```
In [1]:  a = 4
          b = a
          a += 1
          print (b)

          4

In [2]:  a = [1,2,3]
          b = a
          a.append(4)
          print (b)

          [1, 2, 3, 4]

In [3]:  a = [1,2,3]
          b = [1,2,3]
          c = a

In [4]:  # a.append(4)
          a == b

Out[4]:  True

In [5]:  a == c

Out[5]:  True

In [6]:  a is b

Out[6]:  False

In [7]:  a is c

Out[7]:  True

In [11]: a = [1,2,3]
          b = list(a)

In [9]:  a is b

Out[9]:  False

In [12]: b = [x for x in a]

In [13]: import copy

In [15]: b = copy.copy(a)

In [16]: def f(x):
              x += 1

In [17]: a = 3
          f(a)
          print (a)

          3
```

```
In [18]: a = [1,2]
         def f(x):
             x.append(3)
         f(a)
         print (a)

         [1, 2, 3]
```

```
In [19]: a = [1,2,3]
         b = [x+1 for x in a]
         print (b)

         [2, 3, 4]
```

```
In [20]: def f(x):
             return x+1
         a = [1,2,3]
         b = list(map(f, a))
         print (b)

         [2, 3, 4]
```

```
In [21]: a = 3+4
```

```
In [22]: 3+4 = a

           Input In [22]
             3+4 = a
             ^
         SyntaxError: cannot assign to operator
```

```
In [23]: a = 3
         if a < 4:
             b = 6
         else:
             b = 10
         print (b)

         6
```

```
In [24]: b = 6 if a<4 else 10
         print (b)

         6
```

```
In [26]: b = (6 if a<4 else 10) + 10
         print (b)

         16
```

```
In [27]: b = 3 + 4
```

```
In [28]: True and False

Out[28]: False
```

```
In [29]: not False

Out[29]: True
```

```
In [30]: -8

Out[30]: -8
```

In [31]:
```
+77
```
Out[31]:
```
77
```
In [32]:
```
bmi = 40
if bmi < 20:
    diag = 'leptos'
elif bmi < 30:
    diag = 'normal'
else:
    diag = 'fat'
```
In [33]:
```
bmi = 40
if bmi < 20:
    diag = 'leptos'
else:
    if bmi < 30:
        diag = 'normal'
    else:
        diag = 'fat'
```
In [34]:
```
bmi = 40
if bmi < 20:
    diag = 'leptos'
else:
    diag = 'normal' if bmi < 30 else 'fat'
```
In [35]:
```
diag = 'leptos' if bmi < 20 else ('normal' if bmi < 30 else 'fat')
```
In [37]:
```
a = [1,2,3]
b = []
for x in a:
    b.append(x+1)
print (b)
```
```
[2, 3, 4]
```
In [38]:
```
a = [1,2,3]
b = [x+1 for x in a]
print (b)
```
```
[2, 3, 4]
```
In [40]:
```
def f(x):
    return x+1
a = [1,2,3]
b= list(map(f, a))
print (b)
```
```
[2, 3, 4]
```
In [41]:
```
def f(x):
    return x+1
```
In [42]:
```
f = lambda x : x+1
```
In [43]:
```
def f(a,b):
    return a+b
```

```
In [44]: f = lambda a,b : a+b

In [45]: f(6,7)

Out[45]: 13

In [47]: a = [1,2,3]
         b = list(map(lambda x : x+1, a))
         print (b)

         [2, 3, 4]

In [48]: a = ['sdfgsdfg', 'sdfgsdfgsadfg', 'shrtytyt4653gg']

In [49]: def f(x):
             return x.count('s')
         b = sorted(a, key=f)
         print (b)

         ['shrtytyt4653gg', 'sdfgsdfg', 'sdfgsdfgsadfg']

In [50]: b = sorted(a, key=lambda x : x.count('s'))
         print (b)

         ['shrtytyt4653gg', 'sdfgsdfg', 'sdfgsdfgsadfg']

In [51]: f = lambda x : x+1

Out[51]: 5

In [52]: (lambda x : x+2)(3)

Out[52]: 5

In [53]: f = lambda x : x+1

In [54]: callable(f)

Out[54]: True

In [55]: type(f)

Out[55]: function

In [61]: def f(a):
             def g(b):
                 return a+b
             return g

In [59]: aa = f(3)

In [60]: aa(6)

Out[60]: 9

In [62]: f(3)(6)

Out[62]: 9
```

```
In [ ]: def f(a):
            def g(b):
                return a+b
            return g
```

```
In [63]: def f(a):
             return lambda b:a+b
```

```
In [64]: f = lambda a: lambda b:a+b
```

```
In [65]: f(3)(6)

Out[65]: 9
```

Generators

```
In [73]: def f(x):
             return x+1

         b = map(f, [1,2,3])
         #print (list(b))
```

```
In [ ]:
```

```
In [67]: b

Out[67]: <map at 0x1cf94c19700>
```

```
In [68]: next(b)

Out[68]: 2
```

```
In [69]: next(b)

Out[69]: 3
```

```
In [70]: next(b)

Out[70]: 4
```

```
In [71]: next(b)

---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Input In [71], in <cell line: 1>()
----> 1 next(b)

StopIteration:
```

```
In [75]: def f():
             return 3
             return 4
             return 5
```

```
In [76]: a = f()
         print (a)
```

```
3
```

In [84]:
```python
def f():
    yield 3
    yield 4
    yield 5
    yield 6
```

In [79]:
```python
a  = f()
print (a)
```

```
<generator object f at 0x000001CF96762350>
```

In [80]:
```python
next(a)
```

Out[80]:
```
3
```

In [81]:
```python
next(a)
```

Out[81]:
```
4
```

In [82]:
```python
next(a)
```

Out[82]:
```
5
```

In [85]:
```python
a = f()
```

In [86]:
```python
type(a)
```

Out[86]:
```
generator
```

In [89]:
```python
def f():
    yield 3
    yield 4
    yield 5
    yield 6
```

In [90]:
```python
a = f()
```

In [91]:
```python
next(a)
```

Out[91]:
```
3
```

In [92]:
```python
next(a)
```

Out[92]:
```
4
```

In [93]:
```python
list(a)
```

Out[93]:
```
[5, 6]
```

In [94]:
```python
def f():
    yield 3
    yield 4
    yield 5
    yield 6
```

In [95]:
```
for x in f():
    print (x)
```
```
3
4
5
6
```

In [104…
```
def f():
    yield 3
    yield 4
    yield 5
    yield 6
```

In [97]:
```
a = f()
next(a)
next(a)
for x in a:
    print (x)
```
```
5
6
```

In [98]:
```
type(f)
```
Out[98]:
```
function
```

In [99]:
```
g = f()
```

In [100…
```
type(g)
```
Out[100…
```
generator
```

In [107…
```
a = [2,3,4]
b = [x+1 for x in a]
print (b)
```
```
[3, 4, 5]
```

In [110…
```
def f():
    for x in a:
        yield x+1
```

In [111…
```
g = f()
```

In [112…
```
next(g)
```
Out[112…
```
3
```

In [114…
```
next(g)
```
Out[114…
```
4
```

In [115…
```
next(g)
```
Out[115…
```
5
```

In [116…
```
a = [2,3,4]
b = (x+1 for x in a)
```

```
In [117…  type(b)

Out[117…  generator

In [118…  next(b)

Out[118…  3

In [128…  a = range(1, 100_000_000_000)

In [129…  b = map(lambda x: x+3, a)

In [130…  c = filter(lambda x:x%2==1, b)

In [131…  d = (int(str(x)[-1]) for x in c)

In [132…  next(d)

Out[132…  5

In [133…  next(d)

Out[133…  7

In [134…  next(d)

Out[134…  9

In [135…  next(d)

Out[135…  1

In [136…  !dir

 Volume in drive C has no label.
 Volume Serial Number is 2E18-E674

 Directory of C:\Users\user

06/04/2022  05:25 ££    <DIR>          .
06/04/2022  05:25 ££    <DIR>          ..
06/04/2022  03:49 ££    <DIR>          .ipynb_checkpoints
30/03/2022  03:41 ££    <DIR>          .ipython
06/04/2022  03:49 ££    <DIR>          .jupyter
24/02/2022  10:57 §£    <DIR>          3D Objects
30/03/2022  05:00 ££            12.313 alex.docx
30/03/2022  05:29 ££               391 alex2.txt
24/02/2022  10:57 §£    <DIR>          Contacts
06/04/2022  01:03 ££    <DIR>          Desktop
30/03/2022  05:00 ££    <DIR>          Documents
30/03/2022  03:45 ££    <DIR>          Downloads
24/02/2022  10:57 §£    <DIR>          Favorites
30/03/2022  05:37 ££                54 findings.txt
24/02/2022  10:57 §£    <DIR>          Links
30/03/2022  03:38 ££    <DIR>          miniconda3
30/03/2022  03:43 ££                19 mitsos.txt
24/02/2022  10:57 §£    <DIR>          Music
24/02/2022  12:53 ££    <DIR>          OneDrive
24/02/2022  10:58 §£    <DIR>          Pictures
```

```
30/03/2022  04:58 ££              488 results.txt
24/02/2022  10:57 §£    <DIR>          Saved Games
24/02/2022  10:58 §£    <DIR>          Searches
06/04/2022  02:51 ££    <DIR>          teaching_VIOT_I
30/03/2022  06:16 ££           40.623 test_1.ipynb
06/04/2022  05:25 ££           29.852 Untitled.ipynb
04/03/2022  01:49 ££    <DIR>          Videos
               7 File(s)         83.740 bytes
              20 Dir(s)  198.915.809.280 bytes free
```

In [137…
```
!type results.txt
```

```
this is a fantastic file
very precious data
much science. bravo!
nobel


dyjfjfghjfkhjthjhkhkhjfgfgmgdd.fgjdalkgjshdlfkgjhsdlkfjghsldkfjghsldkjghskl
dfjghlsdkfjhglskdfjghlskdfjghskldjfghsldkfjghsldkjgh sljgsldkj ghskldjgh sl
dkgjh skldfjgh skldfjghsldkjghsldkfjgh skldjg hsldfkjghsldkjg hsldfkjgh sld
kjgh sldkgjh sldkjg hsdlfkjgh sdfkjg hsldkfjg hsldkfjg hsldkfjg hsldkfjg hs
ldkfjgh sldkfjg sldkfjgh sldkjgh sldfjgh sldkfjgh sldkfjgh sldkfjg hsldkfjg
hsldkfjgh sldkfjgh sldkfjgh fjkldh

aaa
```

In [138…
```
f = open('results.txt')
```

In [139…
```
next(f)
```

Out[139…
```
'this is a fantastic file\n'
```

In [ ]:
```

```

In [125…
```
next(c)
```

Out[125…
```
5
```

In [126…
```
next(c)
```

Out[126…
```
7
```

In [127…
```
# list(c)
```

In [151…
```
%%writefile a.txt
123
234
546
567
687
```

```
Overwriting a.txt
```

In [154…
```
def f():
    with open('a.txt') as f:
        for l in f:
            if int(l)%2==1:
                yield l.strip()
```

```
In [155…  g = f()
          for x in range(2):
              b = next(g)
              print (b)

          123
          567

In [ ]:

In [144…  g = f()

In [145…  next(g)

Out[145…  '123'

In [146…  next(g)

Out[146…  '567'

In [147…  next(g)

Out[147…  '687'

In [148…  for x in g:
              print (x)

In [156…  import antigravity

In [157…  import this

          The Zen of Python, by Tim Peters

          Beautiful is better than ugly.
          Explicit is better than implicit.
          Simple is better than complex.
          Complex is better than complicated.
          Flat is better than nested.
          Sparse is better than dense.
          Readability counts.
          Special cases aren't special enough to break the rules.
          Although practicality beats purity.
          Errors should never pass silently.
          Unless explicitly silenced.
          In the face of ambiguity, refuse the temptation to guess.
          There should be one-- and preferably only one --obvious way to do it.
          Although that way may not be obvious at first unless you're Dutch.
          Now is better than never.
          Although never is often better than *right* now.
          If the implementation is hard to explain, it's a bad idea.
          If the implementation is easy to explain, it may be a good idea.
          Namespaces are one honking great idea -- let's do more of those!

In [ ]:

In [158…  from my_fabulous_code import f # 1

In [159…  f()

          hello
```

```
In [1]: from my_fabulous_code import * # IOUOUOU # 3
```

```
In [2]: a
```

```
Out[2]: 'biolohgy'
```

```
In [3]: f()
```

```
hello
```

```
In [1]: import my_fabulous_code # 2
```

```
In [2]: my_fabulous_code.a
```

```
Out[2]: 'biolohgy'
```

```
In [3]: my_fabulous_code.f()
```

```
hello
```

```
In [4]: import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```
In [1]: from kostas.mitsos.test import g
```

```
In [2]: g()
```

```
sdfgsdfg
```

```
In [3]: from collections import Counter
```

```
In [4]: Counter('sdklfjghsdlkjghsldkjghsldkjghsldkfjghsldkjghsldkghsldkfjghskldath
```

```
Out[4]: Counter({'s': 9,
                 'd': 9,
                 'k': 9,
                 'l': 10,
                 'f': 3,
                 'j': 7,
                 'g': 9,
                 'h': 9,
              'a': 1,
              't': 1,
              'q': 1,
              'r': 1,
              'u': 1,
              'i': 1,
              'o': 1,
              'e': 1})
```

```
In [5]: Counter([1,2,3,2,3,4,3,4,6,4,7,8,9])

Out[5]: Counter({1: 1, 2: 2, 3: 3, 4: 3, 6: 1, 7: 1, 8: 1, 9: 1})

In [6]: a = Counter('faklrutsleaiuhsldkfjghsldkfjghsdkfjghsldjkheailughsakldjhsldk

In [7]: b = Counter('sd;kfgjhlaskeryawliefhLAJFGHD;lawrieyueailfj:OAfghsealghsoeai

In [8]: a

Out[8]: Counter({'f': 5,
                 'a': 4,
                 'k': 7,
                 'l': 8,
                 'r': 1,
                 'u': 3,
                 't': 1,
                 's': 7,
                 'e': 2,
                 'i': 2,
                 'h': 8,
                 'd': 6,
                 'j': 6,
                 'g': 5})

In [9]: b

Out[9]: Counter({'s': 4,
                 'd': 1,
                 ';': 2,
                 'k': 2,
                 'f': 4,
                 'g': 4,
                 'j': 2,
                 'h': 5,
                 'l': 5,
                 'a': 6,
                 'e': 6,
                 'r': 2,
                 'y': 2,
                 'w': 2,
                 'i': 4,
                 'L': 1,
                 'A': 2,
                 'J': 1,
                 'F': 1,
                 'G': 1,
                 'H': 1,
                 'D': 1,
                 'u': 2,
                 ':': 1,
                 'O': 1,
                 'o': 1})
```

```
In [10]: a + b
```

```
Out[10]: Counter({'f': 9,
                  'a': 10,
                  'k': 9,
                  'l': 13,
                  'r': 3,
                  'u': 5,
                  't': 1,
                  's': 11,
                  'e': 8,
                  'i': 6,
                  'h': 13,
                  'd': 7,
                  'j': 8,
                  'g': 9,
                  ';': 2,
                  'y': 2,
                  'w': 2,
                  'L': 1,
                  'A': 2,
                  'J': 1,
                  'F': 1,
                  'G': 1,
                  'H': 1,
                  'D': 1,
                  ':': 1,
                  'O': 1,
                  'o': 1})
```

```
In [11]: from collections import defaultdict
```

```
In [12]: a = {}
```

```
In [13]: print (a['mitsos'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [13], in <cell line: 1>()
----> 1 print (a['mitsos'])

KeyError: 'mitsos'
```

```
In [15]: a = defaultdict(int)
```

```
In [16]: print (a['mitsos'])
```

```
0
```

```
In [17]: a = 'aljrgajkfajkldfgskdaghasdljghsdaklfghkldafhsdkljghsklghskldfghskldfjgl
```

```
In [22]: b = {}
         for x in a:
             if not x in b:
                 b[x] = 0
             b[x] += 1
```

In [25]:
```
b = defaultdict(int)
for x in a:

    #b[x] += 1
    b[x] = b[x] + 1
```

In [26]:
```
b = defaultdict(list)

for x in [(1, 'a'), (2,'b'), (1, 'c')]:

    b[x[0]].append(x[1])

print (b)
```

```
defaultdict(<class 'list'>, {1: ['a', 'c'], 2: ['b']})
```

In [27]:
```
b = {}

for x in [(1, 'a'), (2,'b'), (1, 'c')]:

    if not x[0] in b:
        b[x[0]] = []

    b[x[0]].append(x[1])

print (b)
```

```
{1: ['a', 'c'], 2: ['b']}
```

In [24]:
```
b
```

Out[24]:
```
defaultdict(int,
            {'a': 7,
             'l': 10,
             'j': 6,
             'r': 1,
             'g': 10,
             'k': 10,
             'f': 6,
             'd': 9,
             's': 8,
             'h': 9})
```

In [28]:
```
import random
```

In [29]:
```
random.random()
```

Out[29]:
```
0.3170388002120018
```

In [53]:
```
random.randint(18,20)
```

Out[53]:
```
20
```

In [71]:
```
counter = 0
N = 1_000_000
for y in range(N):
    if [random.randint(1, 6) for x in range(10)].count(6) == 3:
        counter += 1
print (counter/N)
```

```
0.155197
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: