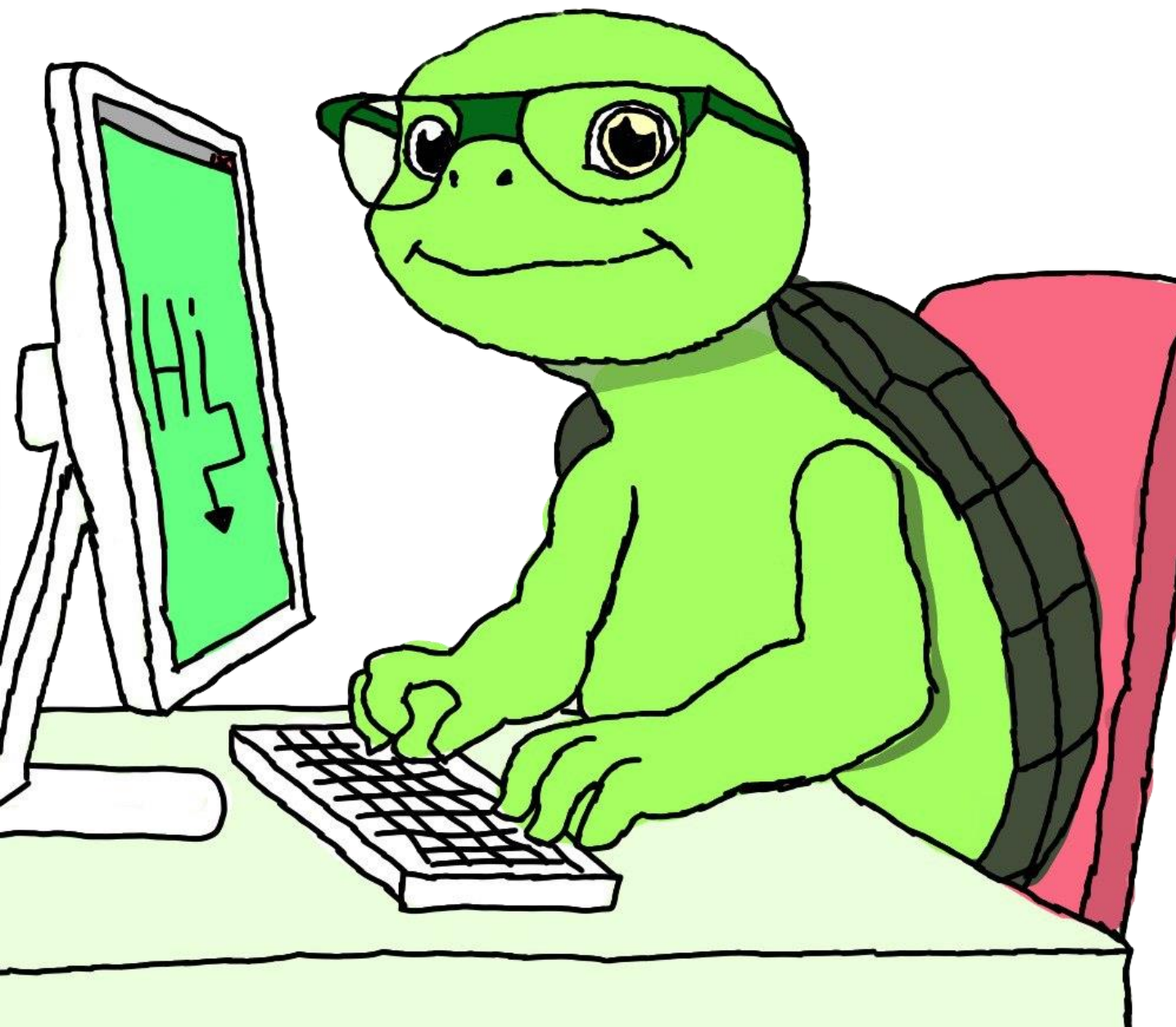
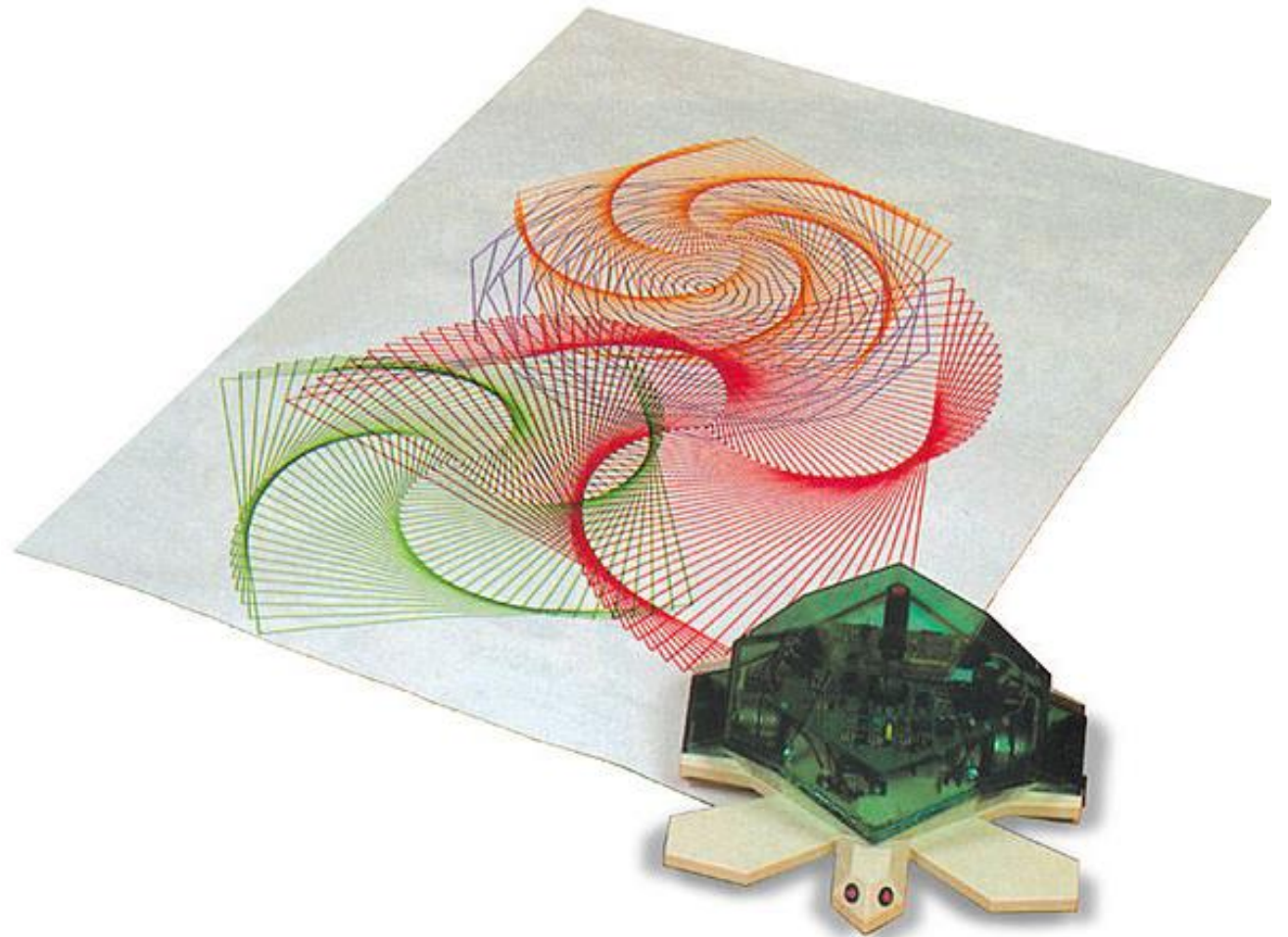




Черепащья графика

НАЧАЛО ПРОГРАММИРОВАНИЯ НА PYTHON

По сравнению с другими видами учебной деятельности программирование – наиболее щадящий процесс, где дети могут делать ошибки и учиться на них.

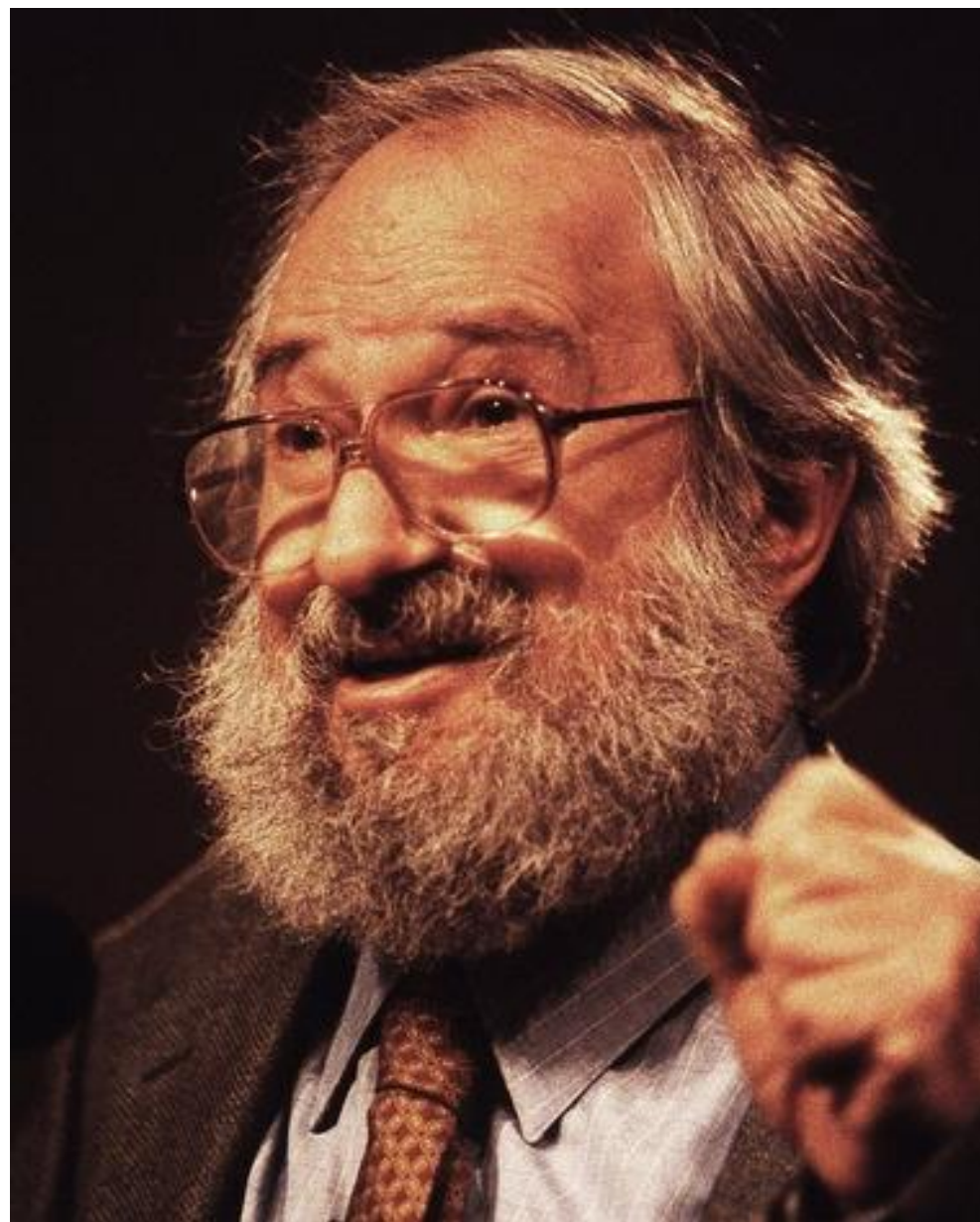


Seymour Papert
Сеймур Пейперт

1928 – 2016

математик,
программист,
психолог и педагог.

Создать языка
программирования
LOGO.



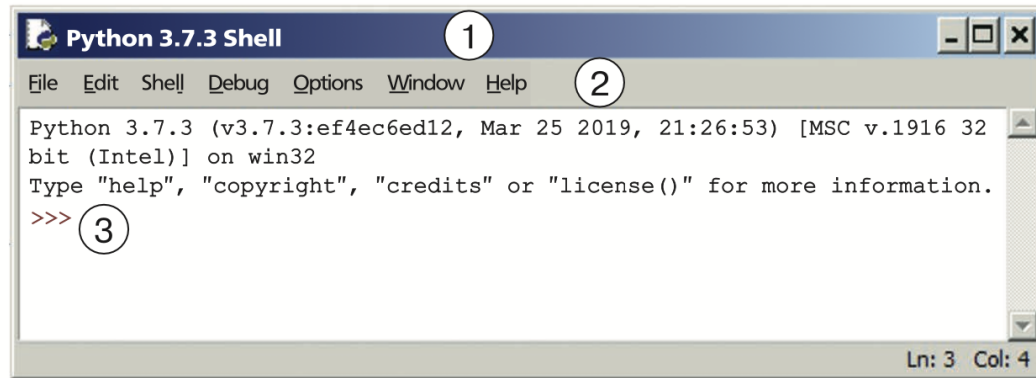
Пуск → Все программы → IDLE (Python 3).

У IDLE есть два окна:

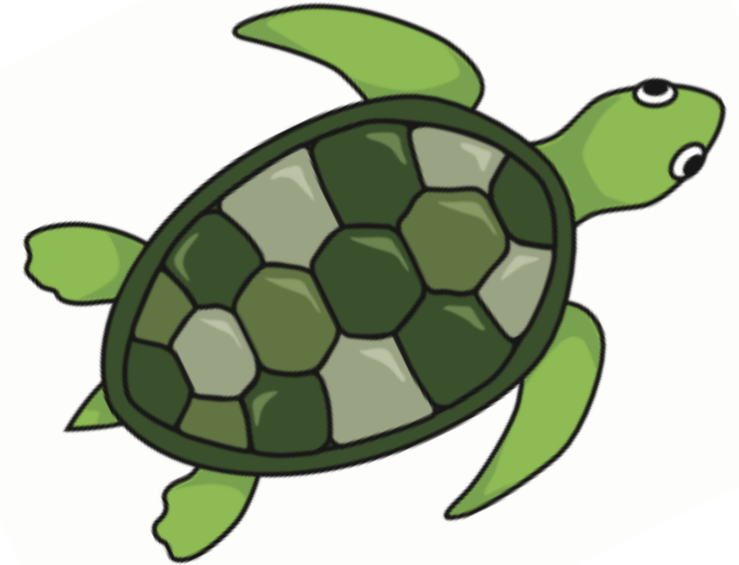
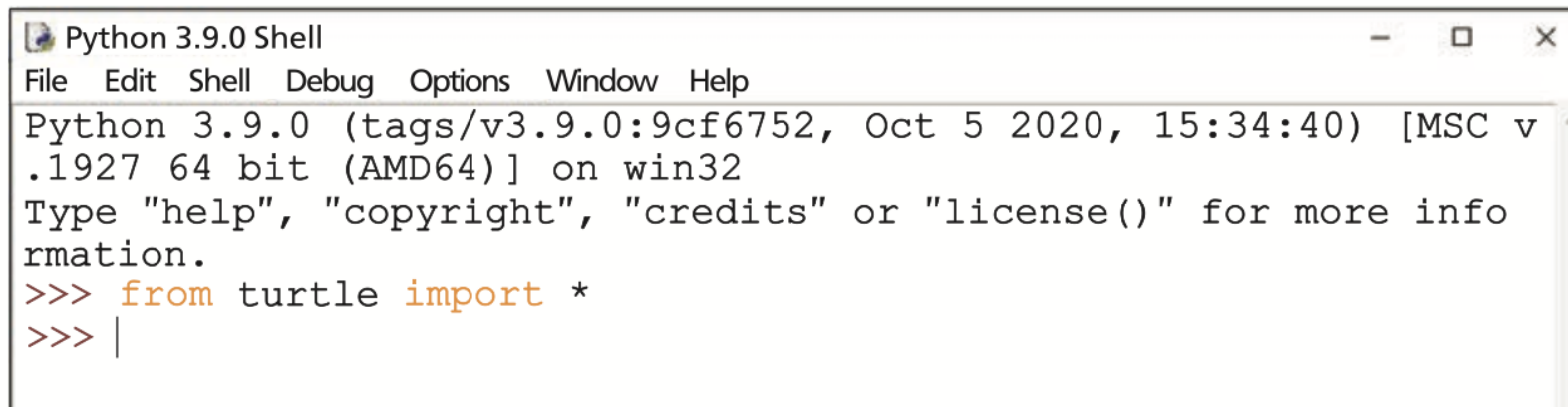
- 1) окно консоли для выполнения отдельных команд;
- 2) окно программы, в котором пишут всю программу.

Познакомимся с командным режимом работы.

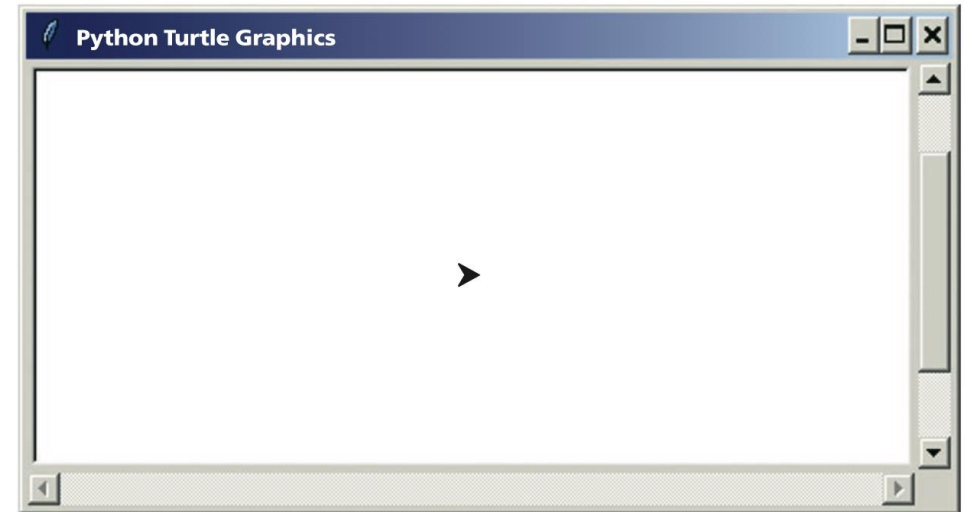
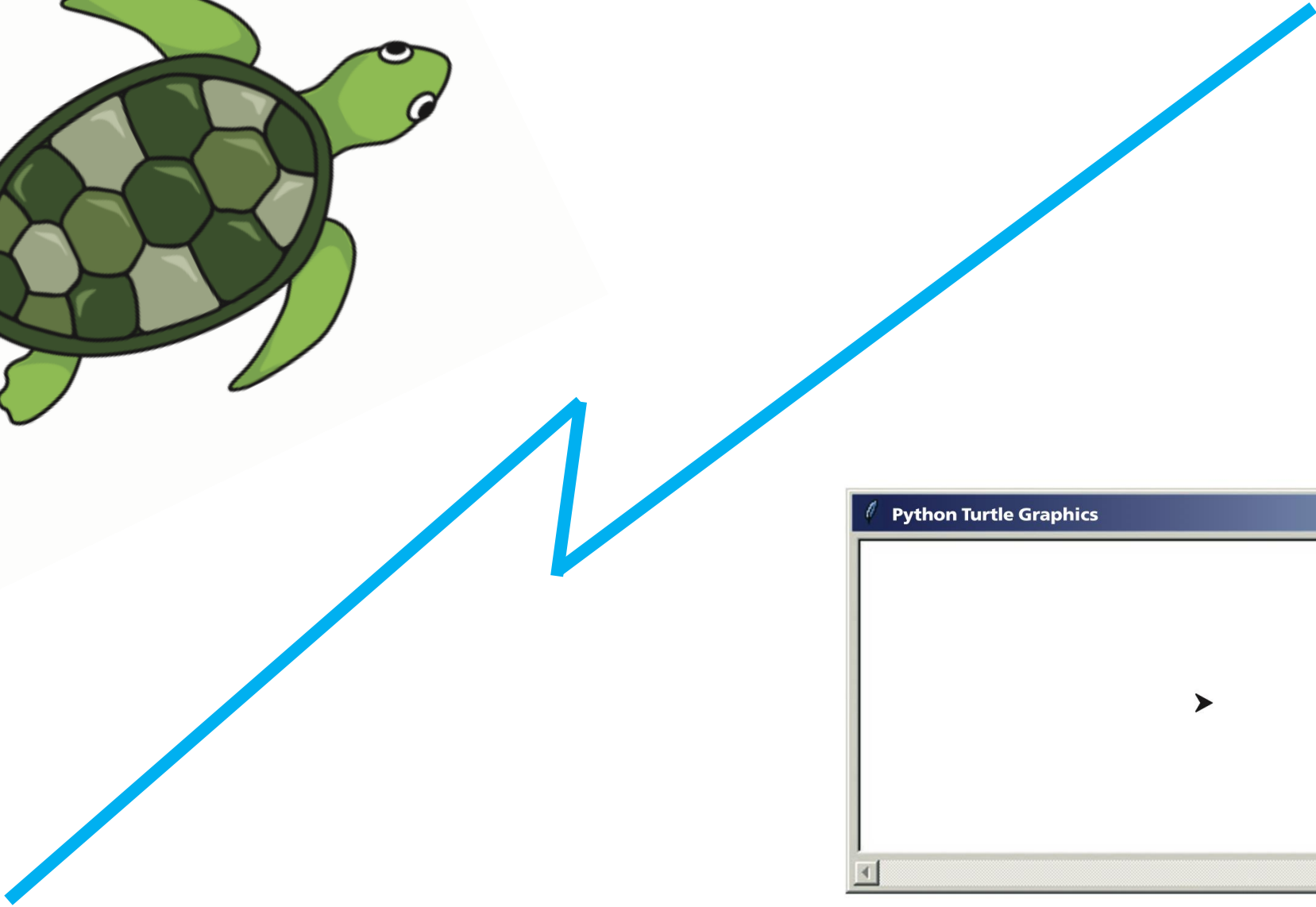
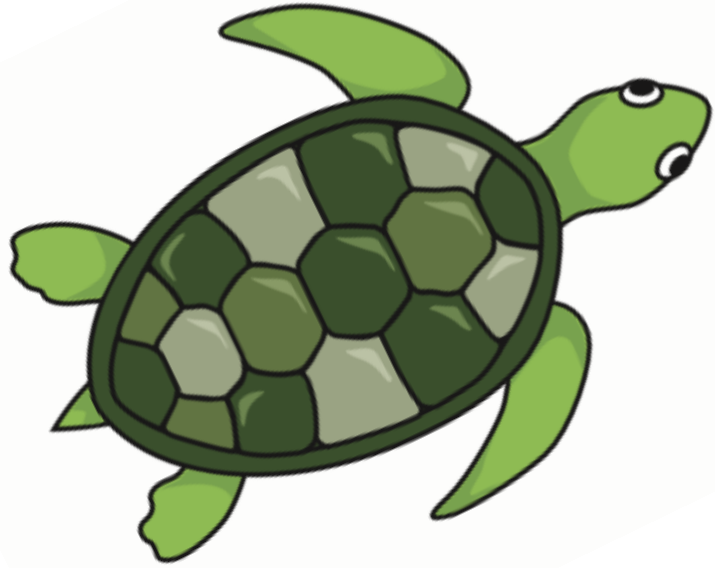
Рассмотрим окно консоли.

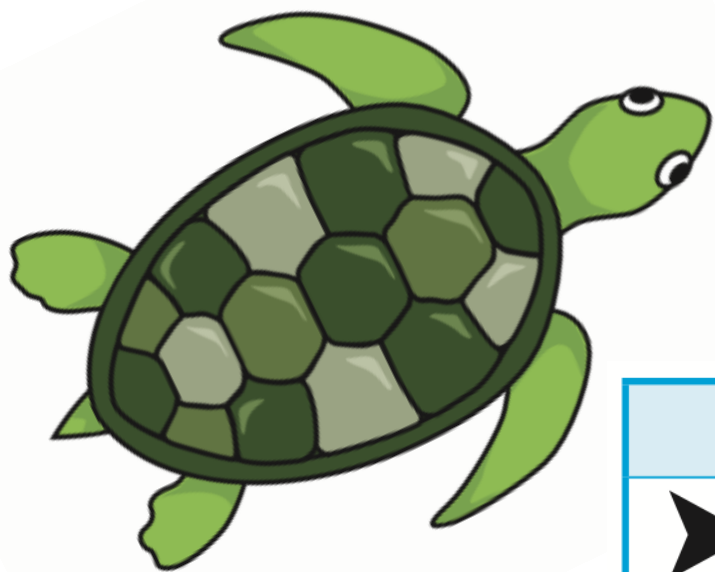


Окно консоли: 1 — строка заголовка, 2 — строка меню,
3 — строка для ввода команд

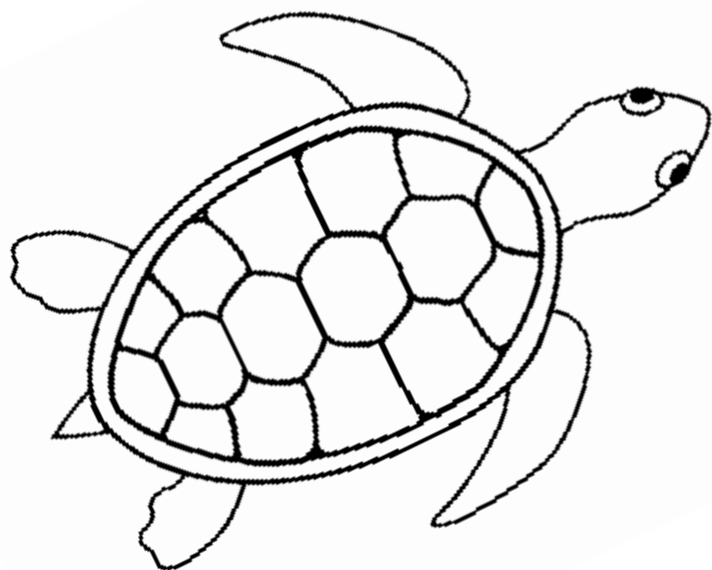


reset()





Форма Черепашки	Команда выбора формы Черепашки
 стрелочка (по умолчанию)	<code>shape ('classic')</code>
 стрелка	<code>shape ('arrow')</code>
 круг	<code>shape ('circle')</code>
 квадрат	<code>shape ('square')</code>
 треугольник	<code>shape ('triangle')</code>
 черепашка	<code>shape ('turtle')</code>



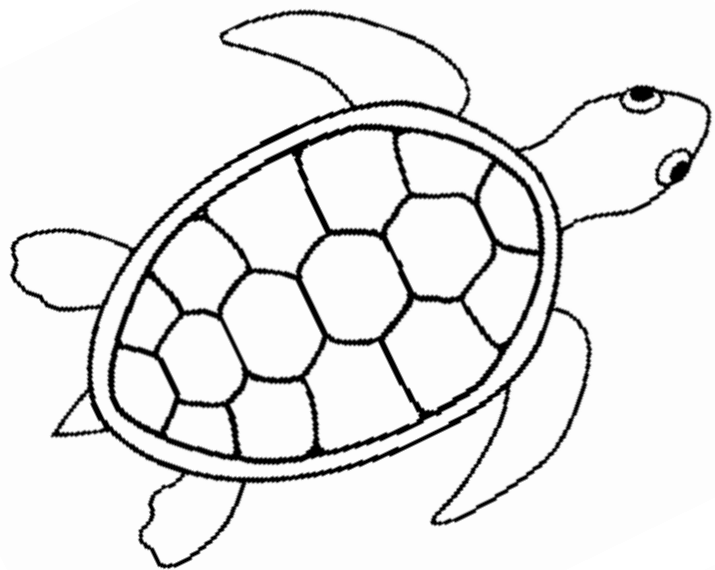
`color(s1, s2)`

цвет контура

цвет заливки

white ●, black ●, gray ●, navy ●,
blue ●, cyan ●, green ●, yellow ●,
red ●, orange ●, brown ●, violet ●,
maroon ●, purple ●

```
>>> color('brown', 'red')
```

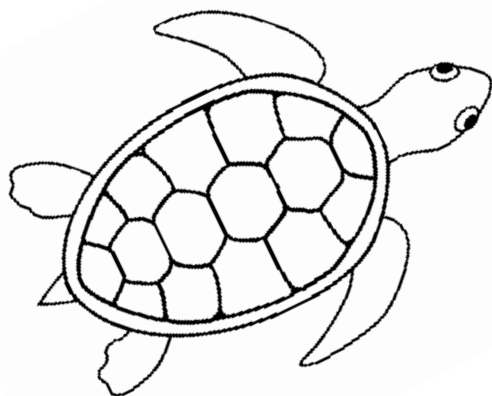
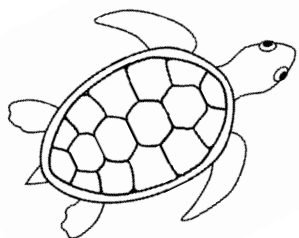


`shape_size(m, n, k)`

ширина

высота

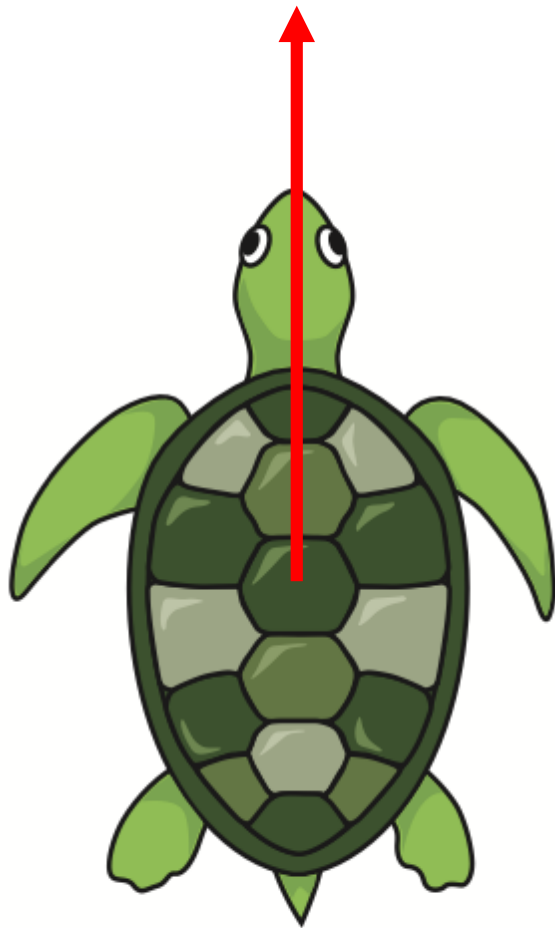
толщина контура



```
>>> shape_size(2)
```

```
>>> shape_size(2, 4)
```

```
>>> shape_size(2, 4, 2)
```



<code>forward(n)</code>	Перемещение Черепашки вперёд на n пикселей
<code>backward(n)</code>	Перемещение Черепашки назад на n пикселей
<code>left(m)</code>	Поворот Черепашки налево (против часовой стрелки) на m градусов
<code>right(m)</code>	Поворот Черепашки направо (по часовой стрелке) на m градусов

Можно пользоваться сокращённой формой записи этих команд:

<code>fd(n)</code>	Перемещение Черепашки вперёд на n пикселей
<code>bk(n)</code> или <code>back(n)</code>	Перемещение Черепашки назад на n пикселей
<code>lt(m)</code>	Поворот Черепашки налево (против часовой стрелки) на m градусов
<code>rt(m)</code>	Поворот Черепашки направо (по часовой стрелке) на m градусов

Очистите экран с помощью команды `reset()`. Черепашка возвращается домой — в центр экрана.

Вводите после приглашения команды перемещения Черепашки и наблюдайте за изменением её положения на холсте:

```
>>> forward(100)
```

```
>>> left(45)
```

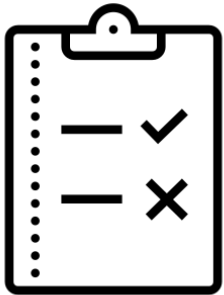
```
>>> backward(50)
```

```
>>> right(180)
```

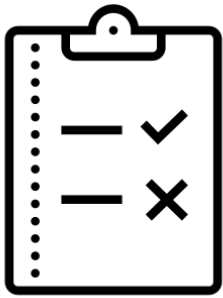
<code>down()</code>	Опускает перо, чтобы оставлять след при перемещении; по умолчанию перо опущено
<code>up()</code>	Поднимает перо
<code>width(n)</code>	Устанавливает ширину следа Черепашки в n пикселей
<code>bgcolor(s)</code>	Устанавливает цвет фона; по умолчанию цвет фона белый
<code>color(s)</code>	Устанавливает цвет следа Черепашки и её цвет; по умолчанию цвет чёрный
<code>stamp()</code>	Оставляет отпечаток текущей формы Черепашки на холсте
<code>speed(n)</code>	Устанавливает скорость Черепашки от 1 (медленно) до 10 (быстро) или 0 (мгновенно)
<code>reset()</code>	Очищает экран, убирает все настройки, возвращает Черепашку домой — в центр холста
<code>clear()</code>	Очищает экран, удаляет все рисунки Черепашки с экрана. При этом Черепашка не перемещается, её состояние и положение не изменяются

Введите после приглашения поочерёдно следующие команды:

```
>>> reset()  
>>> forward(100)  
>>> left(90)  
>>> forward(100)
```



Какие ещё команды надо отдать Черепашке, чтобы на холсте было получено изображение квадрата?



Запишем последовательность команд, с помощью которых Черепашка изобразит на экране прямоугольник со сторонами 100 и 60 пикселей.



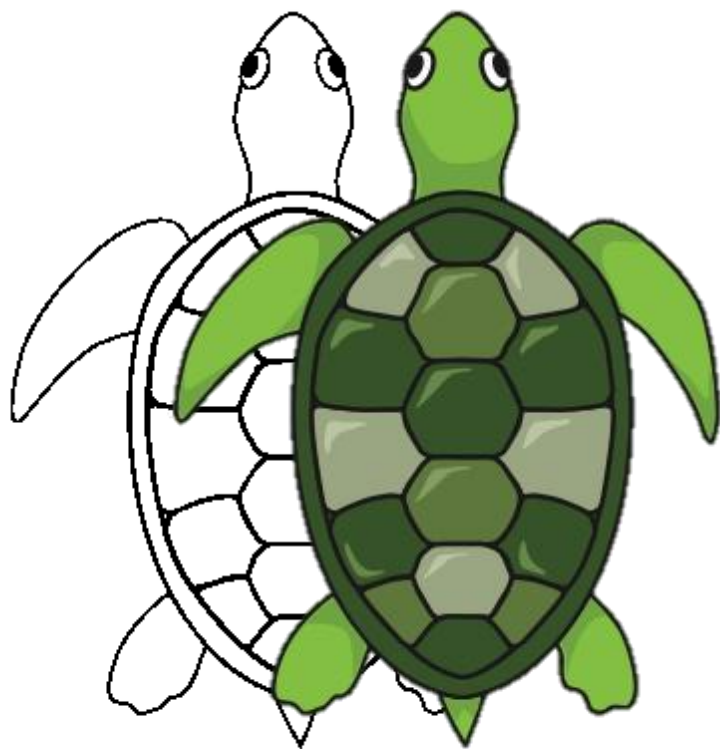
Очистите экран с помощью команды `reset()`. Черепашка возвращается домой — в центр экрана.

Вводите после приглашения команды перемещения Черепашки и наблюдайте за изменением её положения на холсте: Очистите экран с помощью команды `reset()`.

Введите после приглашения поочерёдно следующие команды:

```
>>> reset()
>>> shape('turtle')
>>> fd(100)
>>> rt(90)
>>> fd(60)
>>> rt(90)
>>> fd(100)
>>> rt(90)
>>> fd(60)
```

`stamp()`



После ввода команды `stamp()` в графическом окне не будет изменений, так как поверх отпечатанной на экране копии находится сама Черепашка. Но можно увидеть, что в окне консоли появилось целое число, это идентификатор (номер)

Дополните вашу программу:

```
>>> stamp()
```

```
>>> fd(100)
```

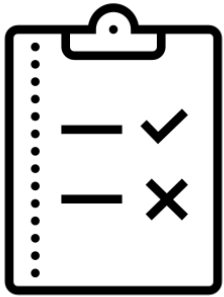
```
>>> stamp()
```

и повторите последние две команды ещё раз.

Уберём первое изображение Черепашки, для этого будем использовать команду `clearstamp()`, указав в скобках номер копии.

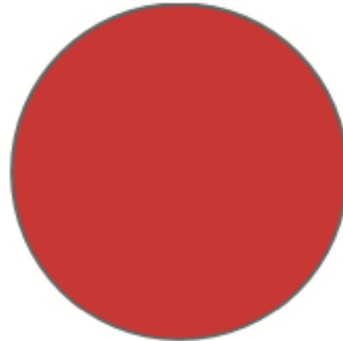
```
>>> stamp()  
14  
>>> fd(100)  
>>> stamp()  
15  
>>> fd(100)  
>>> stamp()  
16  
>>> clearstamp(14)  
>>> |
```

Обратите внимание, что на рисунке номер первого штампа Черепашки равен 14. В вашем окне он может иметь другое значение. Вы можете выбрать собственное значение штампа для удаления.



Напишем программу для изображения дорожного знака «Въезд запрещён». Для этого разделим задачу на две части:

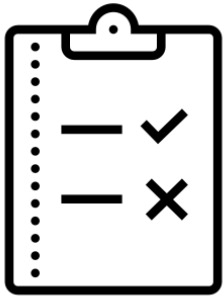
- 1) создадим большой красный круг;
- 2) поместим узкий белый прямоугольник в круг (рис. 8).



шаг 1

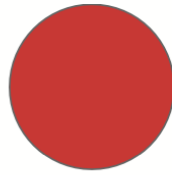


шаг 2



Напишем программу для изображения дорожного знака «Въезд запрещён». Для этого разделим задачу на две части:

- 1) создадим большой красный круг;
- 2) поместим узкий белый прямоугольник в круг (рис. 8).



шаг 1



шаг 2

В новом окне введите полный текст программы изображения дорожного знака «Въезд запрещён»:

```
>>> from turtle import *
>>> reset()
>>> shape('circle')
>>> color('black', 'red')
>>> shapesize(30)
>>> stamp()
>>> color('white')
>>> shape('square')
>>> shapesize(5, 10)
>>> shapesize(5, 15)
>>> shapesize(5, 20)
>>> stamp()
```

Командой `right(m)` мы поворачиваем Черепашку направо (по часовой стрелке) на m градусов. Командой `left(m)` мы поворачиваем Черепашку налево (против часовой стрелки) на m градусов.

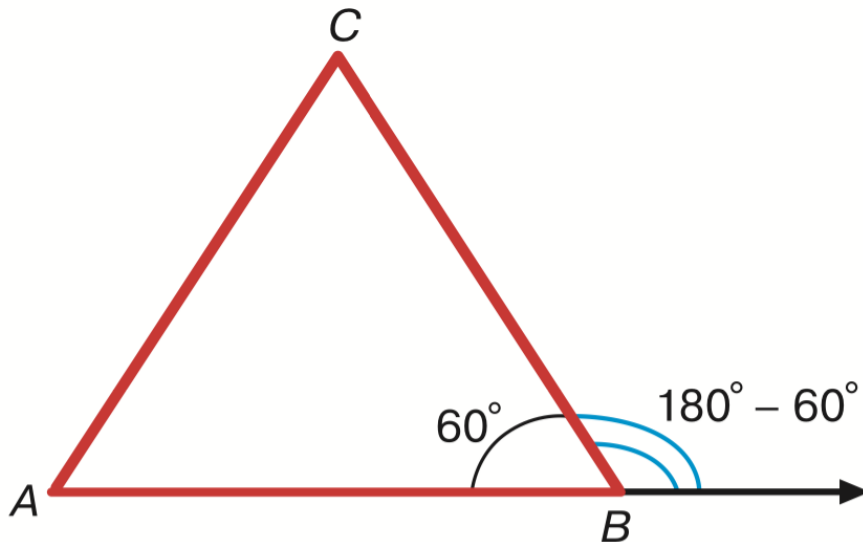
Развёрнутый угол образован двумя лучами. Стороны этого угла составляют прямую линию, на которой лежит вершина развёрнутого угла. Величина развёрнутого угла равна 180° .



Пример 1

Предположим, мы хотим изобразить равносторонний треугольник . Каждый из углов равностороннего треугольника равен 60° . Предположим, сторона нашего треугольника равна 150.

Начав перемещение из точки A , в результате выполнения команды `fd(150)` Черепашка окажется в точке B . Чтобы построить угол в 60° , Черепашка должна сделать поворот на 120° ($180^\circ - 60^\circ = 120^\circ$).



В новом окне введите полный текст программы изображения равностороннего треугольника:

```
>>> from turtle import *
>>> width(5)
>>> color('red')
>>> fd(150)
>>> lt(120)
>>> fd(150)
>>> lt(120)
>>> fd(150)
>>> lt(120)
```

Для изображения угла величиной m градусов надо использовать команду поворота на угол величиной $180 - m$.

Пример 2

Когда Черепашка обходит любой выпуклый многоугольник и возвращается в исходное положение, суммарный поворот, который она при этом совершает, равен 360° . Воспользуемся этим фактом, чтобы разработать алгоритм изображения правильного пятиугольника. В правильном пятиугольнике все углы и все стороны равны (рис. 10).

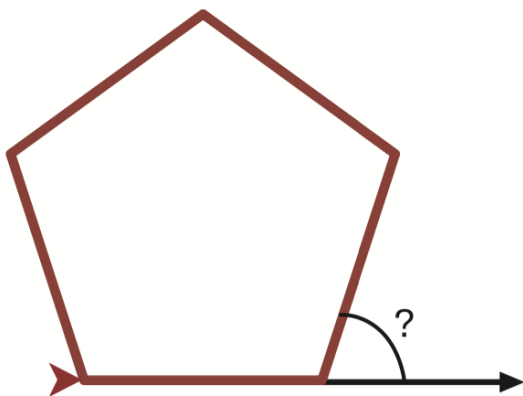


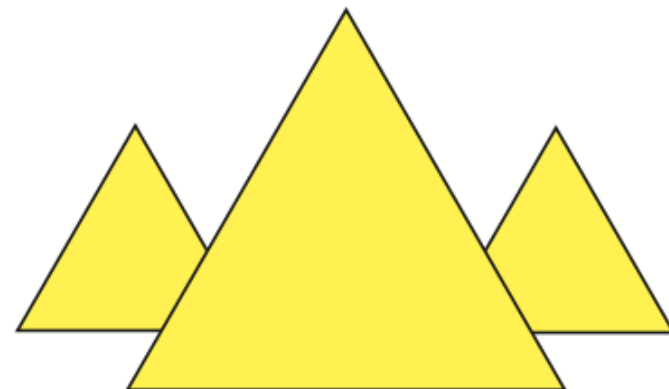
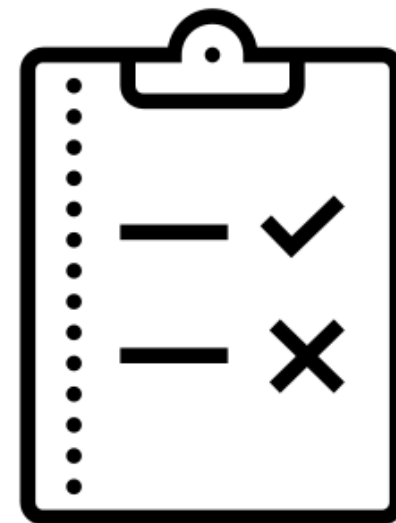
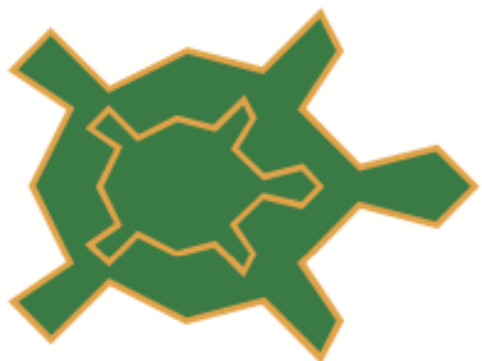
Рис. 10. Правильный пятиугольник

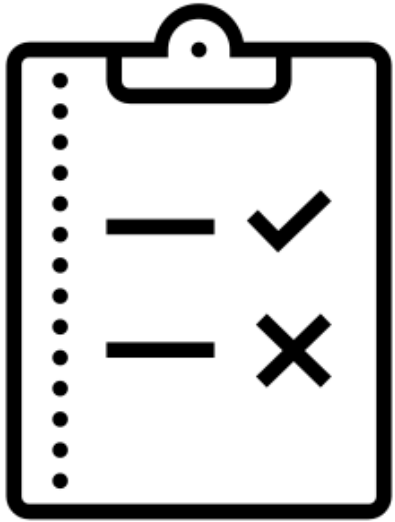
Чтобы изобразить правильный пятиугольник, надо пять раз построить сторону определённой длины и выполнить пять таких одинаковых поворотов, чтобы Черепашка повернулась на 360° . $360^\circ / 5 = 72^\circ$.

В новом окне введите полный текст программы изображения правильного пятиугольника:

```
>>> from turtle import *
>>> width(5)
>>> color('brown')
>>> fd (100)
>>> lt(72)
>>> fd (100)
>>> lt(72)
>>> fd (100)
>>> lt(72)
>>> fd (100)
>>> lt(72)
>>> fd (100)
>>> lt(72)
```

Угол поворота при изображении правильного n -угольника рассчитывается по формуле: угол поворота = $360 / n$.



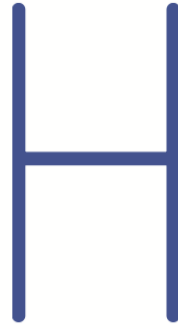


Используйте команды `forward(n)`, `left(m)`, `right(m)`, `width(n)`, `bgcolor(s)`, `color(s)` для построения на экране следующего изображения.

а)



б)



в)



г)



д)

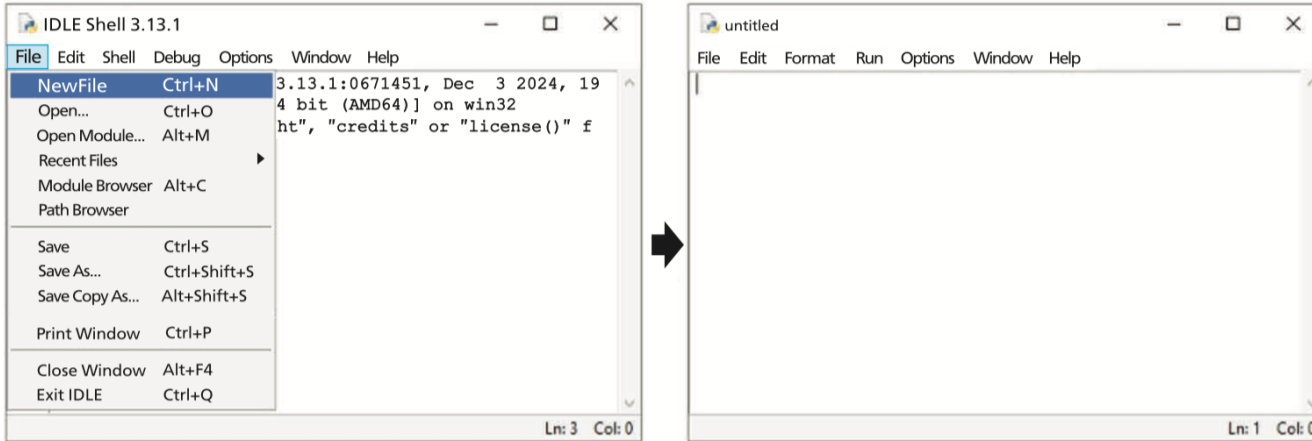


е)



Рекомендация. Для выполнения рисунков а)–е) установите высоту 100, ширину 50.

File → New File



```
from turtle import *  
width(3)  
fd(100)  
rt(90)  
fd(50)  
lt(90)  
fd(100)
```

Далее программу следует сохранить с помощью команды

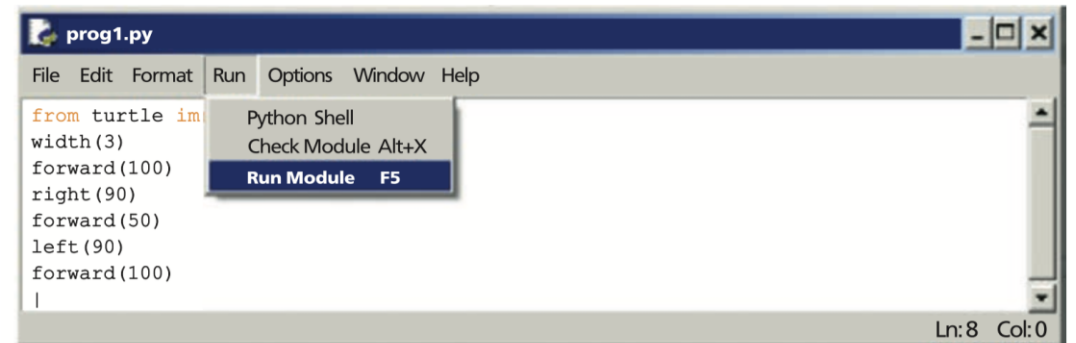
File → Save,

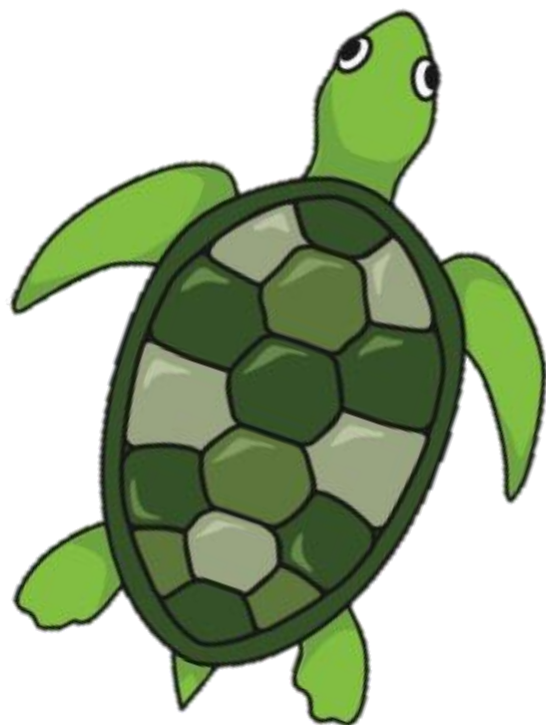
указав имя файла, например prog1.py. Расширение py файл получит автоматически.

С помощью команды

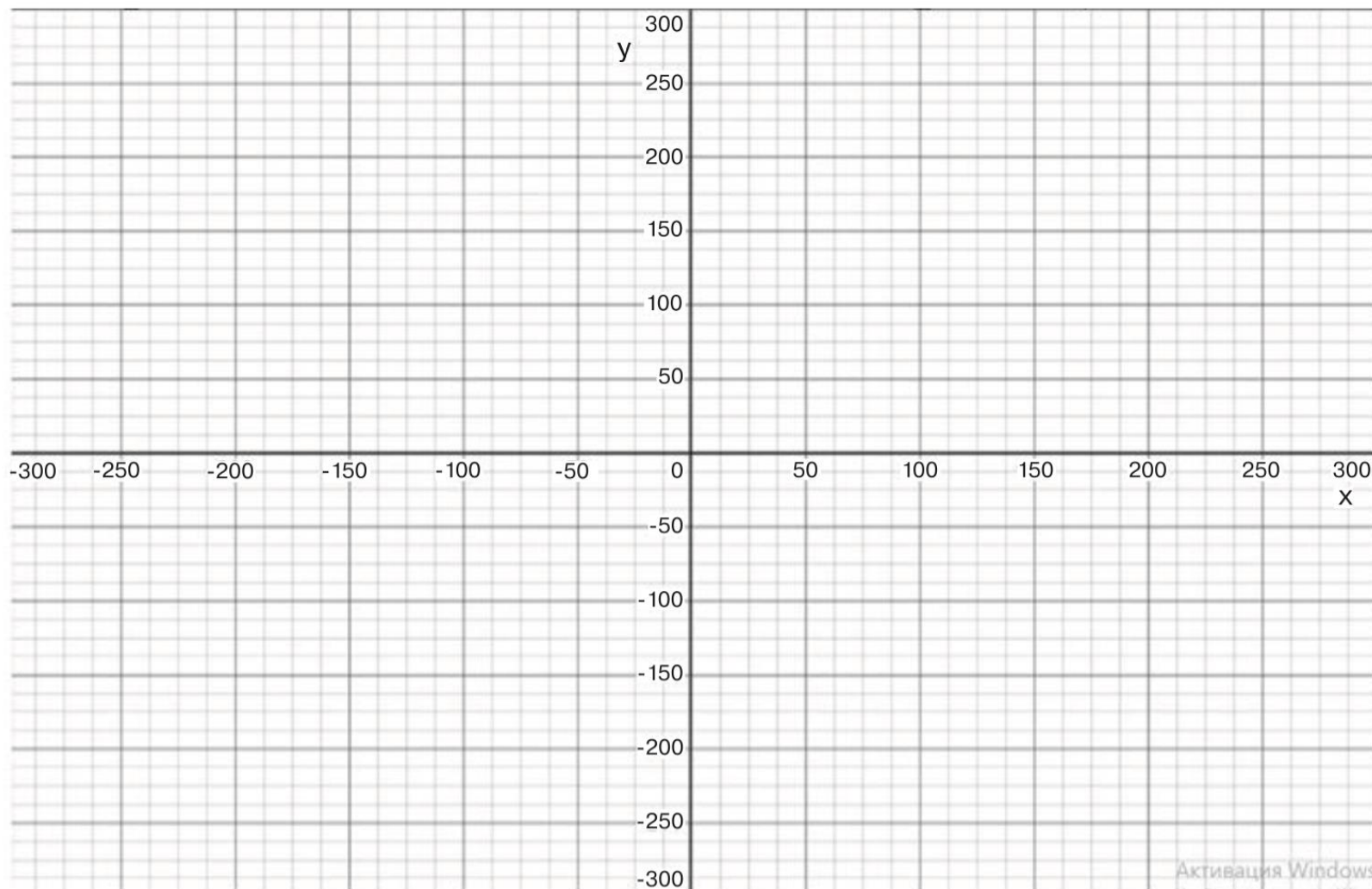
Run → Run Module

программа отправляется на выполнение (рис. 13).





Наша Черепашка прекрасно разбирается в координатах. Начало координат в графическом окне для Черепашки находится в центре окна. Размер графического окна вы можете установить с помощью команды `setup(x, y)`. Например, командой `setup(600, 600)` устанавливается размер окна для рисования 600×600 пикселей (рис. 16).



Для перемещения в точку с заданными координатами предназначены команды:

<code>goto (x, y)</code>	Перемещение Черепашки в точку с координатами (x, y)
<code>setx (x)</code>	Установка значения координаты x Черепашки; координата y остаётся без изменений
<code>sety (y)</code>	Установка значения координаты y Черепашки; координата x остаётся без изменений
<code>home ()</code>	Возвращение Черепашки домой — в точку с координатами (0, 0)

Подключите модуль черепашьяй графики:

```
from turtle import *
```

Установите размер графического окна 400х400 пикселей:

```
setup(400, 400)
```

Выберите для Черепашки форму круга:

```
shape('circle')
```

Поднимите перо и переведите, не оставляя следа, Черепашку в точку с координатами (-150; 110):

```
up()
```

```
goto(-150, 110)
```

Опустите перо, чтобы при дальнейших перемещениях Черепашка оставляла след, и отдайте ей команду переместиться в точку с координатами (-90; 90), а затем в точки с координатами (-60; 60) и (-20; 40):

```
down()
```

```
goto(-90, 90)
```

```
goto(-60, 60)
```

```
goto(-20, 40)
```

Измените только значение координаты x, координата y остаётся без изменений. Переместите Черепашку в точки с координатами (60; 40), (50; 0), (-10; 0) и (-20; 40):

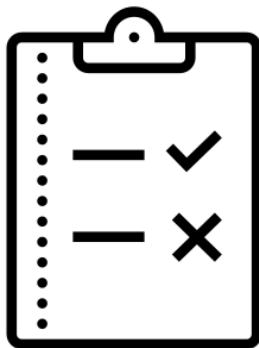
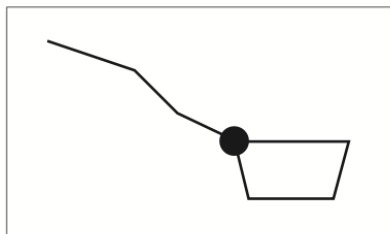
```
setx(60)
```

```
goto(50, 0)
```

```
setx(-10)
```

```
goto(-20, 40)
```

После сохранения и запуска программы у вас должно получиться такое изображение:



Запишем программу, с помощью которой Черепашка изобразит на экране ковш Большой Медведицы, созвездия северного полушария.

Если Черепашка перемещается по экрану слишком быстро, то можно изменить её скорость с помощью команды `speed(n)`. Скорость может изменяться от 1 (медленно) до 10 (быстро). Если в скобках указать 0, то перемещение будет мгновенным.

Изменим ширину следа, оставляемого Черепашкой, с помощью команды `width(n)`. Добавим команду в начало программы перед поднятием пера.

Добавьте в программу команды, чтобы её начало выглядело так:

```
from turtle import *  
setup(400, 400)  
shape('circle')  
speed(1)  
width(2)  
up()
```

С помощью команды `bgcolor()` установим цвет фона, по умолчанию он белый. Командой `color()` изменим цвет следа Черепашки и её цвет.

▶ Добавьте следующие команды в программу:

```
bgcolor('navy blue')
```

```
color('yellow')
```

Теперь наше изображение выглядит так:



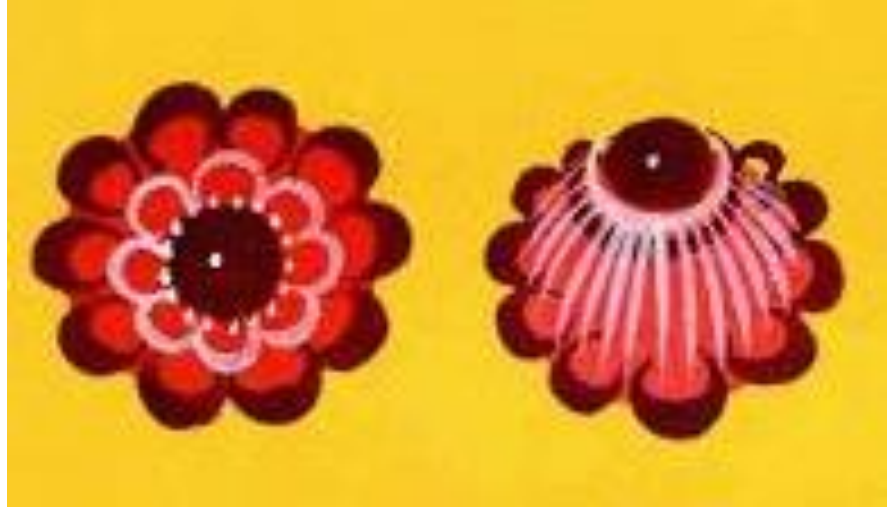
▶ С помощью команды `stamp()` дополните программу, чтобы получить следующее изображение:



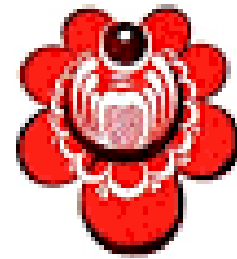
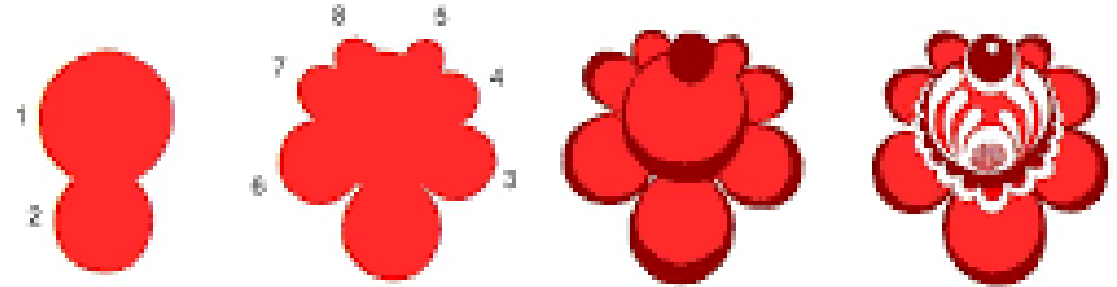


Проект «Городецкая роспись». Возникшая в Нижегородской губернии в середине XIX века, городецкая роспись своими яркими красками и сюжетами украшала предметы быта. Закрепите полученные знания трёх уроков — составьте программу для рисования панно «Городецкая роспись». Варианты представлены на рисунке.





ГОРОДЕЦКАЯ РОЗА



Купавка

