



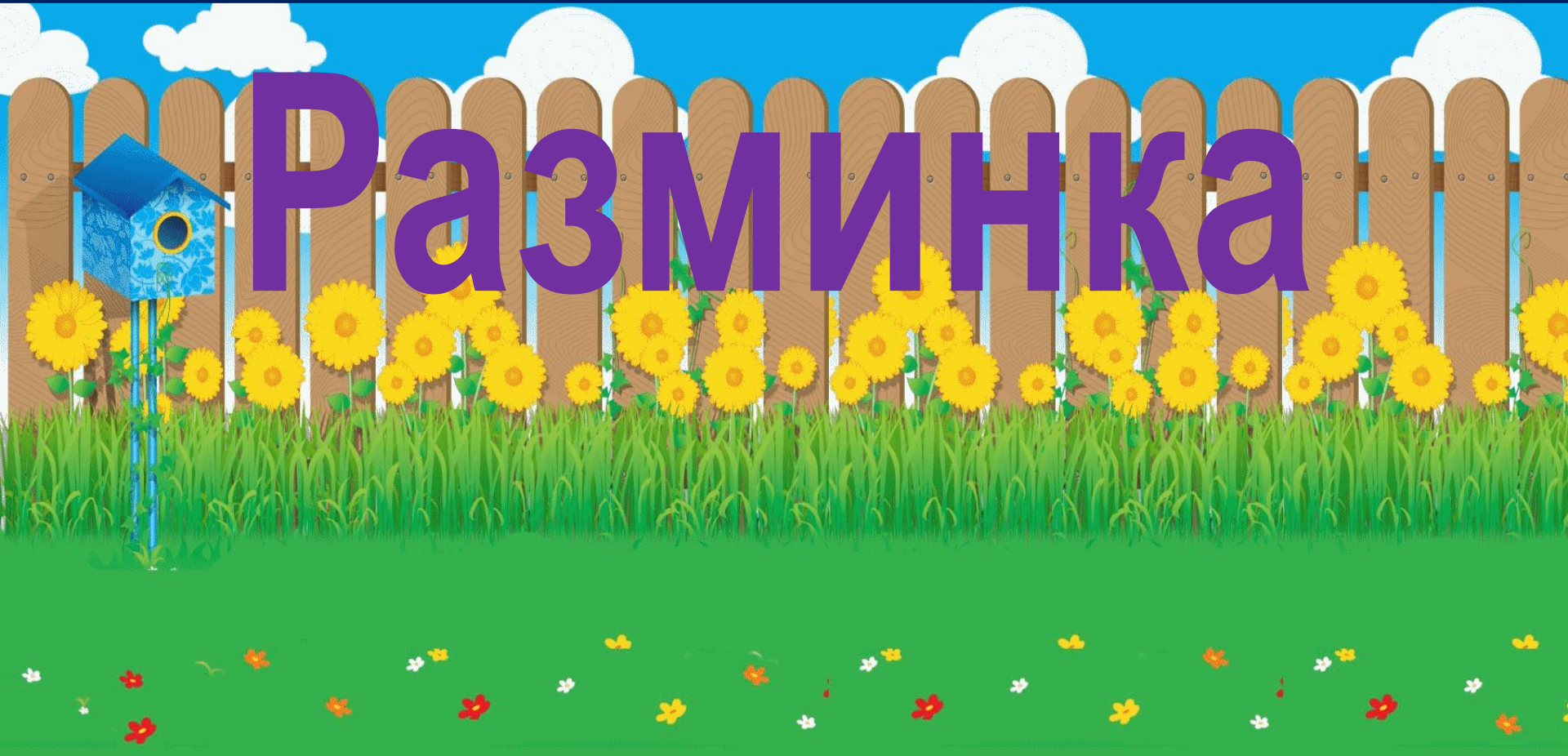
ИНФОРМАТИКА





ИНФОРМАТИКА

Разминка

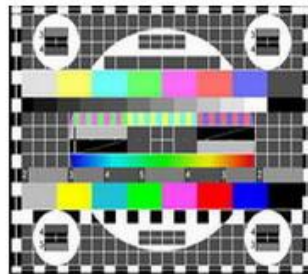




ИНФОРМАТИКА



“БЮ



” $\frac{p}{q}$



 3 = P





ИНФОРМАТИКА

КОМПЬЮТЕРНАЯ

ИГРА

Программирование

Глава

3

Учебник
«ИНФОРМАТИКА 7-9 КЛАСС»

И. Н. Цыбуля, Л. А. Самыкбаева,
А. А. Беляев, Н. Н. Осипова, У. Э. Мамбетакунов

Урок №25

3.9.Тема:
***«Черепашья графика» в
Python. Создаём свои
команды (функции).***



1

**Циклы в «Черепашьей графике».
Решение практических задач**

2

**Создаём свои команды. Функции в
«Черепашьей графике»**

3

Решение практических задач



Цикл *while*

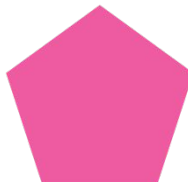
```
while True:  
    <тело цикла>  
    if <условие цикла>:  
        break
```



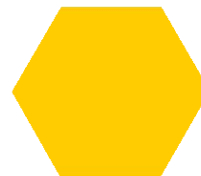
1



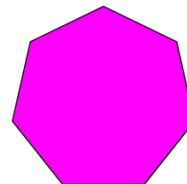
2



3



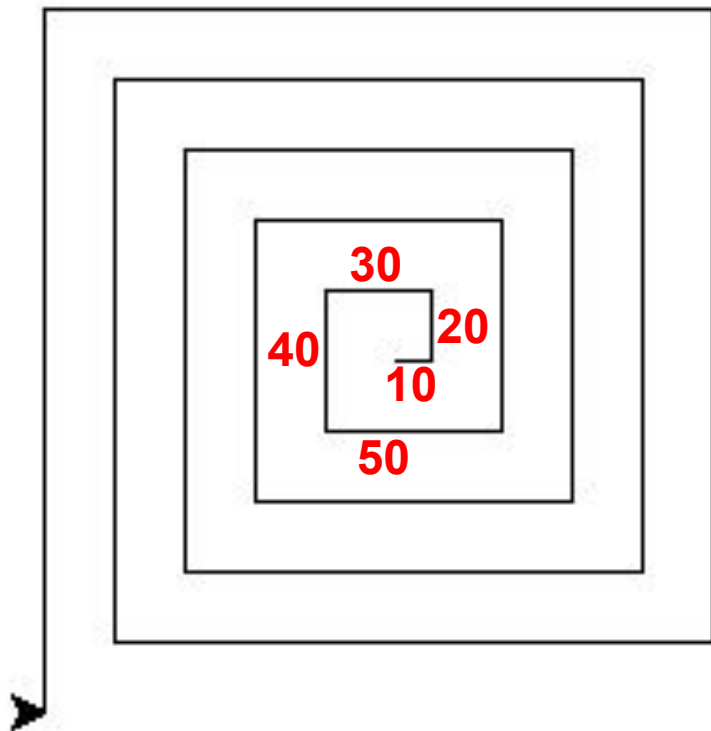
4



5



Пример1. Рисуем спираль



$10+10=20+10=30+10=40+10=50+\dots$

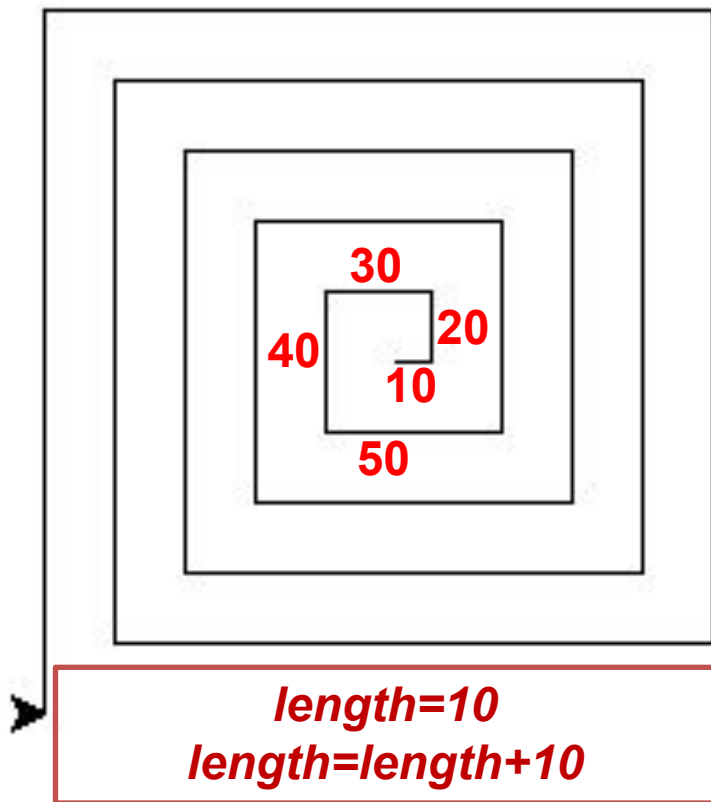
Увеличивается каждый раз на 10
пикселей!

$length=10$ # переменная, обозначающая
длину стороны

$length=length+10$



Пример1. Рисуем спираль



Переменная
«длина»

Счетчик

Увеличиваем
длину на 10
пикселей

Увеличиваем
счётчик на
единицу

```
from turtle import*
```

```
length=10
```

```
count=0
```

```
while count<20:
```

```
    fd(length)
```

```
    lt(90)
```

```
    length=length+10
```

```
    count=count+1
```

```
exitonclick()
```



Пример1. Рисуем спираль

File Edit Format Run Options Window Help

```
from turtle import *
```

```
speed(1) # скорость, с которой наша черепашка рисует (0-10)
```

```
#1- минимальная скорость (самое медленное построение рисунка)
```

```
#10 - максимальное значение, которое может принимать данная команда
```

```
# 0 - мгновенное отображение рисунка
```

```
length=10 # length - переменная, обозначающая длину стороны
```

```
count=0 # count - количество витков в спирали
```

```
while count<20:
```

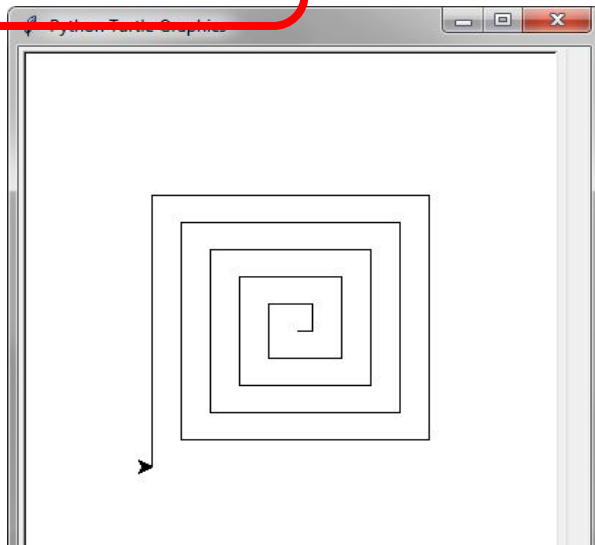
```
    fd(length)
```

```
    lt(90)
```

```
    length=length+10
```

```
    count=count+1
```

```
exitonclick()
```

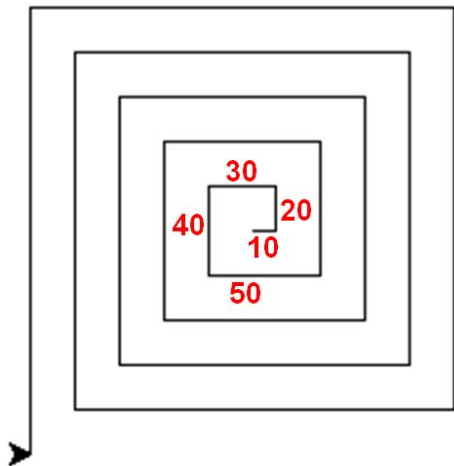




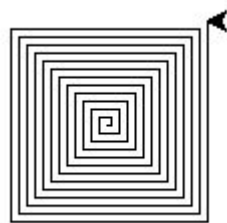
Размеры спирали



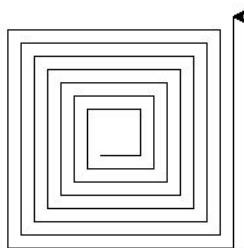
Цикл *while*



```
length=10  
length=length+10
```



```
length=2  
length=length+2
```



```
length=30  
length=length+5
```

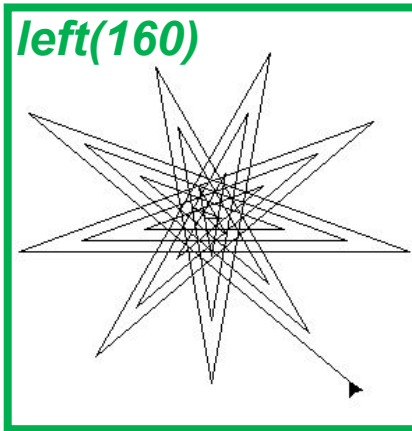
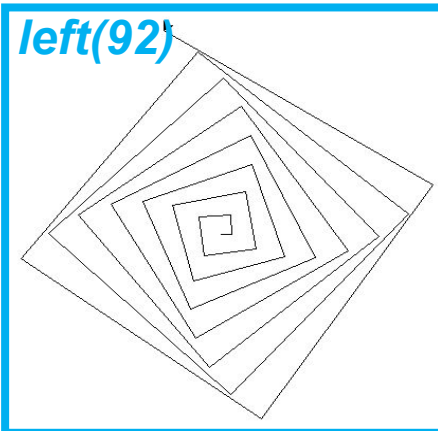
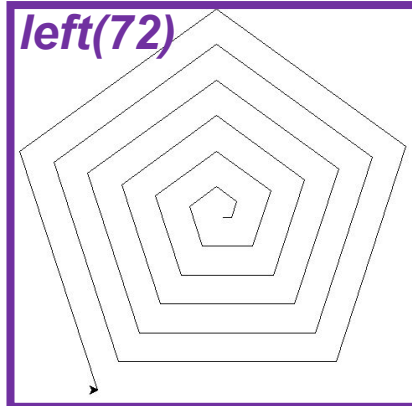
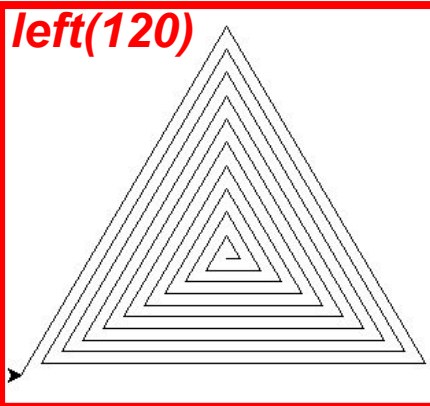
```
from turtle import*  
length=10  
count=0  
while count<50:  
    fd(length)  
    lt(90)  
    length=length+10  
    count=count+1  
exitonclick()
```



ПРОГРАММИРОВАНИЕ. 3.9. «Работа с графикой в Python»



Цикл **while**



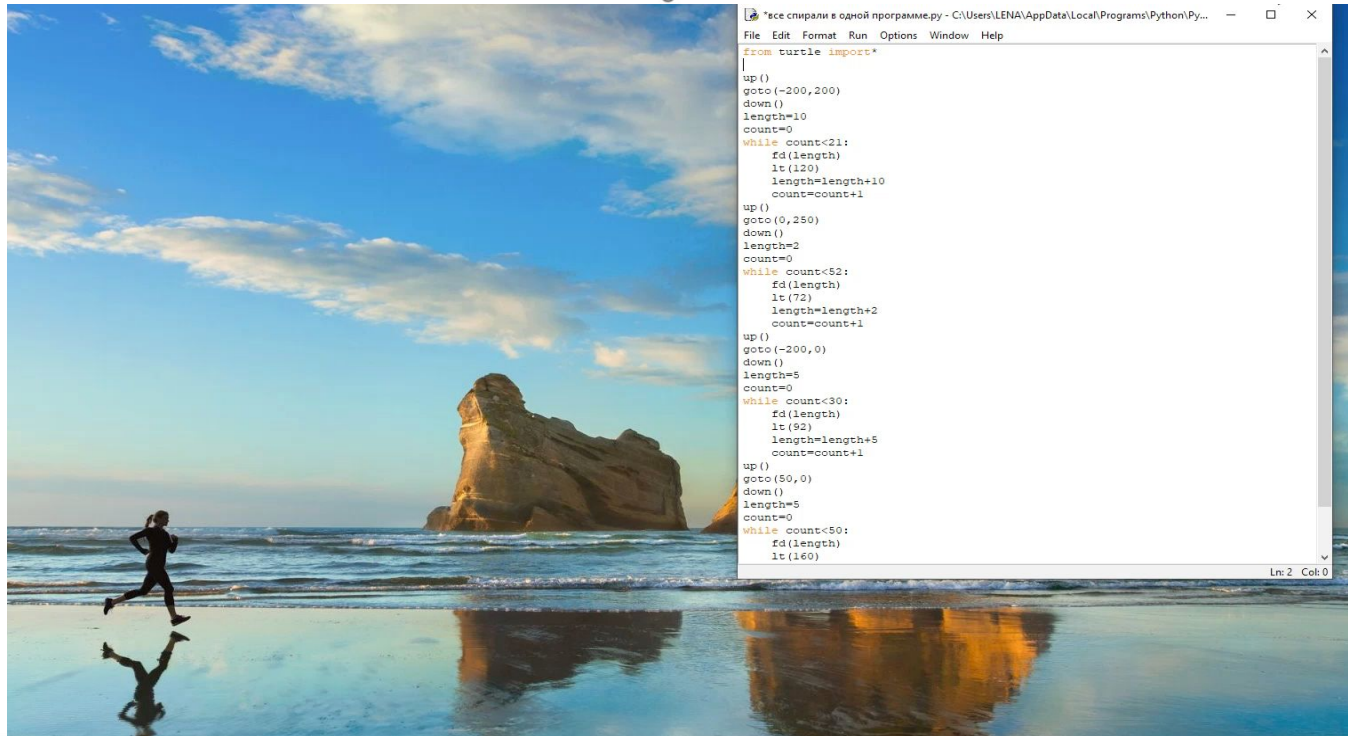
Угол поворота

```
from turtle import*  
length=10  
count=0  
while count<30:  
    fd(length)  
    left(?)  
    length=length+10  
    count=count+1  
exitonclick()
```



ПРОГРАММИРОВАНИЕ.

3.9. «Работа с графикой в Python»





Движение со случайной длиной и поворотом

Начало программы



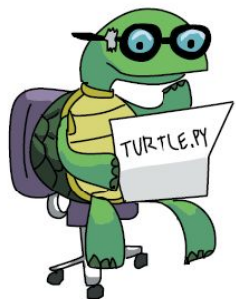
```
import turtle #подключить модуль turtle
import random #подключить модуль random
turtle.shape("turtle")
turtle.position()
(0.00,0.00) # задаем позицию относительно центра
turtle.forward(15) # задаём смещение
turtle.color('blue') # устанавливаем цвет
i=0
```

продолжение





Движение со случайной длиной и поворотом



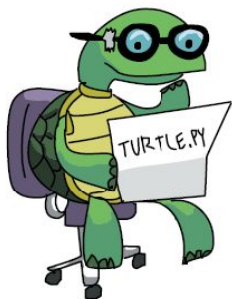
```
while i <= 100: # задаём цикл
    a=random.randint(1,20) # произвольная величина, на которую
    будем сдвигать
    turtle.forward(150+a) # смещаем на 150+произвольную
    величину
    turtle.left(90) # поворот влево на 90 градусов
    turtle.color('#000000')
    a=random.randint(1,35) # новая произвольная величина
    turtle.forward(50+a+i) # новое смещение
    turtle.left(90+a) #изменение угла поворота
```

продолжение





Движение со случайной длиной и поворотом



#остальные команды подобные

```
turtle.color('red')
```

```
a=random.randint(1,25)
```

```
turtle.forward(150+a)
```

```
turtle.left(90+a+i)
```

```
turtle.color('#000000')
```

```
a=random.randint(1,20)
```

```
turtle.forward(50+a)
```

```
turtle.left(90+a)
```

```
i=i+1
```

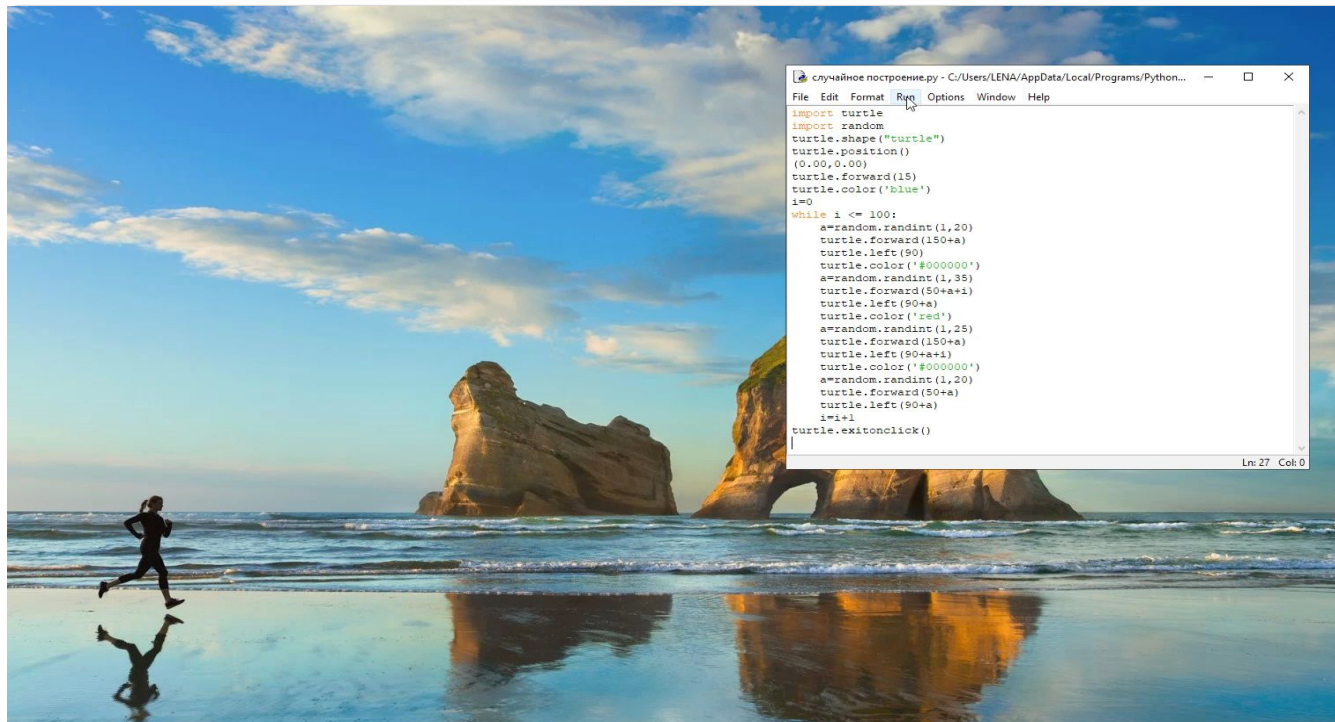
```
turtle.exitonclick()
```



ПРОГРАММИРОВАНИЕ.

3.9. «Работа с графикой в Python»

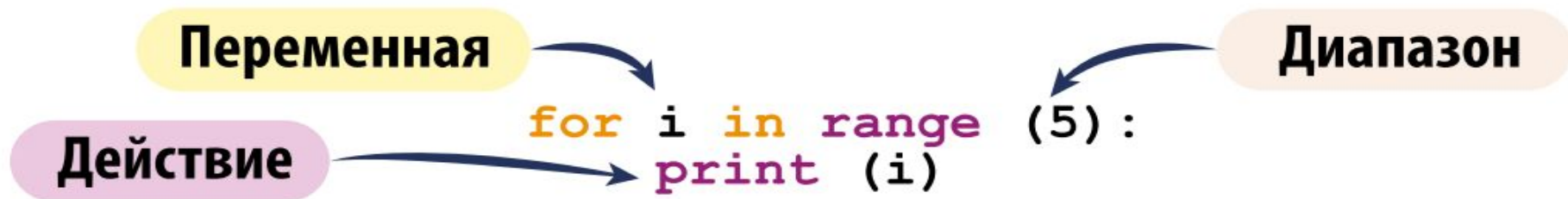
Движение со случайной длиной и поворотом





Циклические алгоритмы. Цикл с параметром.

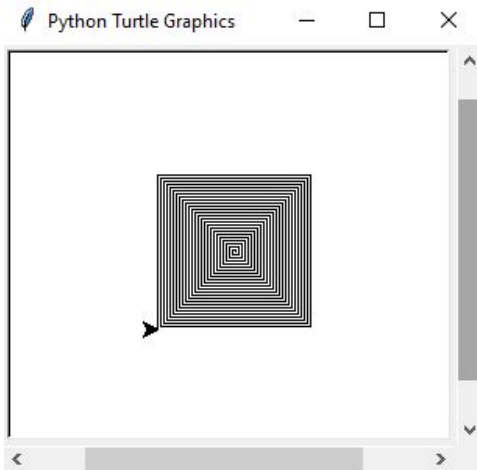
Цикл **for** в Python осуществляется по схеме:





Спирали. Цикл **for**

Наберите эти команды, используя цикл **for** и у вас получится квадратная спираль!

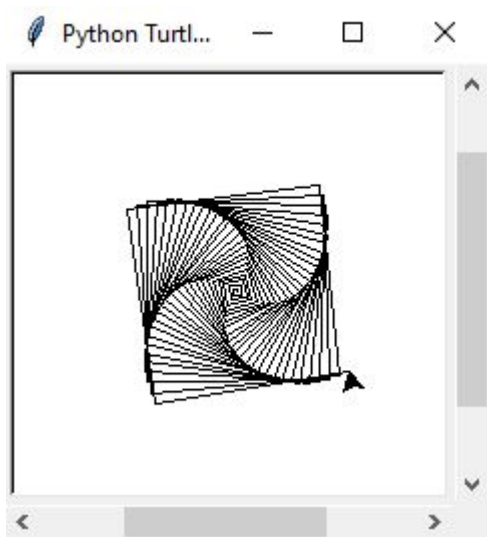


```
import turtle      #подключить модуль turtle  
t=turtle.Pen()    # t - обозначение ручки черепашки  
for x in range(100): # команда цикла в диапазоне от  
                    # 0 до 99  
    t.forward(x)     # идти вперёд x точек на экране  
    t.left(90)       # повернуться налево на 90 градусов
```

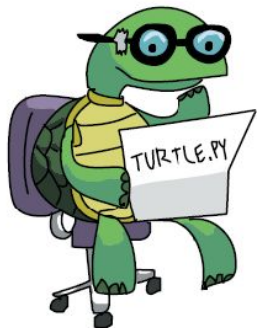


Спирали. Цикл *for*

Измените в последней строке программы угол поворота с **90°** на **91°** и вы получите спиралевидную фигуру.

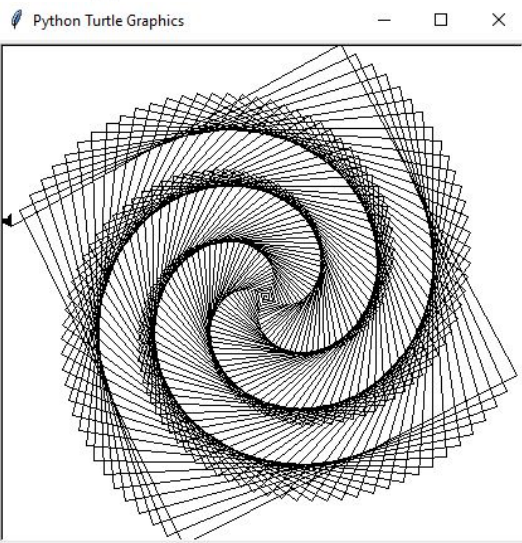


```
import turtle      #подключить модуль turtle
t=turtle.Pen()      # t - обозначение ручки черепашки
for x in range(100): # команда цикла в диапазоне от
                    # 0 до 99
    t.forward(x)    # идти вперёд x точек на экране
    t.left(90)      # повернуться налево на 91 градус
```



Спирали. Цикл **for**

Измените в третьей строке диапазон на **300**

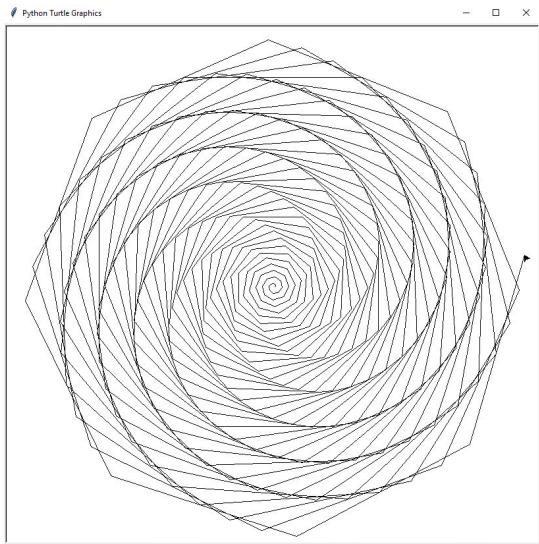


```
import turtle      #подключить модуль turtle  
t=turtle.Pen()    # t - обозначение ручки черепашки  
for x in range(300): # команда цикла в диапазоне от  
                    # 0 до 300  
    t.forward(x)    # идти вперёд x точек на экране  
    t.left( 91)      # повернуться налево на 91 градус
```



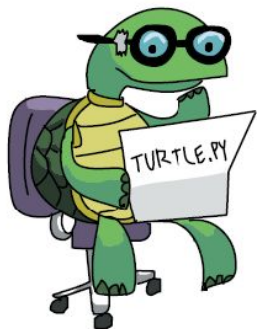
Спирали. Цикл *for*

Поставьте в последней строке программы угол поворота на **46** градусов



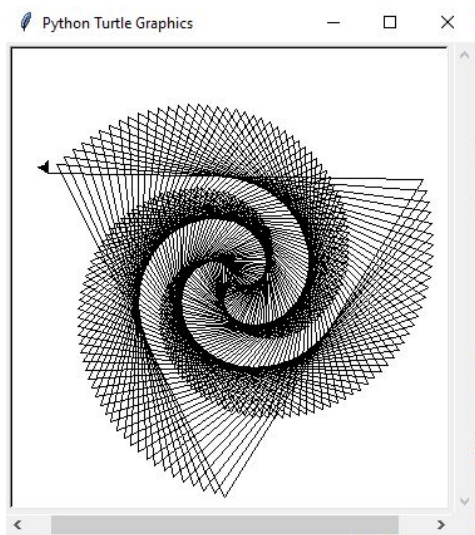
```
import turtle      #подключить модуль turtle
t=turtle.Pen()      # t - обозначение ручки черепашки
for x in range(300): # команда цикла в диапазоне от
                    0 до 300

    t.forward(x)    # идти вперёд x точек на экране
    t.left(46)      # повернуться налево на 46
                    градусов
```



Спирали. Цикл **for**

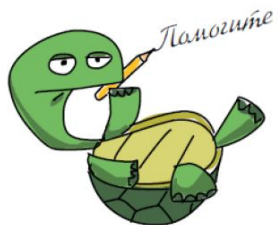
А угол поворота на **121** градус!



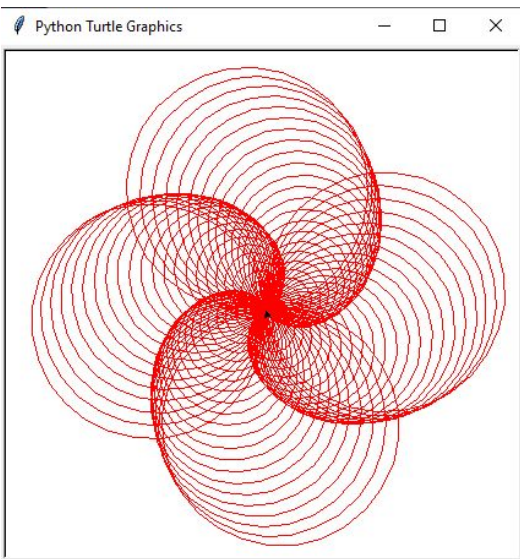
```
import turtle      #подключить модуль turtle
t=turtle.Pen()    # t - обозначение ручки черепашки
for x in range(300): # команда цикла в диапазоне от
                        0 до 99
    t.forward(x)      # идти вперёд x точек на экране
    t.left(121)      # повернуться налево на 121 градус
```



Черепашка закругляется и меняет цвет



Наберите эти команды и у вас получится четыре спиралевидные красные окружности!

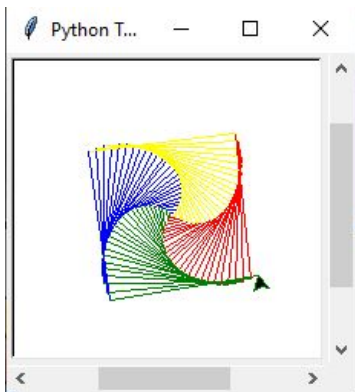


```
import turtle           #подключить модуль turtle
t=turtle.Pen()         # t - обозначение ручки черепашки
t.pencolor('red')      # красный цвет ручки
for x in range(100):    # команда цикла в диапазоне от
                        0 до 99
    t.circle(x)          # нарисовать окружность с радиусом x
    t.left(91)           # повернуться налево на 91 градус
```



Добавим красок

Рисуем четырехцветную спираль

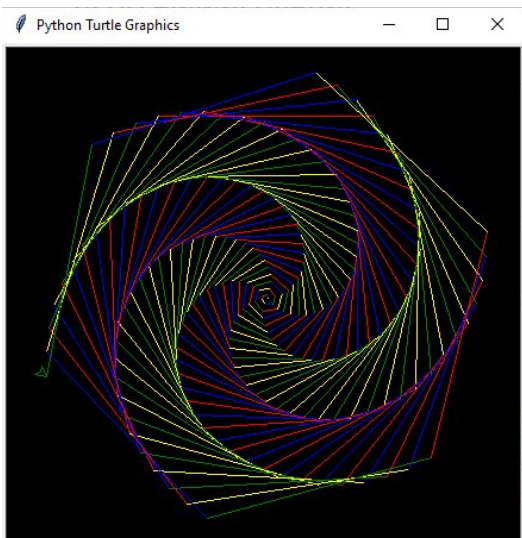


```
import turtle #подключить модуль turtle
t=turtle.Pen() # t - обозначение ручки черепашки
colors = ['red', 'yellow', 'blue', 'green'] # создаём список
цветов
for x in range(100): # команда цикла в диапазоне от
                        0 до 99
    t.pencolor(colors[x%4]) # задаём цвет из списка
    t.forward (x) # идти вперёд x точек на экране
    t.left(91) # повернуться налево на 91 градус
```



Добавим красок

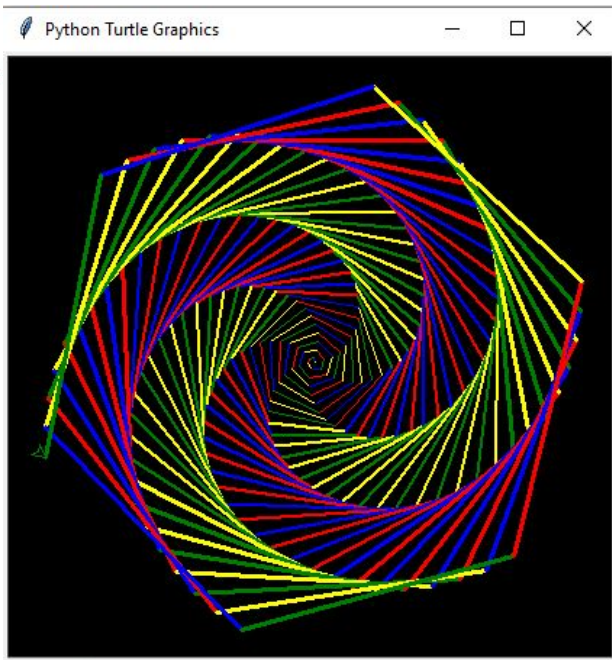
Добавим цвет фона. Изменим угол поворота на **61** градус и диапазон в цикле на **200**!



```
import turtle
t=turtle.Pen()
turtle.bgcolor('black')
colors = ['red', 'yellow', 'blue', 'green']
for x in range(200):
    t.pencolor(colors[x%4])
    t.forward (x)
    t.left(61)
```



Установим толщину пера



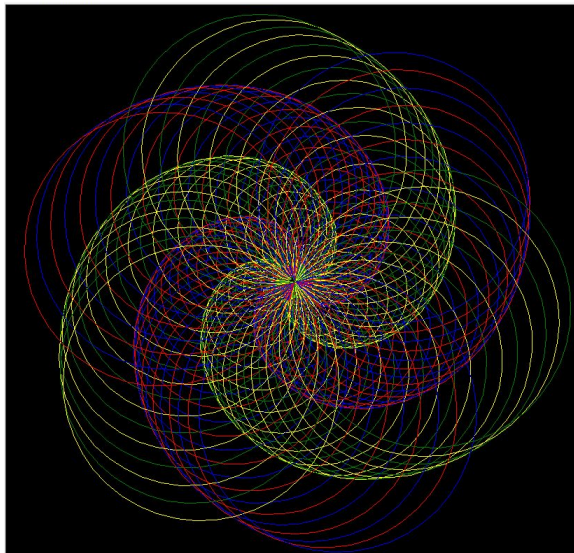
```
import turtle
t=turtle.Pen()
w=1 # толщина пера
turtle.bgcolor('black')
colors = ['red', 'yellow', 'blue', 'green']
for x in range(200):
    t.pencolor(colors[x%4])
    t.forward (x)
    t.left(61)
    t.width(w) # применить толщину пера
    w=w+0.01 # увеличить толщину пера
```



Добавим красок

Поменяем движение вперёд на рисование окружности!

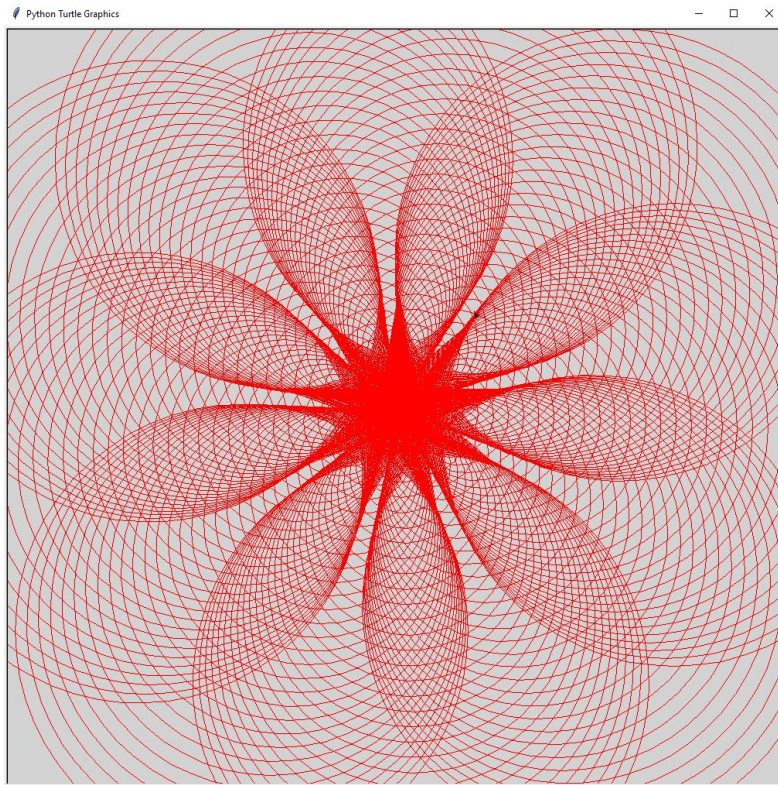
Python Turtle Graphics



```
import turtle
t=turtle.Pen()
turtle.bgcolor('black')
colors = ['red', 'yellow', 'blue', 'green']
for x in range(200):
    t.pencolor(colors[x%4])
    t.circle(x)
    t.left(91)
```



Ромашка



```
import turtle  
t=turtle.Pen()  
t.speed(0)  
turtle.bgcolor('lightgray')  
t.pencolor('red')  
for x in range(300):  
    t.forward (x)  
    t.left(200)  
    t.circle(x)  
t.exitonclick()
```



ПРОГРАММИРОВАНИЕ. 3.9. «Работа с графикой в Python»



File Edit Format Run Options

```
from turtle import *
speed(0)
pencolor('red')
bgcolor('black')
x = 0
up()
rt(45)
fd(90)
rt(135)
down()
while x < 120:
    fd(200)
    rt(61)
    fd(200)
    rt(61)
    fd(200)
    rt(61)
    fd(200)
    rt(61)
    fd(200)
    rt(61)
    fd(200)
    rt(61)
    rt(11.1111)
    x = x+1
exitonclick()
```

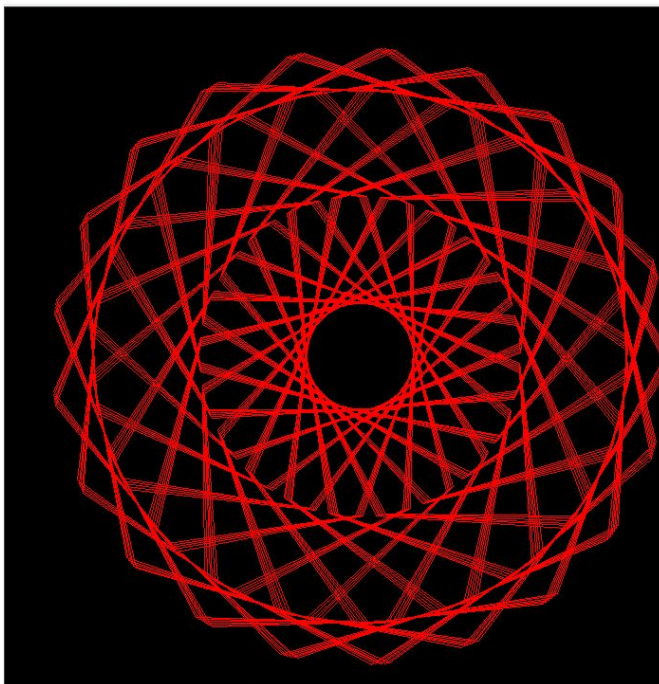


Орнамент. Вложенные циклы

File Edit Format Run Options

```
from turtle import *
speed(0)
pencolor('red')
bgcolor('black')
x = 0
up()
rt(45)
fd(90)
rt(135)
down()
while x < 120:
    k=0
    while k < 6:
        fd(200)
        rt(61)
        k=k+1
        rt(11.1111)
        x = x+1
    exitonclick()
```

Python Turtle Graphics





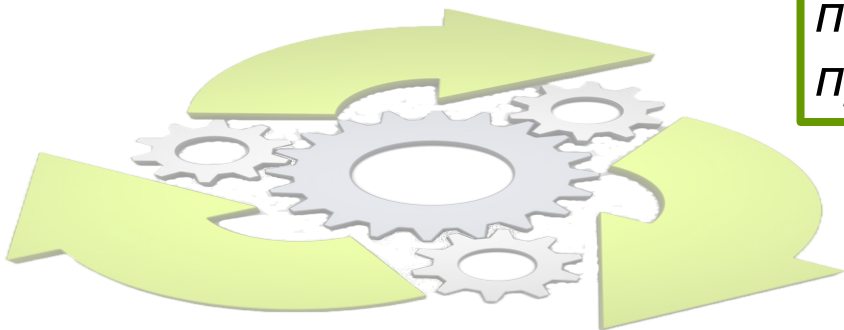
ПРОГРАММИРОВАНИЕ.

3.4. Вложенные условные операции и циклы

Вложенные циклы:

Вложенные циклы - это циклы, которые выполняются в составе других (внешних) циклов.

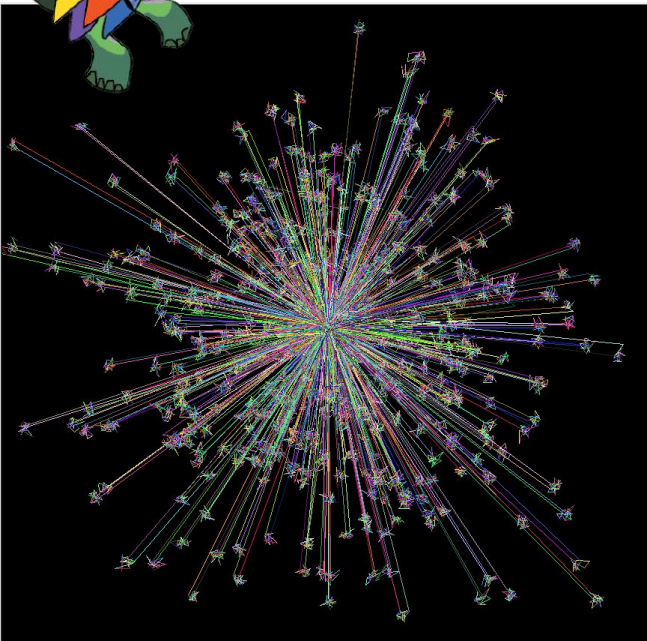
При использовании вложенных циклов важно продумывать количество повторений внешнего цикла и вложенного цикла в нём, так как общее количество повторений вложенного цикла в программе равно их произведению.





ПРОГРАММИРОВАНИЕ. 3.9. «Работа с графикой в Python»

Пучок прямых лучей



File Edit Format Run Options Window Help

```
import turtle
import random
turtle.tracer(0, 0)
mi = turtle.Screen()
mi.colormode(255)
turtle.bgcolor("black")
t = turtle.Turtle()
t.speed(0)
t.goto(0,0)
t.pensize(0)
t.ht()
for i in range(500):
    t.color(random.randrange(256), random.randrange(256), random.randrange(256))
    t.goto(round(random.gauss(0,150), 0), round(random.gauss(0,150), 0))
    x = t.xcor()
    y = t.ycor()
    for j in range(25):
        z = round(random.gauss(0,5), 0)
        a = round(random.gauss(0,5), 0)
        t.color(random.randrange(256), random.randrange(256), random.randrange(256))
        t.pensize(0)
        t.goto(x + z, y + a)
    t.goto(z, a)
turtle.update()
mi.exitonclick()
```



Бесконечное движение черепашек по кругу

File Edit Format Run Options Window Help

```
import turtle
```

```
screen = turtle.Screen()
```

```
screen.tracer(2)
```

```
screen.bgcolor('navy')
```

```
turtles = []
```

```
colors = ['orange', 'red', 'blue', 'grey', 'gold', 'brown', 'black', 'pink']
```

```
for i in range(8):
```

```
    turtles.append(turtle.Turtle('turtle'))
```

```
    turtles[i].color(colors[i])
```

```
for j in range(8):
```

```
    turtles[j].right(j * 45)
```

```
i = -1
```

```
while True:
```

```
    i = i + 1
```

```
    i1 = i % 8
```

```
    turtles[i1].fd(10)
```

```
    turtles[i1].left(10)
```

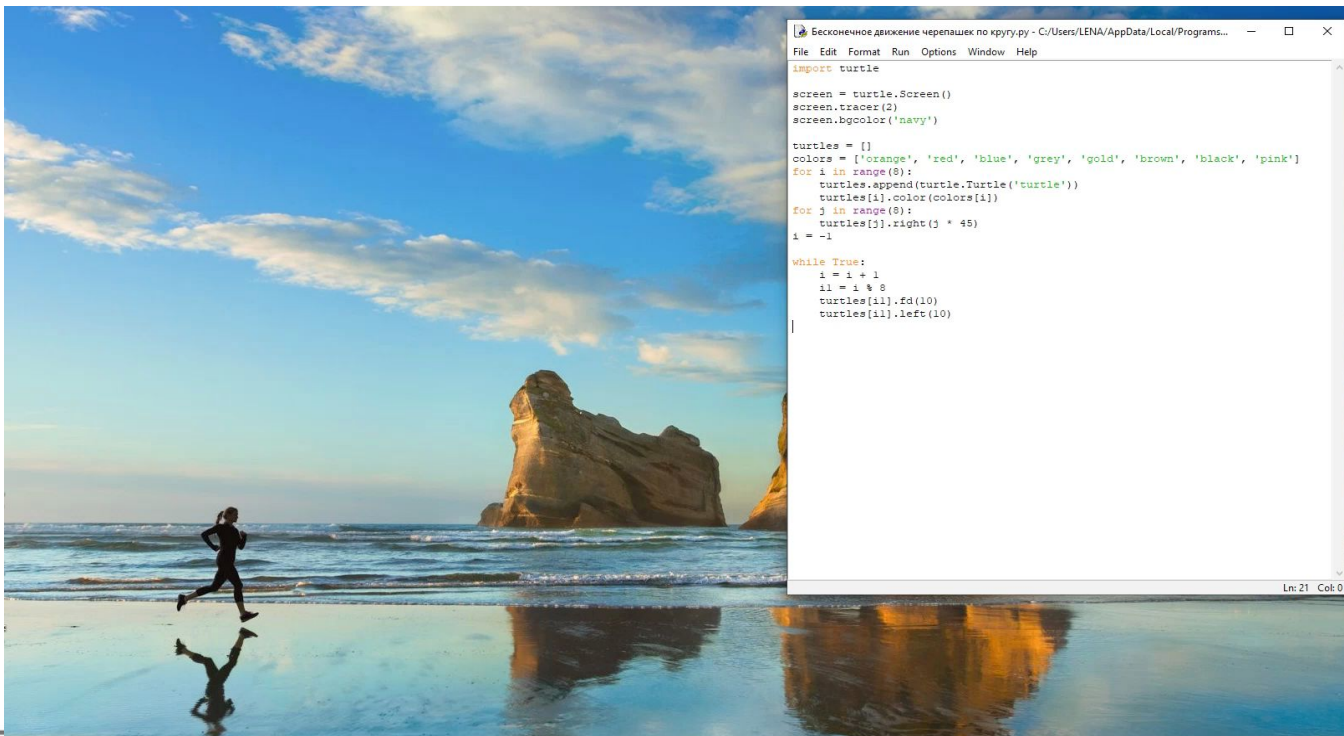


ПРОГРАММИРОВАНИЕ.

3.9. «Работа с графикой в Python»

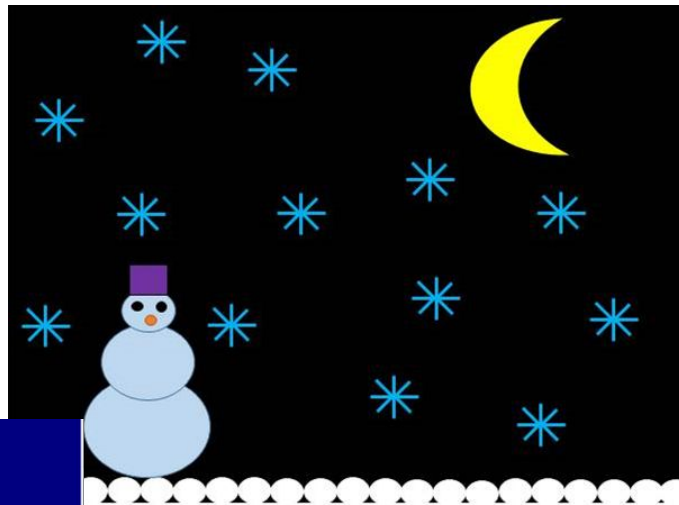
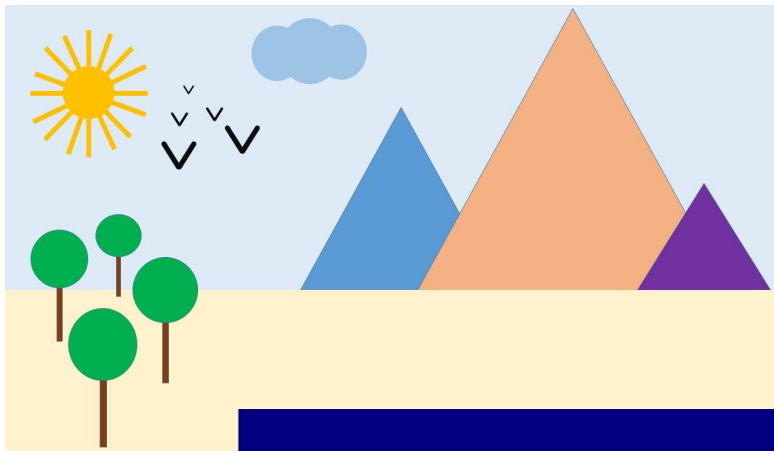


Бесконечное движение черепашек по кругу



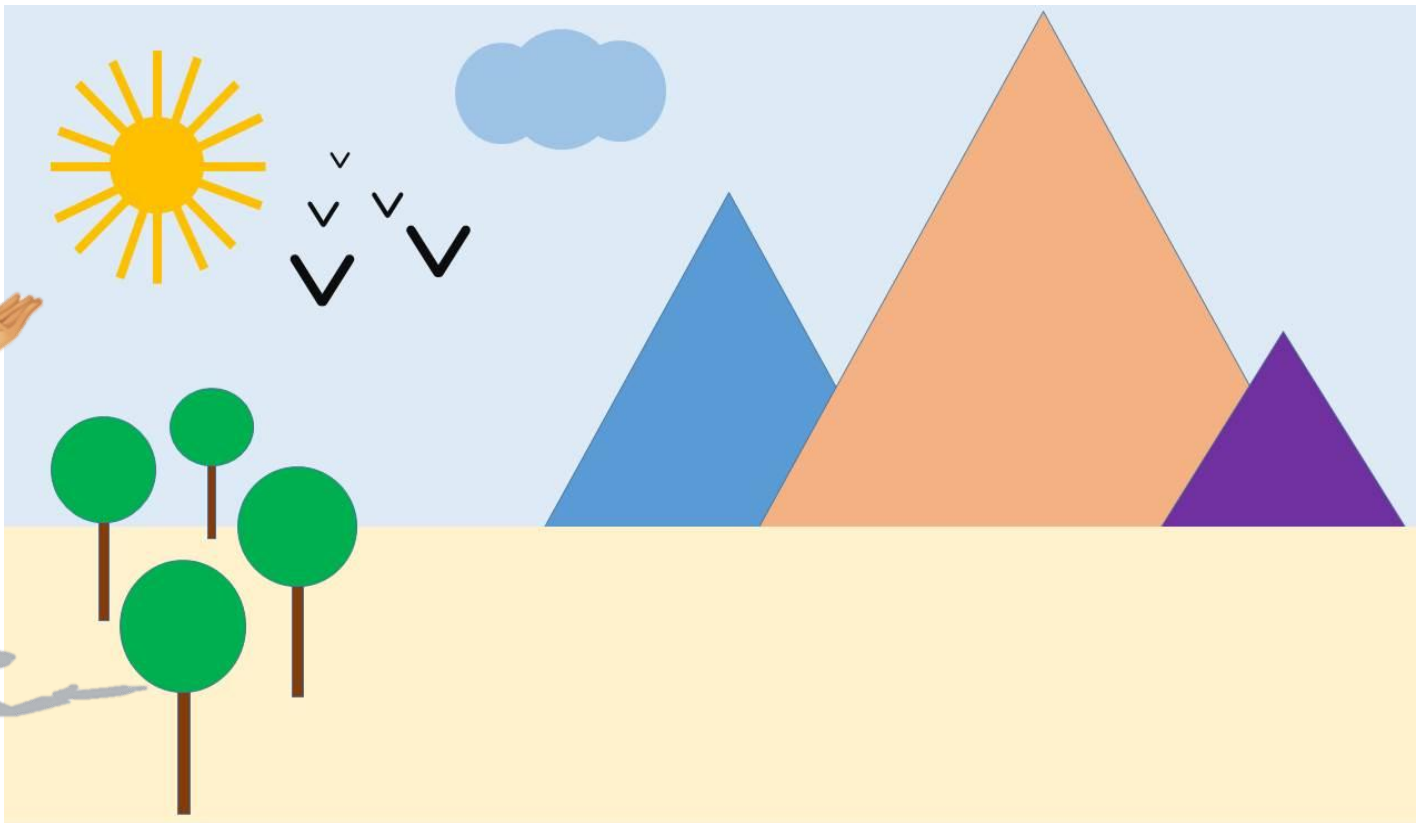


«Питонические» пейзажи



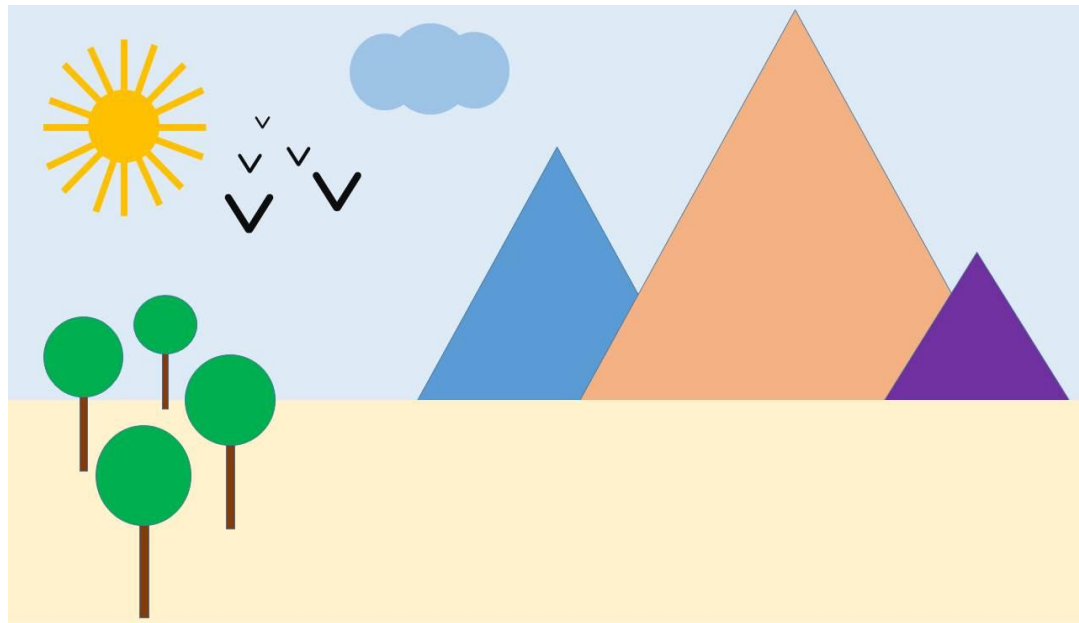


Разберём рисунок





Разберём рисунок



Пример кода:

```
pyramid(300, 'blue')  
fd(200)  
pyramid(400, 'orange')  
fd(350)  
pyramid(100, 'purple')  
bird(-200,200,30)  
bird(-200,250,20)  
bird(-180,300,10)
```



Создание собственной команды - функции

Для создания собственной
команды используется
специальное слово

def

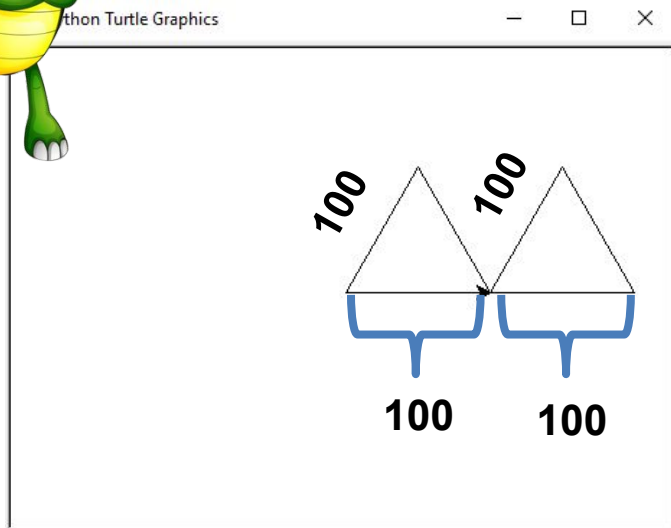


def сокращение от **define** - **определить**



Создание собственной команды - функции

Пример:



```
from turtle import*
```

```
def triangle():
```

Имя функции

```
    fd(100)
```

```
    lt(120)
```

```
    fd(100)
```

```
    lt(120)
```

```
    fd(100)
```

```
    lt(120)
```

Тело функции

```
triangle()
```

```
fd(100)
```

```
triangle()
```

```
exitonclick()
```

Вызов функции

**Шаг 1**

*Рисуем
треугольник*

```
from turtle import*  
fd(100)  
lt(120)  
fd(100)  
lt(120)  
fd(100)  
lt(120)  
exitonclick()
```

Шаг 2

*На основе полученного
кода создаём функцию tr()*

```
from turtle import*  
def tr():  
    fd(100)  
    lt(120)  
    fd(100)  
    lt(120)  
    fd(100)  
    lt(120)  
    exitonclick()
```

Шаг 3

Используем команду вызова

```
from turtle import*  
def tr():  
    fd(100)  
    lt(120)  
    fd(100)  
    lt(120)  
    fd(100)  
    lt(120)  
  
tr()  
fd(100)  
tr()  
exitonclick()
```



Функции с параметром

```
from turtle import *
```

```
def tr(a, cvet):
```

```
    fillcolor(cvet)
```

```
    begin_fill()
```

```
    fd(a)
```

```
    lt(120)
```

```
    fd(a)
```

```
    lt(120)
```

```
    fd(a)
```

```
    lt(120)
```

```
tr(100, green)
```

```
fd(100)
```

```
tr(200, 'grey')
```

```
exitonclick()
```

Параметры функции

Будет использоваться вместо **a**

Будет использоваться вместо **cvet**

РЕЗУЛЬТАТ





Функции с параметром

```
from turtle import*  
def tr(a, cvet):  
    fillcolor(cvet)  
    begin_fill()  
    fd(a)  
    lt(120)  
    fd(a)  
    lt(120)  
    fd(a)  
    lt(120)  
tr(100,'green')  
fd(100)  
tr(200,'grey')  
exitonclick()
```

Команда **forward** тоже является функцией, для того чтобы сказать черепашке сколько шагов ей идти, мы указываем количество шагов в скобках команды.





```
from turtle import*
```

```
def tr(x, y):
```

```
    up()
```

```
    goto(x,y)
```

```
    down()
```

```
    fd(100)
```

```
    lt(120)
```

```
    fd(100)
```

```
    lt(120)
```

```
    fd(100)
```

```
    lt(120)
```

```
tr(0,0)
```

```
fd(100)
```

```
tr(50,200)
```

```
exitonclick()
```

Создание функции



```
from turtle import*
```

```
def tr(x, y):
```

```
    up()
```

```
    goto(x,y)
```

```
    down()
```

```
    stor=0
```

```
while stor<3:
```

```
    fd(100)
```

```
    lt(120)
```

```
    stor=stor+1
```

```
tr(0,0)
```

```
fd(100)
```

```
tr(50,200)
```

```
exitonclick()
```





ПРОГРАММИРОВАНИЕ. 3.9. «Работа с графикой в Python»



Создание функции

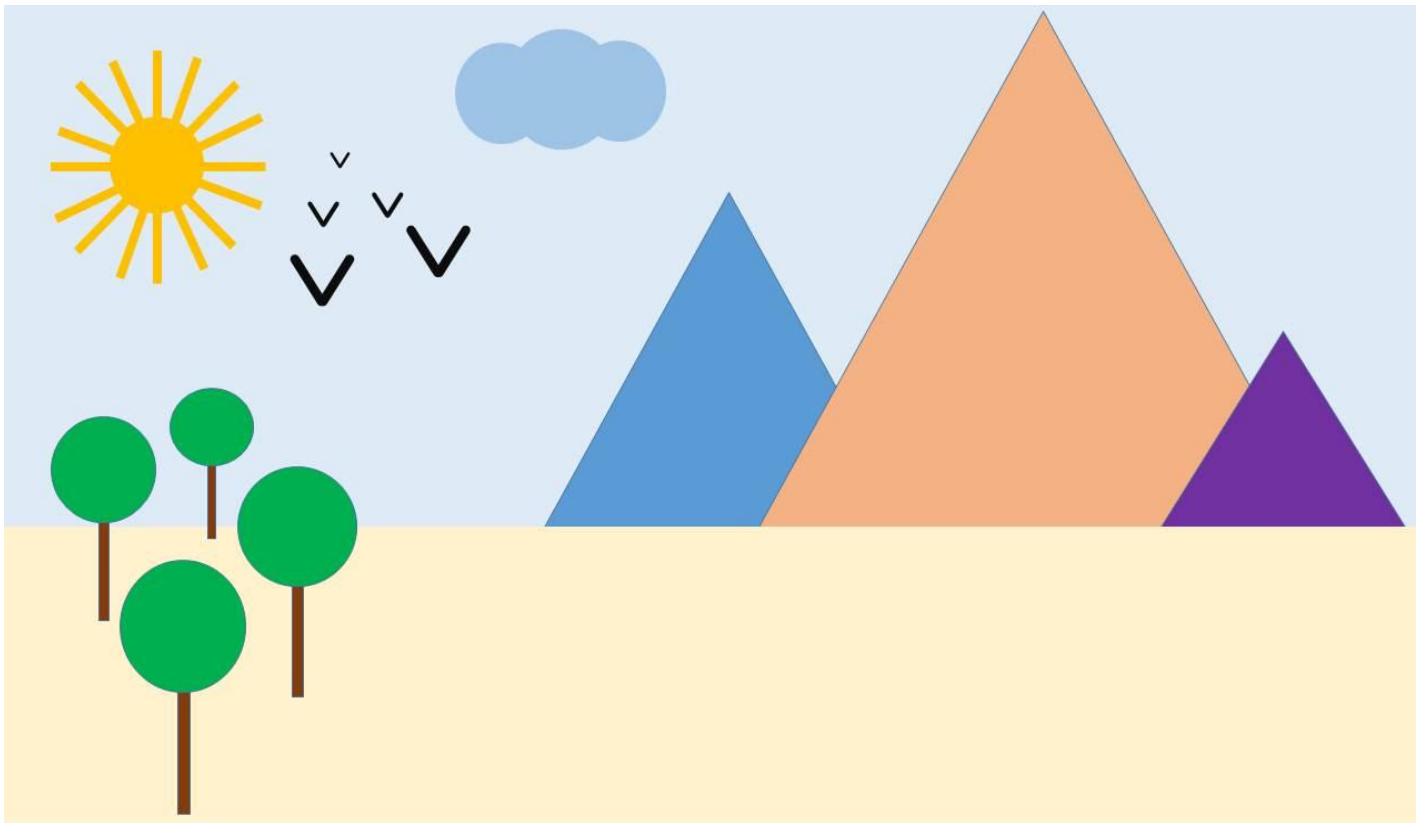


```
knjb.py - C:/Users/LENA/Downloads/knjb.py (3.8.5)
File Edit Format Run Options Window Help
from turtle import *
def tr(x, y):
    up()
    goto(x,y)
    down()
    stor=0
    while stor<3:
        fd(100)
        lt(120)
        stor=stor+1
tr(0,0)
fd(100)
tr(50,200)
exitonclick()
```

Ln: 6 Col: 4



Д.3.: Создание «питонического» пейзажа





Пример. Рисуем персонаж

начало

```
import turtle
```

```
# Основные цвета для персонажа
```

```
BODY_COLOR = 'red'
```

```
GLASS_COLOR = 'skyblue'
```

```
# Главный объект
```

```
t = turtle.Turtle()
```

```
# Метод для рисования тела
```

```
def body():
```

```
    t.pensize(30) # Размер кисти
```

```
    t.fillcolor(BODY_COLOR) # Цвет заполнения
```

```
    t.begin_fill()
```



Пример4. Рисуем персонаж

персонаж

Сторона справа

```
t.right(90)  
t.forward(50)  
t.right(180)  
t.circle(40, -180)  
t.right(180)  
t.forward(200)
```



Голова

```
t.right(180)  
t.circle(100, -180)
```



Пример4. Рисуем персонаж

персонаж

Сторона слева

```
t.backward(20)  
t.left(15)  
t.circle(500, -20)  
t.backward(20)
```

```
t.circle(40, -180)  
t.left(7)  
t.backward(50)
```



```
t.up()  
t.left(90)  
t.forward(10)  
t.right(90)  
t.down()
```

```
t.right(240)  
t.circle(50, -70)
```

```
t.end_fill()
```



Пример4. Рисуем персонаж

персонаж

Рисуем очки

def glass():

Передвигаем черепашку

t.up()

t.right(230)

t.forward(100)

t.left(90)

t.forward(20)

t.right(90)

t.down()



Устанавливаем цвет

t.fillcolor(GLASS_COLOR)

t.begin_fill()

t.right(150)

t.circle(90, -55)

t.right(180)

t.forward(1)

t.right(180)

t.circle(10, -65)

t.right(180)

t.forward(110)

t.right(180)



Пример4. Рисуем персонаж

персонаж

```
t.circle(50, -190)
t.right(170)
t.forward(80)
t.right(180)
t.circle(45, -30)
t.end_fill()
```

Рисуем рюкзак

```
def backpack():
    t.up()
    t.right(60)
    t.forward(100)
```



```
t.right(90)
t.forward(75)
t.fillcolor(GLASS_COLOR)
t.begin_fill()
t.down()
t.forward(30)
t.right(255)
t.circle(300, -30)
t.right(260)
t.forward(30)
t.end_fill()
```



Пример4. Рисуем персонаж

Вызываем все необходимые методы

body()

glass()

backpack()

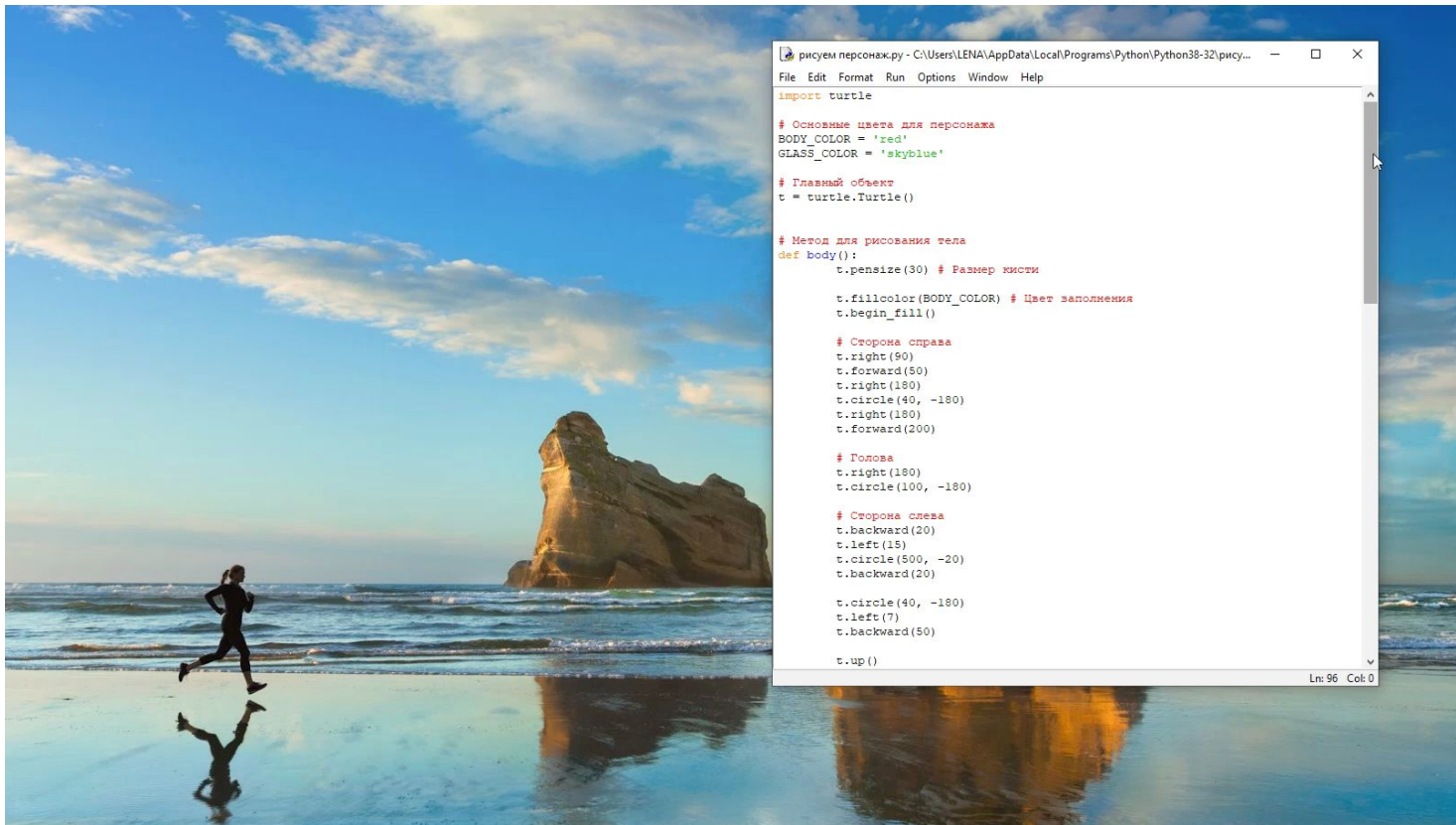
turtle.done()



ПРОГРАММИРОВАНИЕ.

3.9. «Работа с графикой в Python»

Пример4. Рисуем персонаж





Пример4. Рисуем персонаж





1

**Циклы в «Черепашьей графике».
Решение практических задач**

2

**Создаём свои команды. Функции в
«Черепашьей графике»**

3

Решение практических задач



Урок разработала
Клепачёва Е.А.,
учитель информатики УК АФМШЛ №61