



Основы программирования (КИСиП)

[В начало](#) / [Мои курсы](#) / [Основы программирования \(КИСиП\)](#) / Для ЗАОЧНОЙ ФОРМЫ (контрольные точки)

/ [Контрольная работа 21/22 года обучения](#)

Контрольная работа 21/22 года обучения

К контрольной работе приступать после изучения: [лекции](#) , [литература](#) и [Глоссарий \(основные определения\)](#).

Проходной балл 60% (максимально 40% за тестовые вопросы и 60% за программу). Программу проверяют в течении 3 дней.

Циклы

Циклы

Цикл while

Цикл while используется в Python для неоднократного исполнения определенной инструкции до тех пор, пока заданное условие остается истинным. Этот цикл позволяет программе перебирать блок кода.

```
while test_expression:
```

```
    body of while
```

Сначала программа оценивает условие цикла while. Если оно истинное, начинается цикл, и тело while выполняется. Тело будет исполняться до тех пор, пока условие остается истинным. Если оно становится ложным, программа выходит из цикла и прекращает исполнение тела.

Рассмотрим пример, чтобы лучше понять.

```
a = 1
```

```
while a < 10:
```

```
    print('Цикл выполнен', a, 'раз(a)') a = a+1
```

```
print('Цикл окончен')
```

Бесконечный цикл while в Python

Бесконечный цикл while — это [цикл](#), в котором условие никогда не становится ложным. Это значит, что тело исполняется снова и снова, а цикл никогда не заканчивается.

Следующий пример — бесконечный цикл: a = 1

```
while a==1:
```

```
    b = input("Как тебя зовут?")
```

```
    print('Привет', b, ', Добро пожаловать')
```

Else в цикле while

В [Python с циклами](#) while также можно использовать инструкцию else. В этом случае блок в else исполняется, когда условие цикла становится ложным.

```
a = 1
```

```
while a < 5:
```

```
    print('условие верно') a = a + 1
```

```
else:
```

```
    print('условие неверно')
```

Программа исполняет код цикла `while` до тех, пока условие истинно, то есть пока значение `a` меньше 5. Поскольку начальное значение `a` равно 1, а с каждым циклом оно увеличивается на 1, условие станет ложным, когда программа доберется до четвертой итерации — в этот момент значение `a` изменится с 4 до 5. Программа проверит условие еще раз, убедится, что оно ложно и исполнит блок `else`, отобразив «условие неверно».

Прерывания цикла `while` в Python

В Python есть два ключевых слова, с помощью которых можно преждевременно остановить итерацию цикла.

Break — ключевое слово `break` прерывает цикл и передает управление в конец цикла

```
a = 1
```

```
while a < 5: a += 1
```

```
    if a == 3:
```

```
        break print(a)
```

```
# 2
```

Continue — ключевое слово `continue` прерывает текущую итерацию и передает управление в начало цикла, после чего условие снова проверяется. Если оно истинно, выполняется следующая итерация.

```
a = 1
```

```
while a < 5:
```

```
    a += 1
```

```
    if a == 3:
```

```
        continue print(a)
```

```
# 2, 4, 5
```

Python цикл `for` — `for i in range`

Циклы `for` повторяют определённый код для некоторого набора значений.

Из документации Python можно узнать, что в нем циклы `for` работают несколько иначе, чем в таких языках, как JavaScript или C.

Цикл `for` присваивает итерируемой переменной каждое значение из предоставленного списка, массива или строки и повторяет код в теле цикла `for` для каждого установленного таким образом значения переменной-итератора.

В приведенном ниже примере мы используем цикл `for` для вывода каждого числа в нашем массиве.

```
# Простой пример цикла for
```

```
for i in [0, 1, 2, 3, 4, 5]:
```

```
    print(i, end="; ")
```

```
# выведет: 0; 1; 2; 3; 4; 5;
```

В тело цикла `for` можно включить и более сложную логику. В следующем примере мы выводим результат небольшого вычисления, основанного на значении переменной `i`.

```
# Пример посложнее
```

```
for i in [0, 1, 2, 3, 4, 5]:
```

```
x = (i-2)*(i+2) - i**2 + 4
```

```
print(x, end="; ")
```

```
# выведет: 0; 0; 0; 0; 0; 0;
```

Когда значения в массиве для нашего цикла `for` представляют собой некоторую закономерную последовательность, мы можем использовать функцию Python `range()` вместо того, чтобы вписывать содержимое нашего массива вручную.

Функция Range в Python

Функция `range()` возвращает последовательность целых чисел на основе переданных ей аргументов. Дополнительную информацию можно найти в документации Python по функции `range()`.

```
range(stop)
```

```
range(start, stop[, step])
```

Аргумент `start` — это первое значение в диапазоне. Если функция `range()` вызывается только с одним аргументом, то Python считает, что `start = 0`.

Аргумент `stop` — это верхняя граница диапазона. Важно понимать, что само граничное значение не включается в последовательность.

В примере ниже у нас есть диапазон, начинающийся со значения по умолчанию, равному 0, и включающий целые числа меньше 6.

```
# Использование range() с единственным аргументом
```

```
for i in range(6):
```

```
    print(i, end="; ")
```

```
# выведет: 0; 1; 2; 3; 4; 5;
```

В следующем примере мы задаем `start = -2` и включаем целые числа меньше 4.

```
# В этот раз передаются два аргумента
```

```
for i in range(-2, 4):
```

```
    print(i, end="; ")
```

```
# выведет: -2; -1; 0; 1; 2; 3;
```

Необязательное значение `step` (шаг) управляет приращением между значениями последовательности. По умолчанию `step = 1`.

В нашем последнем примере мы используем диапазон целых чисел от -2 до 6 и задаем `step = 2`.

```
# Здесь используются все аргументы range()
```

```
for i in range(-2, 6, 2):
```

```
    print(i, end="; ")
```

```
# выведет: -2; 0; 2; 4;
```

Вопросы

[◀ Проверочное тестирование по лекциям для самоконтроля](#)

Перейти на...

[Итоговый тест\(доступен на сессии\) ▶](#)

Вы зашли под именем Александр Максимович Кулабухов (Выход)

Основы программирования (КИСиП)

Сводка хранения данных

