Блоки кода

- Блоки ограничивают участок кода, принадлежащий управляющей конструкции

- Начинаются с ":", которым оканчивается конструкция

- Все строки блока имеют уровень отступа равным начальной строке блока

- Отступы делаются с помошью табуляции или пробелов

- Блоки могут содержать другие блоки (с более глубокими отступами)

```
1    Some_contruction:
2        y = 2
3        z = x + y
4    #end_of_block
```

Блоки кода

- Блоки это не области видимости переменных. Переменные видны и после выхода из блока

- pass – пустой блок

```
1       Some_contruction:
2           y = 2
3       print y # ok
```

if - Условное выполнение участков кода

```python
1    if   condition1 :
2        pass # excuted if condition1 is true
3    elif condition2 :
4        pass # excuted if condition1 is false and condition2 is
5    #...
6    else:
7        pass # executed if all conditions is false
```

```
1    x = 12
2    sign = 0
3    if x > 0:
4        print x, "positive"
5        sign = 1
6    elif x < 0:
7        print x, "negative"
8        sign = -1
9    else:
10       print x, "== 0"
11       sign = 0
```

inline if

```
1    res = x if x >= 0 else -x
2    # res = (x >= 0 ? x : -x)
```

```python
while condition:
    pass # executed while condition is true
else:
    pass # if no exception or break in body

x = 1
while x < 100:
    print x, "less than 100"
    x *= 2

while x < 100:
    if x ** 2 == 13:
        break  # found square root of 13
    x += 1
else:
    x = None
    print "13 have no integer square root"
```

for - цикл по множеству

```
1    for x in iterable:
2        func(x) # for each element in iterable
3    else:
4        pass # if no exception or break in body
5
6    sum = 0
7    for x in range(100):
8        sum += x
9    print x  # 99 * 100 / 2
10
11   for i in range(n): # xrange(n)
12       pass
```

break & continue как всегда

- break выходит из цикла

- continue переходит к следующей итерации

Нет

- goto

- switch + case PEP-3103

- until

- do + while, do + until

with

```
1    with expr1 as var1, expr2 as var2, ...:
2        block
3
4    with expression as var:
5        block
```

```
1    with expression as var:
2            block
3
4    var = expression
5    var.__enter__()
6    try:
7            block
8    finally:
9            var.__exit__(exception_info)
```

использование with

```
1   with open(r"C:\xxx.bin", "w") as fd:
2       fd.write("-" * 100 + "\n")
3       fd.write("+" * 100 + "\n")
4
5   with open(r"C:\xxx.bin", "r") as fd:
6       for line in fd:
7           print line
8
9   with db.cursor() as cur:
10      curr.execute(update_request_1)
11      curr.execute(update_request_2)
12      # commit or rollback
```

List comprehension

• Фильтрация и преобразование элементов контейнера

```
1    res = [func(i) for i in some_iter if func2(i)]
2
3    data = [-1, -2, -3, 1 , 2, 3, 4]
4
5    print [x ** 0.5 for x in data if x >= 0]
6
7    # [1.0, 1.41.., 1.73.., 2.0]
8
9    res = {func(i) for i in some_iter if func2(i)}
```

List comprehension

```
1    # set
2    res = {x ** 0.5 for x in data if x >= 0}
3
4    # dict
5    res = {x:x ** 0.5 for x in data if x >= 0}
6
7    # generator
8    res = (x ** 0.5 for x in xrange(1000000) if x >= 0)
```

Функции - минимум

```python
1    def func_name1(param1, param2):
2        "documentation"
3        # block
4        x = param1 + param2
5        return x
6
7    def func_name2(param1, param2):
8        "documentation"
9        # block
10       x = param1 + param2
11       if x > 0:
12           return x
13       else:
14           return 0
```

Unit tests - find

```
1    assert find("abc", "b") == 1
2    assert find("abc", "b") == "abc".find("b")
3
4    assert find("abc", "a") == 0
5    assert find("abca", "a") == 0
6    assert find("dabca", "a") == 1
7    assert find("", "a") == -1
8    assert find("a", "a") == 0
9    assert find("ab", "abc") == 0
10   assert find("b" * 1000 + "abc", "abc") == 1000
11   assert find("b" * 1000 + "abc", "abcd") == -1
12
13   all_symbols = "".join([chr(i) for i in range(255)])
14   assert find(all_symbols, chr(100)) == 100
15
16   assert find("", "") == 0
17   assert find("", "") == "".find("")
```

Program template

```python
1      #!/usr/bin/end python
2      # -*- coding:utf8 -*-
3      ......
4
5      def main():
6          res = 0
7          .....
8          return res
9
10     if __name__ == "__main__":
11         exit(main())
```