
KeyError                                Traceback (most recent call last)
<ipython-input-4-351cab8a0992> in <module>
----> 1 d["age"]

KeyError: 'age'

In [5]: d.get("animal")
Out[5]: 'cat'

In [6]: d.get("age", 10)
Out[6]: 10

In [7]: d
Out[7]: {'name': 'Tom', 'animal': 'cat'}

In [8]: d["age"] = 10

In [9]: d
Out[9]: {'name': 'Tom', 'animal': 'cat', 'age': 10}

In [10]: d.setdefault("type", "cartoon")
Out[10]: 'cartoon'

In [11]: d
Out[11]: {'name': 'Tom', 'animal': 'cat', 'age': 10, 'type': 'cartoon'}

In [12]: d.setdefault("gender")

In [13]: d
Out[13]: {'name': 'Tom', 'animal': 'cat', 'age': 10, 'type': 'cartoon', 'gender': None}

In [14]: del d["type"]

In [15]: d
Out[15]: {'name': 'Tom', 'animal': 'cat', 'age': 10, 'gender': None}

In [17]: d["gender"] = "male"

In [18]: d
Out[18]: {'name': 'Tom', 'animal': 'cat', 'age': 10, 'gender': 'male'}

In [20]: d.items()
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
Out[20]: dict_items([('name', 'Tom'), ('animal', 'cat'), ('age', 10), ('gender', 'male')])
```

```
In [21]: m = {}
```

```
In [22]: l = ['a', 'b', 'c']
```

```
In [23]: m = m.fromkeys(l)
```

```
In [24]: m
```

```
Out[24]: {'a': None, 'b': None, 'c': None}
```

```
In [25]: d.update(m)
```

```
In [26]: d
```

```
Out[26]:
```

```
{'name': 'Tom',  
 'animal': 'cat',  
 'age': 10,  
 'gender': 'male',  
 'a': None,  
 'b': None,  
 'c': None}
```

```
In [27]: d.
```

clear()	get()	pop()	update()
copy()	items()	popitem()	values()
fromkeys()	keys()	setdefault()	

## 10 | Conditional and decision making Operators in Python

(14:11m)

Resource: if\_else.py

## 11 | For and While Loops in Python | by Hardik Patel

(13:24m)

Resource: loops.py

## 12 | Functions and Arguments, Lambda function in Python

(13:43m)

Resource: functions.py

## 13 | Modules, import statement, variable scopes in Python

(14:29m)

Resource: calc

local variable has more scope than global variable. global variable can be used anywhere.

## 14 | Python Package, Inbuilt methods

(11m)

`__init__.py` : add this file to create a folder as package

Resource: calc

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

## 15 | Class, Instance, Method concepts using Account application in Python

(20:15m)

Resource: user/user.py

```
...\resources> ipython

In [1]: from user.user import Account

In [2]: a = Account() #creates instance of a class
constructor is called

In [3]: a.accounts
Out[3]: []

In [4]: a.create({})
Provide username

In [5]: a.create({"username": "Zack"})

In [6]: a.accounts
Out[6]: [{'username': 'Zack', 'id': 1}]

In [7]: a.create({"username": "Zack"})
Only one account can be created

In [8]: a.get()
Out[8]: {'username': 'Zack', 'id': 1}

In [9]: a.update(password="123")

In [10]: a.accounts
Out[10]: [{'username': 'Zack', 'id': 1, 'password': '123'}]

In [11]: a.delete()

In [12]: a.accounts
Out[12]: []
```

## 16 | Custom decorators in Python with examples

(20:43m)

Resource: decorators.py

Resource: user/decorators.py

```
...\resources> ipython

In [1]: from user.user import Account

In [2]: a = Account()
constructor is called
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [3]: Account.accounts
Out[3]: []

In [4]: a.create({})
Provide username

In [5]: a.create({"username": "Zack"})

In [6]: Account.accounts
Out[6]: [{'username': 'Zack', 'id': 1}]

In [7]: a.create({"username": "Zack"})
Only one account can be created

In [8]: b = Account()
constructor is called

In [9]: b.create({"username": "Zack"})
Only one account can be created
```

---

**17 | Setup Python virtual environment in Linux** (9:14m)

---

**18 | Setup python virtual environment in Windows** (5:38m)

```
...path> python --version
Python 3.6.8
...path > pip freeze
virtualenv==20.16.5
...path > virtualenv my_env
...path > my_env\Scripts\activate
(my_env) ...path>deactivate
```

---

**19 | Handling Exceptions and Create Custom Exception in Python** (9:2m)

Resource: exceptions.py

---

**20 | File operations in python** (15:48m)

*f = open(filename, mode, encoding)*  
modes: r, w, w+(write&create), a+(append), x(if file not available)

Resource: text.txt

```
In [1]: f = open('text.txt')

In [2]: f.read()
Out[2]: 'my text file\nthis is next line'
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

```
In [3]: f.seek(0)
Out[3]: 0

In [4]: f.read(2)
Out[4]: 'my'

In [5]: f.tell()
Out[5]: 2

In [6]: f.seek(0)
Out[7]: 0

In [8]: print(f.read())
my text file
this is next line

In [9]: f.seek(0)
Out[9]: 0

In [10]: for line in f.readlines():
...:     print(line, end="")
...:
my text file
this is next line

In [11]: f.close()

In [12]: with open('text.txt') as f:
...:     f.read()
...:
#write
In [13]: f = open('text.txt', 'w+', encoding="utf-8")

In [14]: f.write("Hello")
Out[14]: 5

In [15]: f.writelines('\nworld')

In [16]: f.seek(0)
Out[16]: 0

In [17]: print(f.read())
Hello
world
```

# Python Programming for Beginners (version - 3.6)

-Hardik Patel

+++++

In [37]: f.

buffer	detach()	fileno()	line_buffering	newlines	readline()	seekable()	writable()
close()	encoding	flush()	mode	read()	readlines()	tell()	write()
closed	errors	isatty()	name	readable()	seek()	truncate()	writelines()