



Python Programming Tips



Variables

```
# this is a comment
```

```
i = 5      #another comment  
f = 2.0  
string = 'some string'  
l = [1, 2, 3, 'a', 'bcd', f, [1,2,3], (0,1)]  
t = (1, 2, 3)  
s = set([4, 5, 6,])  
d = dict()
```

```
i+f          # 7.0  
i += f       # i = 7.0  
i /= 2       # i = 3.5  
i = 7  
i /= 2       # i = 3  
i = -7  
i /= 2       # i = -4 ← floor  
i ** 2       # 16
```



Variables

```
# this is a comment
```

```
i = 5      #another comment  
f = 2.0  
string = 'some string'  
l = [1, 2, 3, 'a', 'bcd', f, [1,2,3], (0,1)]  
t = (1, 2, 3)  
s = set([4, 5, 6,])  
d = dict()
```

```
i+f          # 7.0  
i += f       # i = 7.0  
i /= 2       # i = 3.5  
i = 7  
i /= 2       # i = 3  
i = -7  
i /= 2       # i = -4 ← floor  
i ** 2       # 16
```

```
str(0.45586423397566611)  
# '0.455864233976'  
`0.45586423397566611`  
# '0.45586423397566611'  
int('4')      # 4  
int(4.0)      # 4  
float(4)      # 4.0  
eval('(3.4, 2)') # (3.4, 2)
```



Variables

```
# this is a comment
```

```
i = 5      #another comment  
f = 2.0  
string = 'some string'  
l = [1, 2, 3, 'a', 'bcd', f, [1,2,3], (0,1)]  
t = (1, 2, 3)  
s = set([4, 5, 6,])  
d = dict()
```

```
i+f          # 7.0  
i += f       # i = 7.0  
i /= 2       # i = 3.5  
i = 7  
i /= 2       # i = 3  
i = -7  
i /= 2       # i = -4 ← floor  
i ** 2       # 16
```

```
str(0.45586423397566611)  
# '0.455864233976'  
`0.45586423397566611`  
# '0.45586423397566611'  
int('4')      # 4  
int(4.0)      # 4  
float(4)      # 4.0  
eval('(3.4, 2)') # (3.4, 2)
```

```
i == -4       # True  
i >= -3      # False  
-5 < i <= -3 # True  
(i > 3) or (-5 < i <= -3)  
(i % 2 == 0) and (-5 < i <= -3)  
i != 3  
not (i == 3)
```



Variables and Functions

```
i = 2
```

```
j = 4
```

```
def foo(j, k = 3, l = 0, m=None):  
    global i  
    print i, j, k, l, m  
    i+=2  
    return j
```

```
print foo(3)  
print i  
print foo(m=5, j='a')
```

```
def foo2(*args, **keywordArgs):  
    for a in args: print a  
    for kw in keywordArgs:  
        print kw, ':', keywordArgs[kw]  
  
print foo2(1,2,3,3,2,1, a='A', b='B')
```



Lambdas

```
def make_incrementor(n):  
    return lambda x: x + n
```

```
f = make_incrementor(42)  
f(0)  
42  
f(1)  
43
```

```
def foo(f, x):  
    return f(x)
```

```
foo(lambda x: x**2, 3)
```



Lists

```
>>> l=range(3)
>>> l*=2
>>> l+=['a','b','c']
>>> l
[0, 1, 2, 0, 1, 2, 'a', 'b', 'c']
>>> l.sort()
>>> l.append('hello')
>>> l
[0, 0, 1, 1, 2, 2, 'a', 'b', 'c', 'hello']
>>> l[4:]
[2, 2, 'a', 'b', 'c', 'hello']
>>> l[:4]
[0, 0, 1, 1]
>>> l[4]
2
>>> l[4:8]
[2, 2, 'a', 'b']
>>> l[4:-1]
[2, 2, 'a', 'b', 'c']
>>> l[-4:-1]
['a', 'b', 'c']
```



Lists

```
>>> l
[0, 0, 1, 1, 2, 2, 'a', 'b', 'c', 'hello']
>>> l.reverse()
>>> l
['hello', 'c', 'b', 'a', 2, 2, 1, 1, 0, 0]
>>> l.remove(2)
>>> l
['hello', 'c', 'b', 'a', 2, 1, 1, 0, 0]
>>> l.insert(1, 'x')
>>> l
['hello', 'x', 'c', 'b', 'a', 2, 1, 1, 0, 0]
>>> l.count(1)
2
>>> l.index(1)
6
>>> l.insert(l.index('x'), 'y')
>>> l
['hello', 'y', 'x', 'c', 'b', 'a', 2, 1, 1, 0, 0]
>>> del(l[1:2])
>>> l
['hello', 'x', 'c', 'b', 'a', 2, 1, 1, 0, 0]
>>> float(sum((l[-5:]))) / len(l[-5:])
```

List as FILO Queue

```
>>> l=[]
>>> l.append(1)
>>> l.append(2)
>>> l.pop()
2
>>> l.pop()
1
```



List as FIFO Queue

```
>>> l=[]
>>> l.append(1)
>>> l.append(2)
>>> l.pop(0)
1
>>> l.pop(0)
2
```



Sets

```
>>> l
['hello', 'x', 'c', 'b', 'a', 2, 1, 1, 0, 0]
>>> s=set(l)
>>> s.union(range(5))
set(['a', 1, 'c', 'b', 4, 0, 2, 3, 'x', 'hello'])
>>> s
set(['a', 1, 'c', 'b', 0, 2, 'x', 'hello'])
>>> s.add(3)
>>> s.intersection(range(5))
set([0, 1, 2, 3])
>>> s.difference(range(5))
set(['a', 'c', 'b', 'x', 'hello'])
>>> for i in s: print i,
a 1 c b 0 2 3 x hello
>>> 1 in s
True
>>> 2 not in s
False
```



Dictionaries

```
>>> d=dict()
>>> d['a']=20
>>> d[2]=10
>>> d[(1,2)]=[1,2]
>>> d
{'a': 20, (1, 2): [1, 2], 2: 10}
>>> d['a']
20
>>> d.has_key((1,2))
True
>>> d.has_key((1,2,3))
False
>>> d.keys()
['a', (1, 2), 2]
```



Strings

```
>>> s='abc'+ 'def'
>>> s
'abcdef'
>>> s*=2
>>> s
'abcdefabcdef'
>>> s[2]
'c'
>>> s[-5:]
'bcdef'
>>> s+=' words words words ::'
>>> s
'abcdefabcdef words words words ::'
>>> s.strip(':')
'abcdefabcdef words words words '
>>> s
'abcdefabcdef words words words ::'
>>> s.strip(':').split()
['abcdefabcdef', 'words', 'words', 'words']
>>> s.strip(':').split('w')
['abcdefabcdef ', 'ords ', 'ords ', 'ords ']
>>> s=''the way out
... is to keep typing
... something in "quotes"
... and maybe some ' ' ' or whatever...''
>>> s.split()
['the', 'way', 'out', 'is', 'to', 'keep', 'typing',
'something', 'in', '"quotes"', 'and', 'maybe', 'some',
'""', '""', '""', 'or', 'whatever...']
```

Strings

```
>>> s
'the way out\nis to keep typing\nsomething in
"quotes"\nand maybe some \' \' \' or whatever...'
```

```
>>> s.startswith('the')
True
```

```
>>> s.find('keep')
18
```

```
>>> s.rfind('e')
82
```

```
>>> s.replace('e', '3')
'th3 way out\nis to k33p typing\nsom3thing in
"quot3s"\nand mayb3 som3 \' \' \' or what3v3r...'
```

```
>>> s
'the way out\nis to keep typing\nsomething in
"quotes"\nand maybe some \' \' \' or whatever...'
```



Classes

```
class point: x=y=z=0          #define class

p1=point()                    #instance 1
p1.x, p1.y, p1.z = 3, 2, 1    #instance 1 x,y,z initiated

point.x=100                    #class changed
print p1.x                    #will still print 3

p2=point()                    #instance 2 created
p2.y, p2.z = 22, 11          #instance 2 y,z initiated

print p2.x, p2.y, p2.z       #will print "(100, 22, 11)"
```



Classes

```
class point:    #rewriting the class to include functions
    def __init__(self, x, y, z):
        self.x, self.y, self.z = x, y, z
    def p(self):
        return self.x, self.y, self.z
```

```
p1=point(24, 32, 4)
p2=point(3, 2, 1)
p1.p()
p2.p()
```

```
class point4d(point):    #example of inheritance
    t=0
    def p(self):
        return self.x, self.y, self.z, self.t
```



for

```
>>> l
['hello', 'x', 'c', 'b', 'a', 2, 1, 1, 0, 0]
>>> for i in l:
...     print i,
...
hello x c b a 2 1 1 0 0
>>> for i,j in enumerate(l):
...     print i,j
...
0 hello
1 x
2 c
3 b
4 a
5 2
6 1
7 1
8 0
9 0
>>> ['x'+str(i)+'x' for i in l]
['xhellox', 'xxx', 'xcx', 'xbx', 'xax', 'x2x', 'x1x',
'x1x', 'x0x', 'x0x']
```



while

```
>>> i=0
>>> while i<20:
...     print i,
...     i+=1
...
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
>>> while True:
...     print i,
...     i+=1
...     if i>20: break
...
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```



file

```
>>> f=file('tmp.txt','w')
>>> f.write('hello\n')
>>> f.close()
>>> f=file('tmp.txt')
>>> d=f.read()
>>> f.close()
>>> d
'hello\n'
```



url

```
>>> from urllib2 import urlopen
>>> d=urllib.urlopen('http://craigslist.org').read()
>>> d[:200]
'<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">\n<html>\n<head>\n<title>craigslist
classifieds: jobs, housing, personals, for sale,
services, commun'
```



catching errors

```
import sys

try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except IOError as (errno, strerror):
    print "I/O error({0}): {1}".format(errno, strerror)
except ValueError:
    print "Could not convert data to an integer."
except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
```



random

```
import random
random.random()
random.gauss(0,1)
random.randrange(5,10)
l=range(10)
random.sample(l, 5)
random.shuffle(l)
random.choice(l)
```



numpy

```
import numpy
a=numpy.zeros((4,5))
>>> a
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]])
>>> numpy.random.rand(4,5)
array([[ 0.12364675,  0.48592078,  0.84557778,  0.17855823,  0.55675879],
       [ 0.91936541,  0.80503347,  0.95564469,  0.84484192,  0.86495162],
       [ 0.60533778,  0.11077146,  0.54446113,  0.70351497,  0.51997724],
       [ 0.87754571,  0.19586059,  0.80485262,  0.14922077,  0.11180152]])
a=numpy.ones((4,5))
a=numpy.zeros((4,5))-1
a-1
a*2
a[0]+a[1]
a[0].mean()
a[0].std()
```

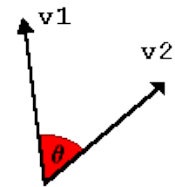


Vector Cosine

Length = SquareRoot($x*x + y*y + z*z$)

DotProduct = ($x1*x2 + y1*y2 + z1*z2$)

Cos(theta) = DotProduct(v1,v2) / (length(v1) * length(v2))



```
import numpy
def cosv(v1,v2):
    return numpy.dot(v1,v2) / ((numpy.dot(v1,v1)*numpy.dot(v2,v2))**.5)
```

```
a1=numpy.array([10,0])
```

```
a2=numpy.array([5,5])
```

```
cosv(a1,a2) #will be 0.70710678118654746
```

```
a3=numpy.array([10,0,10])
```

```
a4=numpy.array([10,0,0])
```

```
cosv(a3,a4) #will be 0.70710678118654746
```



MySQLdb

```
import MySQLdb
db = MySQLdb.connect("localhost", "mydb", "root", "")
cursor = db.cursor()

def sql_fetch(s):
    cursor.execute(s)
    r=cursor.fetchall()
    return r

sql_fetch("show databases")

cursor.execute("select * from mytable")
rone = cursor.fetchone()      #will fetch one record
rlst = cursor.fetchall()      #will fetch all records
rlst = cursor.fetchmany(5)     #will fetch 5 records
```



SQL

```
select * from mytable;
```

```
select field1, field2 from mytable;
```

```
select * from mytable where field1 = 1;
```

```
select count(0) from mytable where field1=1;
```

```
select count(0) from mytable group by field2;  
select avg(field1) from mytable group by field2;
```

```
select t1.id,t1.name,t2.birthdate  
    from mytable as t1, mytable2 as t2  
    where t1.id = t2.id
```

```
delete from mytable [where field1=x]  
update mytable set field1=0 where field1<.5  
insert into mytable (field1, field2)  
    values (1,2);
```



MySQLdb

```
import MySQLdb
db = MySQLdb.connect("localhost", "root", "", "myDB")
cursor = db.cursor()

cursor.executemany(
    """INSERT INTO breakfast (name, spam, eggs, sausage, price)
VALUES (%s, %s, %s, %s, %s)""",
    [
        ("Spam and Sausage Lover's Plate", 5, 1, 8, 7.95 ),
        ("Not So Much Spam Plate", 3, 2, 0, 3.95 ),
        ("Don't Wany ANY SPAM! Plate", 0, 4, 3, 5.95 )
    ] )
```



SQL

- ▶ **Primary key**
 - ▶ proper indexing
 - ▶ secondary indexes
- ▶ **Data types**
 - ▶ int, bigint, smallint
 - ▶ float(d,x)
 - ▶ char(numberofchars)
 - ▶ varchar(max)
 - ▶ text

