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

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
```

```python
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)
```

```
1    while condition:
2        pass # executed while condition is true
3    else:
4        pass # if no error or break in body
5
6    x = 1
7    while x < 100:
8        print x, "less than 100"
9        x *= 2
```

for - цикл по множеству

```python
1    for x in iterable:
2        func(x) # for each element in iterable
3    else:
4        pass # if no error 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
13
14   n = 121213
15
16   dividers = []
17   while n > 3:
```

```python
        for divider in range(2, int(n ** 0.5) + 1):
            if n % divider == 0:
                break
        else:
            break
        n //= divider
        dividers.append(divider)

    if n != 1:
        dividers.append(n)
```

```python
1    for x in container:
2        f(x)
3
4    # some times equal to
5
6    _tmp = 0
7    while _tmp < len(container):
8        x = container[_tmp]
9        f(x)
10       _tmp += 1
```

break & continue как всегда

- break выходит из цикла

- continue переходит к следующей итерации

Нет

- goto

- switch + case

- until

- dowhile, dountil

```python
1    with expression as var:
2        block
3
4    # mostly the same as
5
6    var = expression
7    var.__enter__()
8
9    block
10
11   if error_happened:
12       if var.__exit__(error_data):
13           # pass_error_further
14       else:
15           # supress_error
16   else:
17       var.__exit__()
```

использование with

```python
with open("rC:\xxx."bin, "w") as fd:
    fd.write"""(- * 100 + "\n")
    fd.write"""(+ * 100 + "\n")

with open("rC:\xxx."bin, "r") as fd:
    for line in fd:
        print line

with db.cursor() as cur:
    curr.execute(update_request_1)
    curr.execute(update_request_2)
    # commit or rollback
```

List comprehension

```python
1      res = [func(i) for i in some_iter if func2(i)]
2
3      res = ["{:.2f}".format(i ** 0.5)
4                for i in [-1, 0, 1, 2, 3]
5                  if i >= 0]
6
7      res == ['0.00', '1.00', '1.41', '1.73']
8
9      res = [(i + 0j) ** 0.5 for i in [-1, 0 ,1, 2, 3]]
10     res = {func(i) for i in some_iter if func2(i)}
```

Функции - минимум

```python
def func_name1(param1, param2):
    "documentation"
    # block
    x = param1 + param2
    return x

def func_name2(param1, param2):
    "documentation"
    # block
    x = param1 + param2
    if x > 0:
        return x
    else:
        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())
```