# 📘 День 9

## Условия

По умолчанию операторы в сценарии Python выполняются последовательно сверху вниз. Если этого требует логика обработки, последовательный поток выполнения может быть изменен двумя способами:

- Условное выполнение: блок из одного или нескольких утверждений будет выполнен, если определенное выражение будет истинным

- Повторяющееся выполнение: блок из одного или нескольких операторов будет повторяться до тех пор, пока определенное выражение истинно. В этом разделе мы рассмотрим операторы \_if\_, \_else\_, \_elif\_. Здесь пригодятся операторы сравнения и логические операторы, которые мы изучали в предыдущих разделах.

### Если условие

В python и других языках программирования ключевое слово \_if\_ используется для проверки истинности условия и выполнения кода блока. Помните об отступе после двоеточия.

``py

# синтаксис

если условие:

эта часть кода выполняется для условий истинности

```

\*\*Пример: 1\*\*

``py

a = 3

если a > 0:

print('A - положительное число')

# A - положительное число

```

Как видно из приведенного примера, 3 больше 0. Условие было истинным, и код блока был выполнен. Однако если условие ложно, мы не увидим результата. Для того чтобы увидеть результат ложного условия, мы должны иметь еще один блок, который будет \_else\_.

### If Else

Если условие истинно, то будет выполнен первый блок, если нет, то будет выполнено условие else.

``py

# синтаксис

если условие:

эта часть кода выполняется для условий истинности

еще:

эта часть кода выполняется для ложных условий

```

\*\*Пример: \*\*

``py

a = 3

если a < 0:

print('A - отрицательное число')

еще:

print('A - положительное число')

```

Условие выше оказалось ложным, поэтому был выполнен блок else. А как быть, если наше условие больше двух? Мы можем использовать \_ elif\_.

### If Elif Else

В нашей повседневной жизни мы ежедневно принимаем решения. Мы принимаем решения, проверяя не одно или два условия, а множество условий. Как и в жизни, в программировании тоже много условий. Мы используем \_elif\_, когда у нас есть несколько условий.

``py

# синтаксис

если условие:

код

elif условие:

код

еще:

код

```

\*\*Пример: \*\*

``py

a = 0

если a > 0:

print('A - положительное число')

elif a < 0:

print('A - отрицательное число')

еще:

print('A равно нулю')

```

### Короткая рука

``py

# синтаксис

код if условие else код

```

\*\*Пример: \*\*

``py

a = 3

print('A is positive') if a > 0 else print('A is negative') # первое условие выполнено, будет выведено 'A is positive'

```

### Вложенные условия

Условия могут быть вложенными

``py

# синтаксис

если условие:

код

если условие:

код

```

\*\*Пример: \*\*

``py

a = 0

если a > 0:

если a % 2 == 0:

print('A - положительное и четное целое число')

еще:

print('A - положительное число')

elif a == 0:

print('A равно нулю')

еще:

print('A - отрицательное число')

```

Мы можем избежать написания вложенного условия, используя логический оператор \_and\_.

### Условие If и логические операторы

``py

# синтаксис

если условие и состояние:

код

```

\*\*Пример: \*\*

``py

a = 0

если a > 0 и a % 2 == 0:

print('A - четное и положительное целое число')

elif a > 0 и a % 2 != 0:

print('A - целое положительное число')

elif a == 0:

print('A равно нулю')

еще:

print('A отрицательно')

```

### Логические операторы If и Or

``py

# синтаксис

если условие или состояние:

код

```

\*\*Пример: \*\*

``py

пользователь = 'James'

уровень доступа = 3

если user == 'admin' или access\_level >= 4:

print('Доступ предоставлен!')

еще:

print('Доступ запрещен!')

```

🌕 Вы молодец, никогда не сдавайтесь, потому что на великие дела нужно время. Вы только что завершили 9-й день испытаний и сделали 9 шагов вперед на пути к величию. Теперь сделайте несколько упражнений для мозга и мышц.

## 💻 Упражнения: День 9

### Упражнения: Уровень 1

1. Получите данные пользователя с помощью input("Введите ваш возраст: "). Если пользователю 18 лет или больше, дайте обратную связь: Вы достаточно взрослый, чтобы водить машину. Если пользователю меньше 18 лет, выдайте сообщение о необходимости подождать недостающее количество лет. Выходные данные:

``sh

Введите свой возраст: 30 лет

Вы уже достаточно взрослый, чтобы научиться водить машину.

Выход:

Введите свой возраст: 15

Вам нужно еще 3 года, чтобы научиться водить машину.

```

2. Сравните значения my\_age и your\_age с помощью if ... else. Кто старше (я или вы)? Используйте input("Введите ваш возраст: "), чтобы получить возраст в качестве входных данных. Вы можете использовать вложенное условие, чтобы вывести 'year' для разницы в возрасте в 1 год, 'years' для большей разницы, а также пользовательский текст, если my\_age = your\_age. Выходные данные:

``sh

Введите свой возраст: 30 лет

Ты на 5 лет старше меня.

```

3. Получить от пользователя два числа с помощью подсказки ввода. Если a больше b, вернуть a больше b, если a меньше b, вернуть a меньше b, иначе a равно b. Вывод:

``sh

Введите цифру 1: 4

Введите номер два: 3

4 больше 3

```

### Упражнения: Уровень 2

1. Напишите код, который выставляет оценки студентам в соответствии с их баллами:

``sh

80-100, A

70-89, B

60-69, C

50-59, D

0-49, F

```

1. Проверьте, является ли время года осенью, зимой, весной или летом. Если пользователь ввел данные:

Сентябрь, октябрь или ноябрь, время года - осень.

Декабрь, январь или февраль, сезон - Зима.

Март, апрель или май, сезон - весна

Июнь, июль или август, сезон - лето

2. В следующем списке приведены некоторые фрукты:

``sh

фрукты = ['банан', 'апельсин', 'манго', 'лимон'].

```

Если фрукт не существует в списке, добавьте его в список и выведите измененный список. Если фрукт существует print('Этот фрукт уже существует в списке')

### Упражнения: Уровень 3

1. Перед вами словарь человека. Не стесняйтесь изменять его!

``py

человек={

'first\_name': 'Asabeneh',

'last\_name': 'Yetayeh',

'age': 250,

'страна': 'Финляндия',

'is\_marred': True,

'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],

'адрес': {

'улица': 'Космическая улица',

'zipcode': '02210'

}

}

```

\* Проверьте, есть ли в словаре человека ключ навыков, если да, то выведите средний навык в списке навыков.

\* Проверьте, есть ли в словаре персон ключ skills, если да, то проверьте, есть ли у персоны навык 'Python', и выведите результат.

\* Если навыки человека включают только JavaScript и React, выведите('Он front end разработчик'), если навыки человека включают Node, Python, MongoDB, выведите('Он backend разработчик'), если навыки человека включают React, Node и MongoDB, выведите('Он fullstack разработчик'), иначе выведите('неизвестное название') - для более точных результатов можно вложить больше условий!

\* Если человек состоит в браке и проживает в Финляндии, напечатайте информацию в следующем формате:

``py

Асабенех Йетайех живет в Финляндии. Он женат.

```

<div align="center">

<h1>30 дней Python: День 10 - Циклы</h1>

<a class="header-badge" target="\_blank" href="https://www.linkedin.com/in/asabeneh/">

<img src="https://img.shields.io/badge/style--5eba00.svg?label=LinkedIn&logo=linkedin&style=social">

</a>

<a class="header-badge" target="\_blank" href="https://twitter.com/Asabeneh">

<img alt="Twitter Follow" src="https://img.shields.io/twitter/follow/asabeneh?style=social">.

</a>

<sub>Автор:

<a href="https://www.linkedin.com/in/asabeneh/" target="\_blank">Asabeneh Yetayeh</a><br>

<small>Второе издание: Июль, 2021</small>

</sub>

</div>

# 📘 День 10

## Петли

Жизнь полна рутины. В программировании мы также выполняем множество повторяющихся задач. Чтобы справиться с повторяющимися задачами, языки программирования используют циклы. Язык программирования Python также предоставляет следующие типы двух циклов:

1. цикл while

2. цикл for

### While Loop

Для создания цикла while мы используем зарезервированное слово \_while\_. Он используется для многократного выполнения блока операторов до тех пор, пока не будет выполнено заданное условие. Когда условие становится ложным, строки кода после цикла продолжат выполняться.

``py

# синтаксис

пока состояние:

код здесь

```

\*\*Пример:\*\*

``py

счёт = 0

while count < 5:

print(count)

count = count + 1

#печатает от 0 до 4

```

В приведенном выше цикле while условие становится ложным, когда счетчик равен 5. Тогда цикл останавливается.

Если мы хотим выполнить блок кода после того, как условие перестанет быть истинным, мы можем использовать \_else\_.

``py

# синтаксис

пока состояние:

код здесь

еще:

код здесь

```

\*\*Пример:\*\*

``py

счёт = 0

while count < 5:

print(count)

count = count + 1

еще:

print(count)

```

Приведенное выше условие цикла станет ложным, когда счетчик будет равен 5, цикл остановится, и начнется выполнение оператора else. В результате будет выведено 5.

### Перерыв и продолжение - часть 1

- Перерыв: Мы используем break, когда хотим выйти из цикла или остановить его.

``py

# синтаксис

пока состояние:

код здесь

если другое\_условие:

перерыв

```

\*\*Пример:\*\*

``py

счёт = 0

while count < 5:

print(count)

count = count + 1

если счет == 3:

перерыв

```

Приведенный выше цикл while выводит только 0, 1, 2, но когда он достигает 3, то останавливается.

- Продолжить: С помощью оператора continue мы можем пропустить текущую итерацию и продолжить следующую:

``py

# синтаксис

пока состояние:

код здесь

если другое\_условие:

продолжить

```

\*\*Пример:\*\*

``py

счёт = 0

while count < 5:

если счет == 3:

продолжить

print(count)

count = count + 1

```

Приведенный выше цикл while выводит только 0, 1, 2 и 4 (пропускает 3).

### For Loop

Ключевое слово \_for\_ используется для создания цикла for, аналогично другим языкам программирования, но с некоторыми отличиями в синтаксисе. Loop используется для итерации по последовательности (это либо список, кортеж, словарь, набор или строка).

- Цикл For со списком

``py

# синтаксис

для iterator в lst:

код здесь

```

\*\*Пример:\*\*

``py

числа = [0, 1, 2, 3, 4, 5]

for number in numbers: # number - это временное имя для ссылки на элементы списка, действительное только внутри этого цикла

print(number) # числа будут выводиться построчно, от 0 до 5

```

- Цикл For со строкой

``py

# синтаксис

для iterator в string:

код здесь

```

\*\*Пример:\*\*

``py

язык = 'Python'

для письма в языке:

печать(письмо)

for i in range(len(language)):

print(language[i])

```

- Цикл For с кортежем

``py

# синтаксис

для iterator в tpl:

код здесь

```

\*\*Пример:\*\*

``py

числа = (0, 1, 2, 3, 4, 5)

для числа в числах:

print(number)

```

- Цикл For со словарем

Переход по словарю дает вам ключ словаря.

``py

# синтаксис

для итератора в dct:

код здесь

```

\*\*Пример:\*\*

``py

человек = {

'first\_name':'Asabeneh',

'last\_name':'Yetayeh',

'возраст':250,

'страна': 'Финляндия',

'is\_marred':True,

'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],

'адрес':{

'улица': 'Космическая улица',

'zipcode':'02210'

}

}

за ключом лично:

print(key)

для key, value в person.items():

print(key, value) # таким образом мы получим распечатку и ключей, и значений

```

- Петли в наборе

``py

# синтаксис

для итератора в st:

код здесь

```

\*\*Пример:\*\*

``py

it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

для компании в it\_companies:

print(company)

```

### Перерыв и продолжение - часть 2

Краткое напоминание:

\_Перерыв\_: Мы используем break, когда хотим остановить цикл до его завершения.

``py

# синтаксис

для итератора в последовательности:

код здесь

если условие:

перерыв

```

\*\*Пример:\*\*

``py

числа = (0,1,2,3,4,5)

для числа в числах:

print(number)

если число == 3:

перерыв

```

В приведенном выше примере цикл останавливается, когда достигает 3.

Продолжить: Мы используем continue, когда хотим пропустить некоторые шаги в итерации цикла.

``py

# синтаксис

для итератора в последовательности:

код здесь

если условие:

продолжить

```

\*\*Пример:\*\*

``py

числа = (0,1,2,3,4,5)

для числа в числах:

print(number)

если число == 3:

продолжить

print('Следующее число должно быть ', number + 1) if number != 5 else print("loop's end") # для коротких условий нужны оба оператора if и else

print('вне цикла')

```

В приведенном выше примере, если число равно 3, шаг \*после\* условия (но внутри цикла) пропускается и выполнение цикла продолжается, если остались итерации.

### Функция диапазона

Функция \_range()\_ использует список чисел. Функция \_range(start, end, step)\_ принимает три параметра: начало, конец и приращение. По умолчанию он начинается с 0, а шаг равен 1. Последовательность диапазонов требует как минимум 1 аргумента (конец).

Создание последовательностей с использованием диапазона

``py

lst = list(range(11))

print(lst) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

st = set(range(1, 11)) # 2 аргумента указывают начало и конец последовательности, шаг установлен по умолчанию 1

print(st) # {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

lst = list(range(0,11,2))

print(lst) # [0, 2, 4, 6, 8, 10]

st = set(range(0,11,2))

print(st) # {0, 2, 4, 6, 8, 10}

```

``py

# синтаксис

for iterator in range(start, end, step):

```

\*\*Пример:\*\*

``py

for number in range(11):

print(number) # печатает от 0 до 10, не включая 11

```

### Вложенный цикл For Loop

Мы можем писать циклы внутри цикла.

``py

# синтаксис

для x в y:

для t в x:

print(t)

```

\*\*Пример:\*\*

``py

человек = {

'first\_name': 'Asabeneh',

'last\_name': 'Yetayeh',

'age': 250,

'страна': 'Финляндия',

'is\_marred': True,

'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],

'адрес': {

'улица': 'Космическая улица',

'zipcode': '02210'

}

}

за ключом лично:

если key == 'skills':

для skill в person['skills']:

print(skill)

```

### For Else

Если мы хотим выполнить какое-то сообщение при завершении цикла, мы используем else.

``py

# синтаксис

for iterator in range(start, end, step):

делать что-то

еще:

print('Цикл завершился')

```

\*\*Пример:\*\*

``py

for number in range(11):

print(number) # печатает от 0 до 10, не включая 11

еще:

print('Цикл останавливается на', number)

```

### Пас

В python, когда требуется оператор (после точки с запятой), но мы не хотим выполнять там какой-либо код, мы можем написать слово \_pass\_, чтобы избежать ошибок. Также мы можем использовать это слово как заполнитель для будущих утверждений.

\*\*Пример:\*\*

``py

for number in range(6):

пройти

```

🌕 Вы достигли большого рубежа, вас не остановить. Продолжайте! Вы только что завершили 10-й день испытаний, и вы на 10 шагов впереди на пути к величию. Теперь сделайте несколько упражнений для вашего мозга и мышц.

## 💻 Упражнения: День 10

### Упражнения: Уровень 1

1. Выполните итерацию от 0 до 10 с помощью цикла for, то же самое сделайте с помощью цикла while.

2. Выполните итерацию от 10 до 0 с помощью цикла for, сделайте то же самое с помощью цикла while.

3. Напишите цикл, который делает семь вызовов print(), так что на выходе мы получаем следующий треугольник:

``py

#

##

###

####

#####

######

#######

```

4. С помощью вложенных циклов создайте следующее:

``sh

# # # # # # # #

# # # # # # # #

# # # # # # # #

# # # # # # # #

# # # # # # # #

# # # # # # # #

# # # # # # # #

# # # # # # # #

```

5. Напечатайте следующий рисунок:

``sh

0 x 0 = 0

1 x 1 = 1

2 x 2 = 4

3 x 3 = 9

4 x 4 = 16

5 x 5 = 25

6 x 6 = 36

7 x 7 = 49

8 x 8 = 64

9 x 9 = 81

10 x 10 = 100

```

6. Пройдитесь по списку ['Python', 'Numpy', 'Pandas', 'Django', 'Flask'] с помощью цикла for и выведите элементы.

7. Используйте цикл for для итерации от 0 до 100 и выведите только четные числа

8. Используйте цикл for для итерации от 0 до 100 и выведите только нечетные числа

### Упражнения: Уровень 2

1. Используйте цикл for для итерации от 0 до 100 и выведите сумму всех чисел.

``sh

Сумма всех чисел равна 5050.

```

1. Используйте цикл for для итерации от 0 до 100 и выведите сумму всех четных и сумму всех нечетных значений.

``sh

Сумма всех четных равна 2550. А сумма всех шансов равна 2500.

```

### Упражнения: Уровень 3

1. Перейдите в папку data и используйте файл [countries.py](https://github.com/Asabeneh/30-Days-Of-Python/blob/master/data/countries.py). Пройдитесь по странам и извлеките все страны, содержащие слово \_land\_.

1. Это список фруктов, ['банан', 'апельсин', 'манго', 'лимон'] измените порядок с помощью цикла.

2. Перейдите в папку data и используйте файл [countries\_data.py](https://github.com/Asabeneh/30-Days-Of-Python/blob/master/data/countries-data.py).

1. Каково общее количество языков в данных

2. Найдите десять наиболее распространенных языков из полученных данных

3. Найдите 10 самых населенных стран мира