[Черепаха](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html" \l "id36) <http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id3>

Стандартная библиотека Python содержит модуль turtle, предназначенный для обучения программированию. Этот модуль содержит [набор](https://docs.python.org/3/library/turtle.html#methods-of-rawturtle-turtle-and-corresponding-functions) функций, позволяющих управлять черепахой. Черепаха умеет выполнять небольшой набор команд, а именно:

| **Команда** | **Значение** |
| --- | --- |
| forward(X) | Пройти вперёд X пикселей |
| backward(X) | Пройти назад X пикселей |
| left(X) | Повернуться налево на X градусов |
| right(X) | Повернуться направо на X градусов |
| penup() | Не оставлять след при движении |
| pendown() | Оставлять след при движении |
| shape(X) | Изменить значок черепахи (“arrow”, “turtle”, “circle”, “square”, “triangle”, “classic”) |
| stamp() | Нарисовать копию черепахи в текущем месте |
| color() | Установить цвет |
| begin\_fill() | Необходимо вызвать перед рисованием фигуры, которую надо закрасить |
| end\_fill() | Вызвать после окончания рисования фигуры |
| width() | Установить толщину линии |
| goto(x, y) | Переместить черепашку в точку (x, y) |

Например, следующая программа рисует букву S:

**import** **turtle**

turtle.shape('turtle')

turtle.forward(50)

turtle.left(90)

turtle.forward(50)

turtle.left(90)

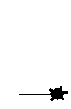
turtle.forward(50)

turtle.right(90)

turtle.forward(50)

turtle.right(90)

turtle.forward(50)

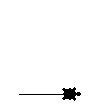


[Упражнение №2: буква S](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id37)

Сохраните и выполните предыдущую программу. Убедитесь в том, что черепаха работает.

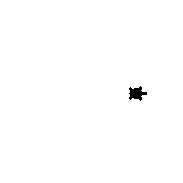
[Упражнение №3: квадрат](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id38)

Нарисуйте квадрат. Пример:



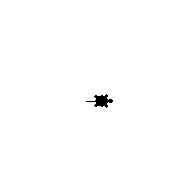
[Упражнение №4: окружность](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id39)

Нарисуйте окружность. Воспользуйтесь тем фактом, что правильный многоугольник с большим числом сторон будет выглядеть как окружность. Пример:



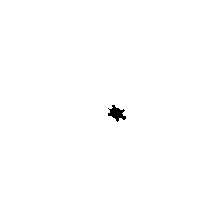
[Упражнение №5: больше квадратов](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id40)

Нарисуйте 10 вложенных квадратов.



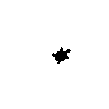
[Упражнение №6: паук](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id41)

Нарисуйте паука с n лапами. Пример n = 12:



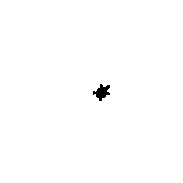
[Упражнение №7: спираль](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id42)

Нарисуйте спираль. См. [теорию](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D0%BC%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0_%D1%81%D0%BF%D0%B8%D1%80%D0%B0%D0%BB%D1%8C). Пример:



[Упражнение №8: квадратная «спираль»](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id43)

Нарисуйте «квадратную» спираль. Пример:



[Написание функций](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id44)

Как было сказано раньше, функции — это своего рода готовые кирпичики, из которых строится программа. До этого момента мы *использовали* стандартные функции (print, input, функции модуля turtle), теперь настало время *написать* функцию:

**>>> def** hello(name):

**...**  **print**('Hello, ', name, '!')

**...**

**>>>** hello('world')

Hello, world!

Это простейший пример функции, которая принимает в качестве **параметра** имя, а затем выводит на экран сообщение Hello, <имя>. Как видно из примера, функции в языке Python описываются при помощи ключевого слова def:

**def** Имя\_функции(параметр\_1, параметр\_2, ...):

Блок\_операций

Так же, как и в случае циклов и условных операторов, **тело** функции выделяется при помощи отступов.

Вызов функции осуществляется по имени с указанием параметров:

hello('world')

Внутри функции можно использовать те же синтаксические конструкции, что и вне её — циклы, ветвления, можно даже описывать новые функции. Естественно, внутри функции можно работать и с переменными.

Написанная ранее функция имеет особенность — она просто просто выводит текст на экран и не возвращает никакого результата. Многие функции, напротив, занимаются вычислением какого-либо значения, а затем **возвращают** его тому, кто эту функцию **вызвал**. В качестве примера можно рассмотреть функцию для сложения двух чисел:

**>>> def** sum(a, b):

**...**  **return** a + b

**...**

**>>>** sum(1, 2)

3

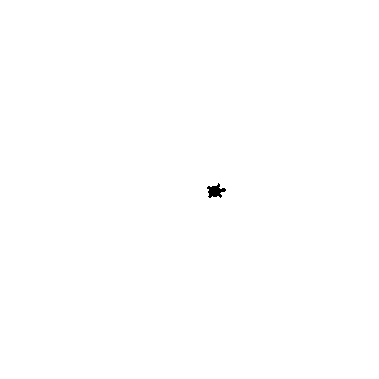
**>>>** sum(5, -7)

-2

Для возврата значения из функции используется оператор return: в качестве параметра указывается значение, которое требуется вернуть.

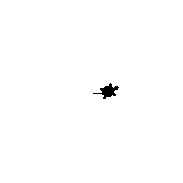
[Упражнение №9: правильные многоугольники](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id45)

Нарисуйте 10 вложенных правильных многоугольников. Используйте функцию, рисующую правильный n-угольник. [Формулы](https://www.fxyz.ru/%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D1%8B_%D0%BF%D0%BE_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%B8/%D0%BF%D0%BB%D0%BE%D1%81%D0%BA%D0%B8%D0%B5_%D1%84%D0%B8%D0%B3%D1%83%D1%80%D1%8B/%D0%B2%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B8_%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%B8/%D1%80%D0%B0%D0%B4%D0%B8%D1%83%D1%81_%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9_%D0%BE%D0%BA%D1%80%D1%83%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8/%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%B0/) для нахождения радиуса описанной окружности. Пример:



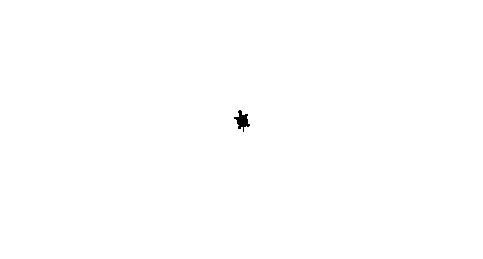
[Упражнение №10: «цветок»](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id46)

Нарисуйте «цветок» из окружностей. Используйте функцию, рисующую окружность. Пример:



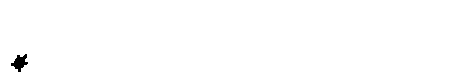
[Упражнение №11: «бабочка»](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id47)

Нарисуйте «бабочку» из окружностей. Используйте функцию, рисующую окружность. Пример:



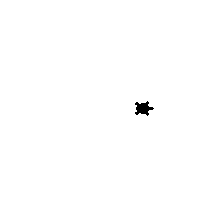
[Упражнение №12: пружина](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id48)

Нарисуйте пружину. Используйте функцию, рисующую дугу. Пример:



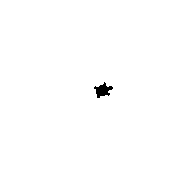
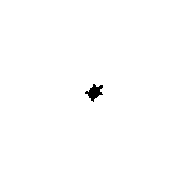
[Упражнение №13: смайлик](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id49)

Нарисуйте смайлик с помощью написанных функций рисования круга и дуги. Пример:



[Упражнение №14: звезды](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id50)

Нарисуйте две звезды: одну с 5 вершинами, другую — с 11. Используйте функцию, рисующую звезду с n вершинами. Пример:

[Красивый код на Python](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab1.html#id51)

Важная мысль создателя языка Python, Гвидо ван Россума: **код читается намного больше раз, чем пишется**.

Поэтому существуют рекомендации о стиле кодирования PEP8. Они направлены на то, чтобы улучшить читаемость и сделать его согласованным между большим числом проектов. В идеале, весь код будет написан в едином стиле, и любой сможет легко его прочесть.

На русском можно прочитать про PEP8 [тут](http://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html), на английском — [здесь](https://www.python.org/dev/peps/pep-0008/%22%3Ehttps:/www.python.org/dev/peps/pep-0008/).