Python . Циклы и ветвления

Для написания кода можно воспользоваться онлайн-компилятором по ссылке: https://www.onlinegdb.com/online_python_compiler

Необходимо ознакомится с теорией и выполнить задачи в компиляторе. После выполнения делать скриншот выполненной задачи.

Циклы

В обычной жизни зачастую приходится выполнять серии одинаковых операций.

Например, задача «перемыть эту груду грязных тарелок» решается так: взять тарелку — помыть — сполоснуть — вытереть — убрать — взять следующую тарелку — помыть — ...и повторять по кругу.

Этот процесс не бесконечен: работа должна продолжаться лишь при условии, что остались грязные тарелки. Тарелки кончились — работа прекращается.

В разработке тоже приходится мыть посуду выполнять одинаковые действия со всеми элементами списка, это довольно востребованная операция.

Возьмём, для примера, список всемирно известных бременских музыкантов.

```
bremen_musicians = ['Кот', 'Пёс', 'Трубадур', 'Осёл', 'Петух']
```

На афише необходимо напечатать имена всех музыкантов, вот таким образом:

```
Кот
Пёс
Трубадур
Осёл
Петух
```

Как это сделать в Python? Можно вручную прописать вывод каждого элемента:

```
print(bremen_musicians[0])
print(bremen_musicians[1])
...
```

Придётся пять раз писать один и тот же код. А если в списке не скромная рок-группа, а симфонический оркестр и академический хор в придачу?

Задача — точно как с тарелками: нужно взять первый элемент списка bremen_musicians, напечатать через print(), потом взять следующий элемент... и продолжать работу до тех пор, пока не будет обработан последний элемент списка.

Для выполнения таких операций придумали **циклы** — программные конструкции, выполняющие определённые действия до тех пор, пока выполняется заданное условие.

Как это пишется

Объявление цикла

Чтобы программа поняла, что сейчас начнётся цикл — нужно сообщить ей об этом специальными словами: **объявить цикл**. В Python есть несколько разновидностей циклов; перебирать список проще всего циклом for … in …, с него и начнём.

Цикл в Python объявляется ключевыми словами for и in; после объявления ставится двоеточие. Ниже объявления пишут **тело цикла** — код, который описывает, что же нужно сделать с каждым элементом списка.

```
for переменная in список_элементов: # Вот оно, объявление цикла
    # Тут будет тело цикла.
```

В условии цикла после for указывают имя переменной, в которую будут поочерёдно передаваться элементы из обрабатываемого списка, а после in ставится имя списка, который надо обработать.

Продолжая метафору, «список элементов» — это груда немытых тарелок, а «переменная, объявленная в условии цикла» — это очередная взятая из груды тарелка.

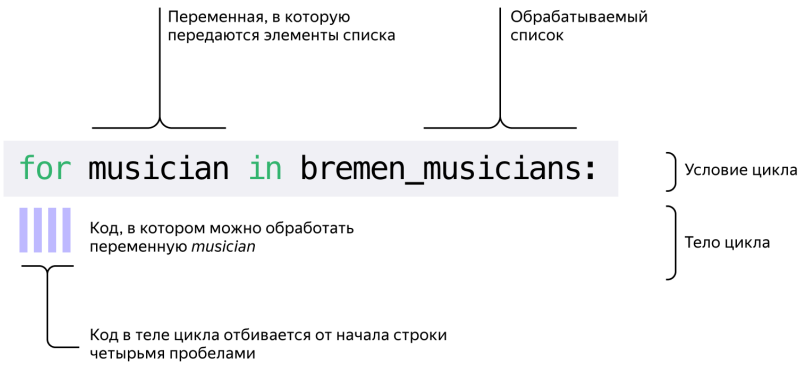
Цикл автоматически прекратит работу, когда переберёт все элементы списка.

Имя переменной в цикле вы можете дать любое, но традиционно эти имена образуют от имени обрабатываемого списка, в единственном числе. Например, если список называется musicians, то переменную лучше назвать musician; если список называется pigs — переменную называют pig.

Тело цикла

На следующей строке после объявления цикла пишут его **тело**. Каждая строка тела цикла обязательно отбивается от начала строки четырьмя пробелами:

```
for переменная in список_элементов:
    # Тут тело цикла: код, который выполняется для каждого элемента
    # Здесь можно обработать переменную, объявленную в условии цикла,
    # например, напечатать её значение: print(переменная)
```



Теперь можно написать цикл, который автоматически напечатает имена всех этих хиппи из команды «Бременские музыканты».

```
bremen_musicians = ['Кот', 'Пёс', 'Трубадур', 'Осёл', 'Петух']

for musician in bremen_musicians:
    # Каждый элемент списка bremen_musicians
    # по очереди будет передан в переменную musician
```

```
    # и напечатан
    print(musician)

# Здесь может быть какой-то код, который выполнится
# только после того, как цикл закончит работу
```

Цикл берёт значение первого элемента из списка bremen_musicians и передаёт его в переменную musician. Затем выполняется код в теле цикла: печатается содержимое переменной musician.

После этого начнётся новый «круг», со следующим элементом списка. И так будет продолжаться до тех пор, пока цикл не переберёт весь список.

Каждый такой «круг» называется **итерацией цикла**.

Когда список закончится — программа выйдет из цикла; после этого сработает код, который написан после цикла.

Задача 1

Напечатайте с помощью цикла названия месяцев из списка months.

Код PYTHON Подсказка • После ключевого слова in укажите имя списка, который будете обрабатывать в цикле: months. • Проследите, чтобы после объявления цикла стояло двоеточие. • В функцию print() передайте переменную month. • Тело цикла отбивается четырьмя пробелами, убедитесь, что они есть перед print(month).	months = ['Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь', 'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь', 'Декабрь'] for month in ...: print(...)

Задача 2

Напечатайте приглашение на ужин: циклом выведите имена приглашённых из списка pigs.

Опишите условие и тело цикла. В результате ваша программа должна напечатать такой текст:

```
Дорогие свиньи!
Ниф-Ниф
Наф-Наф
Нуф-Нуф
приглашаю вас на ужин!
Любящий вас Волк.
```

Подсказка • Циклом for переберите все элементы списка pigs. Внутреннюю переменную можно	pigs = ['Ниф-Ниф', 'Наф-Наф', 'Нуф-Нуф'] print('Дорогие свиньи!')

назвать pig. Условие цикла должно выглядеть так: for <переменная> in <список>: • После условия цикла должно стоять двоеточие. • В теле цикла напечатайте значение внутренней переменной: print(<переменная>). • Код в теле цикла должен быть отбит четырьмя пробелами.	for ... ... print('приглашаю вас на ужин!') print('Любящий вас Волк.')

Отступы в коде циклов

Код программы, написанной на любом современном языке программирования, делится на логические блоки. Чтобы и компьютеру, и человеку было понятно, где начинается и где заканчивается определённый блок — блоки надо как-то отделять друг от друга.

Например, цикл — это отдельный блок кода. Тело цикла должно быть отделено от остального кода, чтобы компьютер знал, какие именно инструкции надо выполнять в цикле.

В качестве основного средства формирования блоков кода в Python используются **отступы**.

В Python отступ — это пустое место в начале строки кода. Каждый вложенный блок отбивается от начала строки четырьмя отступами; код, следующий за блоком, снова пишут от начала строки. В качестве отступов используются пробелы или табы.

Табы или пробелы?

В Python отступы можно задавать пробелами или **табами** — невидимыми символами, которые печатаются при нажатии клавиши Tab. Технической и визуальной разницы нет: код будет работать и с пробелами, и с табами, а отличить таб от пробела на глаз не получится.

Но есть важное ограничение: в коде нельзя одновременно использовать и табы, и пробелы; Python этого не поймёт, нужно выбрать что-то одно.

Если в коде отступы заданы табами и пробелами вперемешку, то при запуске программы Python сообщит об ошибке.

Вот небольшой код, **визуально** с ним все в порядке. Запустите его.

Код	numbers = [1, 2, 3]
	for number in numbers:
	  print('Печатаем элемент списка')
	  print(number)

Ошибка! Inconsistent use of tabs and spaces in indentation — «для отступов применены табы и пробелы». Исправьте ошибку: замените табы на пробелы и запустите программу ещё раз.

Угадать, где в коде табы, а где пробелы, будет сложно; если угадать не удастся — при запуске кода вы снова получите ошибку.

Чтобы не приходилось гадать, какими символами коллеги отбивают свой код — каждая команда разработчиков принимает решение, какой именно символ применять в качестве отступа.

Отступы в циклах

Напишем полезную программу, которая выведет на экран считалку:

```
Раз
Два
Три
Четыре
Пять
Вышел зайчик погулять
```

Соберём числительные в список и пробежимся по этому списку циклом; после цикла выведем финальную фразу:

Код	numbers = ['Раз', 'Два', 'Три', 'Четыре', 'Пять'] for number in numbers: print(number) print('Вышел зайчик погулять')

Отступы в строке с инструкцией print(number) обозначают, что эта строка вложена в тело цикла for number in numbers; в нём только одна строка. Благодаря отступам тело цикла видно и программе, и нам.

Ниже находится та же самая программа, но с неправильным использованием отступов: последняя строка кода тоже отбита отступами — разработчик мог решить, что так код выглядит красивее. Однако программа восприняла это по-своему: «Понятненько: отступы перед строкой означают, что код в этой строке вложен в блок цикла!».

Запускаем — и…

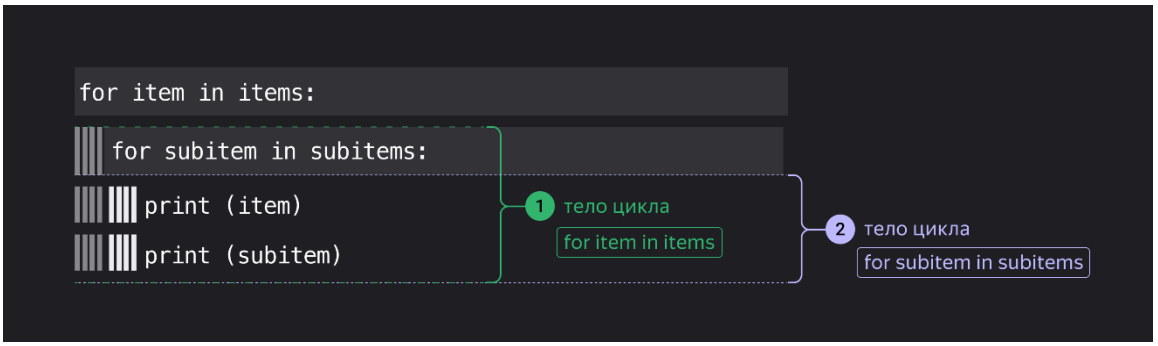
Код	numbers = ['Раз', 'Два', 'Три', 'Четыре', 'Пять']
	for number in numbers:
	  print(number)
	  print('Вышел зайчик погулять')

Логика поломалась, зайчики выходят гулять на каждой итерации цикла; в результате вместо одного зайчика гуляет целых пять.

Отступы во вложенных циклах

Циклы можно вкладывать друг в друга, при этом вложенный цикл будет выполняться для каждой итерации внешнего цикла. Такие циклы называются **вложенными**.

С каждым уровнем вложенности число отступов перед телом цикла растёт. На каждом дополнительном уровне вложенности код отбивают дополнительными четырьмя отступами.



Разберёмся на конкретном примере. У нас есть список имён детей и список конфет. Каждому ребёнку надо дать по одной конфете каждого вида.

sweets = ['Батончик', 'Сникерс', 'Мишка Косолапый', 'Коровка']
kids = ['Витя', 'Маша', 'Марина']
Вите должны достаться «Батончик», «Сникерс», «Мишка Косолапый» и «Коровка». Такой же набор конфет полагается и Маше, и Марине.

Программе нужно по очереди перебрать имена детей; для каждого имени должен сработать вложенный цикл, который для этого имени переберёт список конфет; после этого внешний цикл перейдёт к следующему имени.

В тело внешнего цикла for kid in kids вложен цикл for sweet in sweets. Перед объявлением вложенного цикла должно стоять четыре пробела.

В теле вложенного цикла for sweet in sweets должна быть функция print(kid, 'получает конфету', sweet). Перед ней — восемь пробелов:

- четырьмя пробелами отбито тело цикла for kid in kids;
- ещё четырьмя пробелами — тело вложенного цикла for sweet in sweets.

Пока что программа не работает: в коде проставлены неправильные отступы в телах двух циклов.

Задача. Поправьте код и убедитесь, что каждый ребёнок получил все конфеты!

Код	kids = ['Витя', 'Маша', 'Марина'] sweets = ['Батончик', 'Сникерс', 'Мишка Косолапый', 'Коровка'] for kid in kids: for sweet in sweets: # Внесите изменения в строку. print(kid, 'получает конфету', sweet) # Внесите изменения в строку.

Задача 1. Исправьте ошибки в коде.
Вот считалка, в которую по очереди подставляются имена детей:

```
У Литейного моста
Я поймал в Неве кита,
Спрятал за окошко.
Съела его кошка,
Помогали два кота…
Вот и нет теперь кита!
Ты не веришь другу?
Выходи из круга!
* Из круга выходит <имя ребёнка> *
...повтор про следующего ребёнка
...и про следующего

Всё!
```

Код Подсказка • Перед всеми вызовами функции print(), кроме последней строки, должны стоять четыре пробела: так эти строки окажутся в теле цикла for kid in kids. В том числе и вызов функции print() без аргументов. • Последняя строка должна быть без отступов: она не должна попасть в тело цикла. • На всякий случай проверьте, не замешались ли среди пробелов табы.	kids = ['Витя', 'Маша', 'Марина'] for kid in kids: print ('У Литейного моста') print('Я поймал в Неве кита,') print('Спрятал за окошко.') print('Съела его кошка,') print('Помогали два кота…') print('Вот и нет теперь кита!') print('Ты не веришь другу?') print('Выходи из круга!') print('* Из круга выходит', kid, '*') print() print('Всё!')

Задача 2
Напечатайте таблицу умножения. Установите правильные отступы и проверьте, что всё напечатано без ошибок.

Код
 	
Подсказка
У вложенного цикла тело должно быть отбито **дополнительными** четырьмя отступами.
	numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
	for number_1 in numbers:
	for number_2 in numbers:
	print(number_1, '*', number_2, '=', number_1 * number_2)