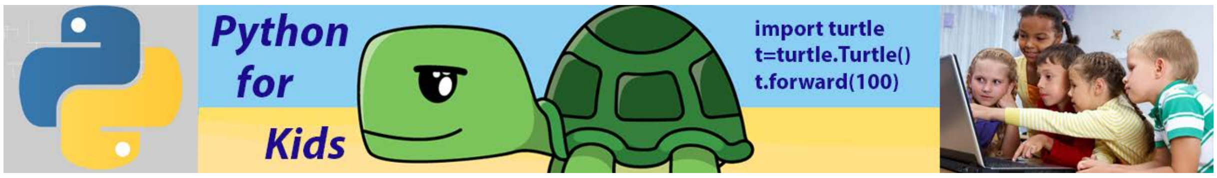
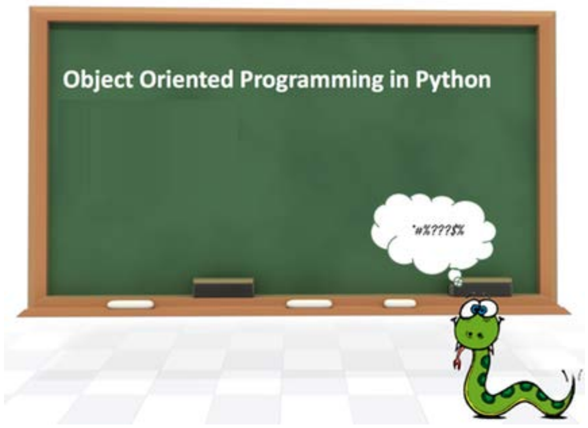


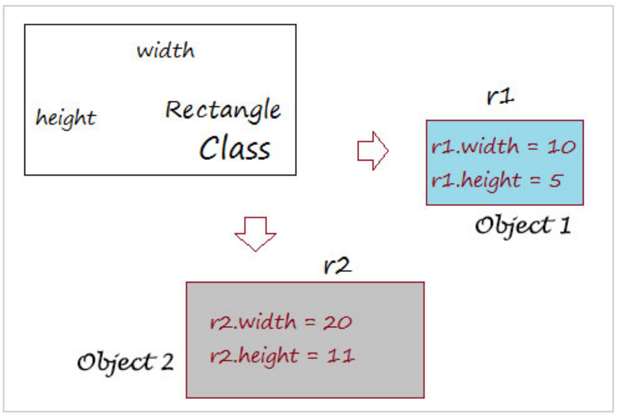
Lesson 18: Classes and Objects

Object-oriented programming
Python is an object-oriented programming language, which means that it provides features that support object oriented programming (OOP). In object-oriented programming the focus is on the creation of objects which contain both data and functionality together. We've already seen classes Turtle and worked with them in lesson #12.



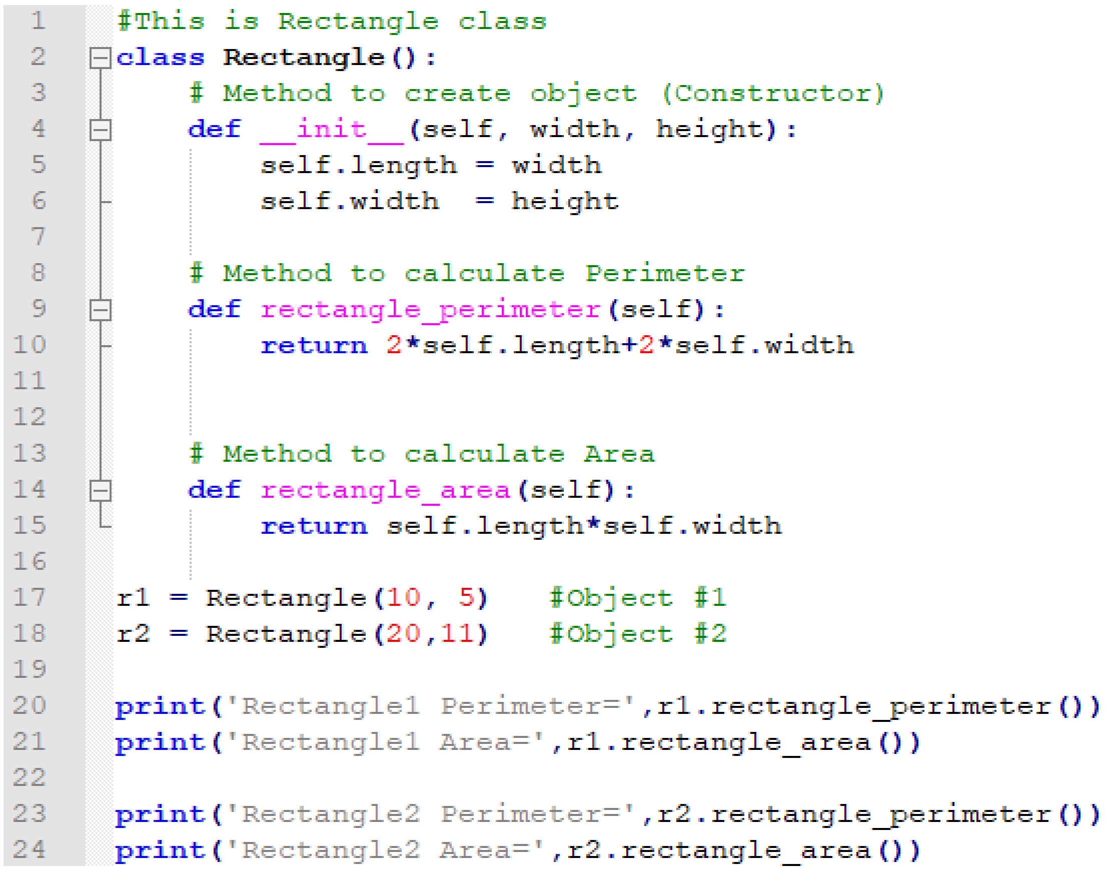
Below it is shown code that calculates rectangle perimeter and rectangle area using class method (with Python).
Figure below shows visually class Rectangle and two objects: object 1 and object 2 that are components of created class. Our goal: to calculate perimeter and area of two objects r1 and r2. First object is a rectangle named as r1 with a width=10 and a height=5, second object r2 has a width=20 and a height=11.

Fig. 1



Code for calculation is shown below

1. Example #1



```
1   #This is Rectangle class
2   class Rectangle():
3       # Method to create object (Constructor)
4       def __init__(self, width, height):
5           self.length = width
6           self.width  = height
7
8       # Method to calculate Perimeter
9       def rectangle_perimeter(self):
10          return 2*self.length+2*self.width
11
12
13      # Method to calculate Area
14      def rectangle_area(self):
15          return self.length*self.width
16
17  r1 = Rectangle(10, 5)   #Object #1
18  r2 = Rectangle(20,11)   #Object #2
19
20  print('Rectangle1 Perimeter=',r1.rectangle_perimeter())
21  print('Rectangle1 Area=',r1.rectangle_area())
22
23  print('Rectangle2 Perimeter=',r2.rectangle_perimeter())
24  print('Rectangle2 Area=',r2.rectangle_area())
```

Let's explain basic structure of the code, that uses classes and objects concept. You need to start with the `class` keyword to indicate that you are creating a class, then you add the name of the class (in our case **Rectangle**) and colon (:) [line #2]. Remember: the class is just for defining the **Rectangle**, not actually creating *instances* of individual rectangles with specific width and height; we'll get to that shortly.

In the class body [lines #4-15], you need to declare attributes(parameters), methods (of calculation in our case), and constructors.

Attribute:

The attribute is a member of the class. Our rectangle code has two attributes (parameters): width and height. All classes create objects, and all objects contain characteristics called attributes (In our case **class**-Rectangle and **objects**-r1, r2, which is shown in Fig 1)

Method:

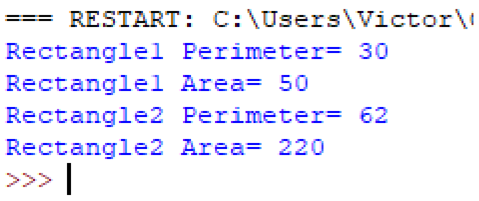
The method of class is similar to a normal function but it is a function of class, in order to use it, you need to call through object. The first parameter of the method is always **self** (a keyword that refers to the class itself).

Constructor:

Constructor is used to create an object. Constructor assigns values of the parameter to the properties of the object that will be created. You can only define a constructor in class. The task of constructors is to initialize (assign values) to the data members of the class when an object of class is created. In Python the __init__() method is called the constructor and is always called when an object is created. The first parameter of the constructor is always **self** (a keyword refers to the class itself). If the class is not defined by the constructor, **Python** assumes that it inherit constructor of parent class.

NOTE: You will never have to call the `__init__()` method; it gets called automatically when you create a new 'Rectangle' instance.

Result of the code, which calculates rectangle parameters is shown below



```
=== RESTART: C:\Users\Victor\
Rectangle1 Perimeter= 30
Rectangle1 Area= 50
Rectangle2 Perimeter= 62
Rectangle2 Area= 220
>>> 
```

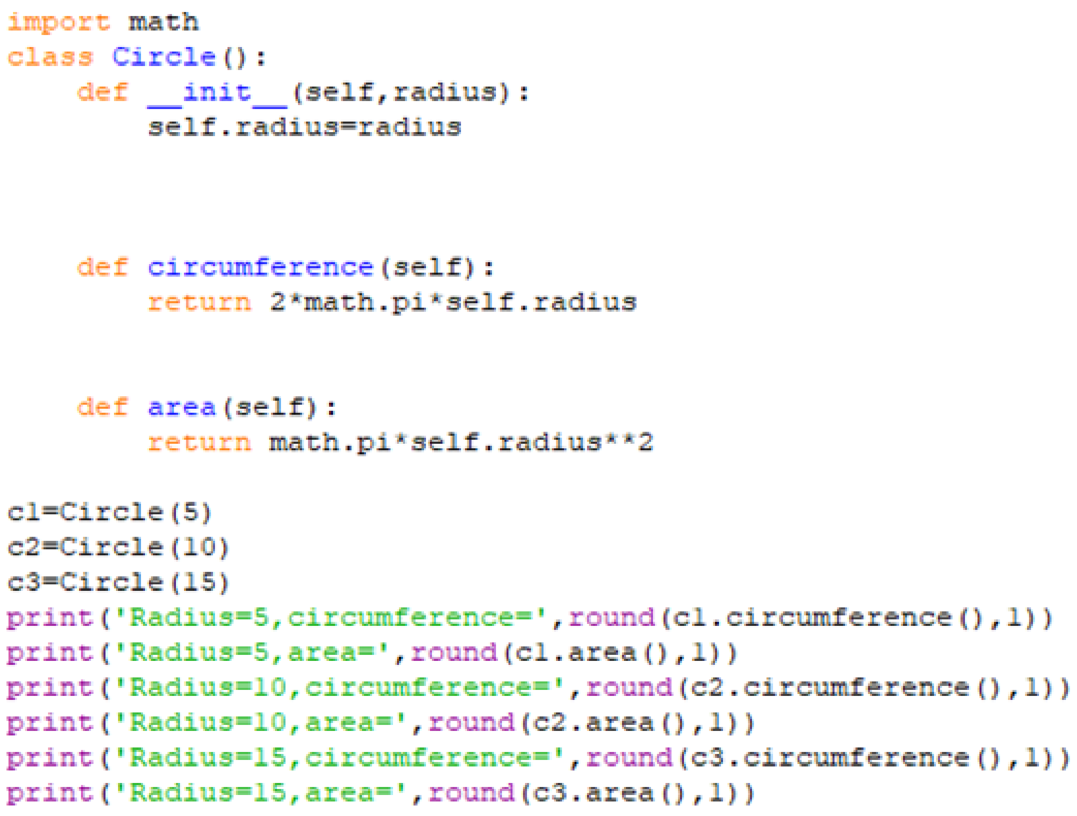
Similar we can add to our code an object r3 with the following parameters width=52, height =7. Code and result of calculation looks like:

2. Example #2

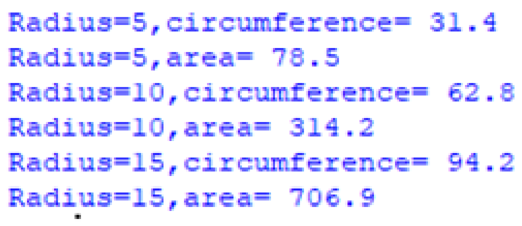
```
#This is Rectangle class
class Rectangle():
    # Method to create object (Constructor)
    def __init__(self, width, height):
        self.length = width
        self.width  = height

    # Method to calculate Perimeter
    def rectangle_perimeter(self):
        return 2*self.length+2*self.width


    # Method to calculate Area
    def rectangle_area(self):
        return self.length*self.width

r1 = Rectangle(10, 5)    #Object #1
r2 = Rectangle(20,11)    #Object #2
r3 = Rectangle(52,7)    #Object #2

print('Rectangle1 Perimeter=',r1.rectangle_perimeter())
print('Rectangle1 Area=',r1.rectangle_area())

print('Rectangle2 Perimeter=',r2.rectangle_perimeter())
print('Rectangle2 Area=',r2.rectangle_area())

print('Rectangle3 Perimeter=',r3.rectangle_perimeter())
print('Rectangle3 Area=',r3.rectangle_area())
```

```
Rectangle1 Perimeter= 30
Rectangle1 Area= 50
Rectangle2 Perimeter= 62
Rectangle2 Area= 220
Rectangle3 Perimeter= 118
Rectangle3 Area= 364
...
```

3. **Example#3** (Circle Calculation with classes concept)

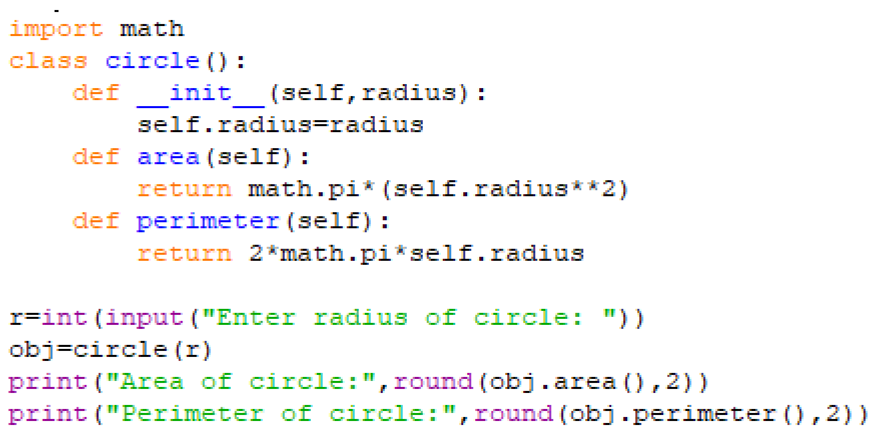


```python
import math
class Circle():
    def __init__(self,radius):
        self.radius=radius



    def circumference(self):
        return 2*math.pi*self.radius


    def area(self):
        return math.pi*self.radius**2

c1=Circle(5)
c2=Circle(10)
c3=Circle(15)
print('Radius=5,circumference=',round(c1.circumference(),1))
print('Radius=5,area=',round(c1.area(),1))
print('Radius=10,circumference=',round(c2.circumference(),1))
print('Radius=10,area=',round(c2.area(),1))
print('Radius=15,circumference=',round(c3.circumference(),1))
print('Radius=15,area=',round(c3.area(),1))
```



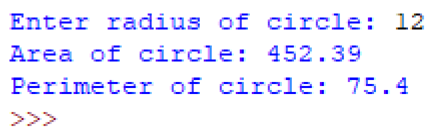
```
Radius=5,circumference= 31.4
Radius=5,area= 78.5
Radius=10,circumference= 62.8
Radius=10,area= 314.2
Radius=15,circumference= 94.2
Radius=15,area= 706.9
```

4. **Example #4**(Circle Calculation with classes concept and input data)



```
import math
class circle():
    def __init__(self,radius):
        self.radius=radius
    def area(self):
        return math.pi*(self.radius**2)
    def perimeter(self):
        return 2*math.pi*self.radius

r=int(input("Enter radius of circle: "))
obj=circle(r)
print("Area of circle:",round(obj.area(),2))
print("Perimeter of circle:",round(obj.perimeter(),2))
```

Output:



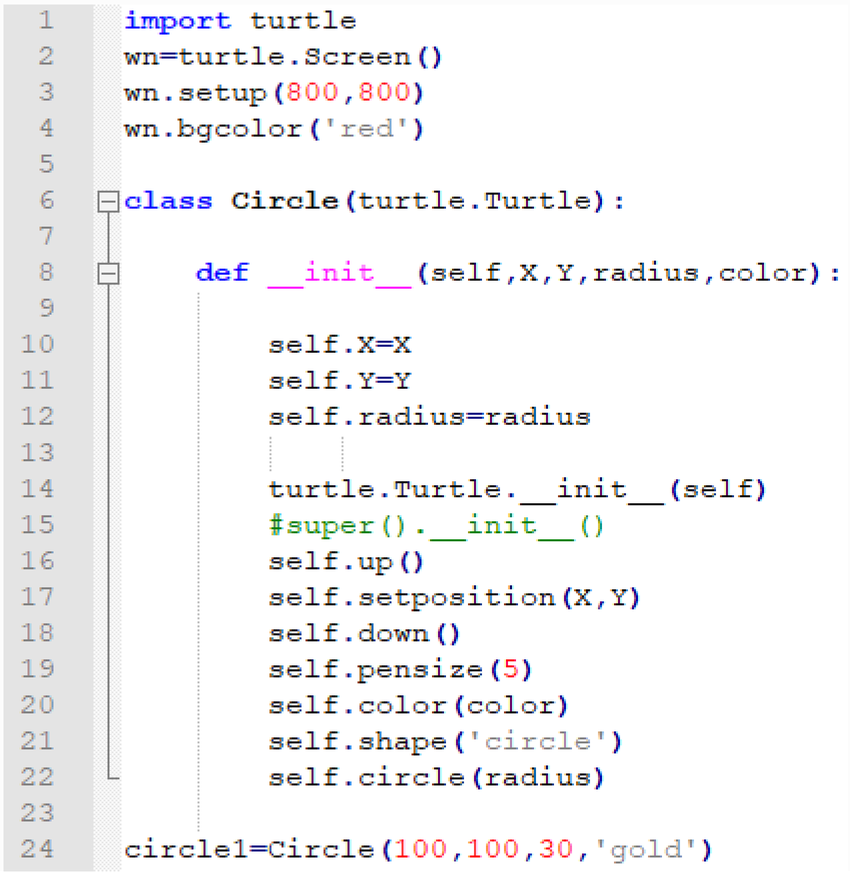
```
Enter radius of circle: 12
Area of circle: 452.39
Perimeter of circle: 75.4
>>>
```

Program Explanation

1. User must enter the value of radius.
   2. A class called circle is created and the __init__() method is used to initialize values of that class.
   3. A method called as area returns math.pi*(self.radius**2) which is the area of the class.
   3. Another method called perimeter returns 2*math.pi*self.radius which is the perimeter of the class.
   5. An object for the class is created.
   6. Using the object, the methods area() and perimeter() are called.
   7. The area and perimeter of the circle is printed.

Below it is shown an example that allows to combine turtle GUI (graphic user interface) with class concept. We create code to draw circle with some certain parameters

5. **Example #5** (combine turtle GUI and class method)



```
1   import turtle
2   wn=turtle.Screen()
3   wn.setup(800,800)
4   wn.bgcolor('red')
5
6   class Circle(turtle.Turtle):
7
8       def __init__(self,X,Y,radius,color):
9
10          self.X=X
11          self.Y=Y
12          self.radius=radius
13
14          turtle.Turtle.__init__(self)
15          #super().__init__()
16          self.up()
17          self.setposition(X,Y)
18          self.down()
19          self.pensize(5)
20          self.color(color)
21          self.shape('circle')
22          self.circle(radius)
23
24  circle1=Circle(100,100,30,'gold')
```

Program Explanation

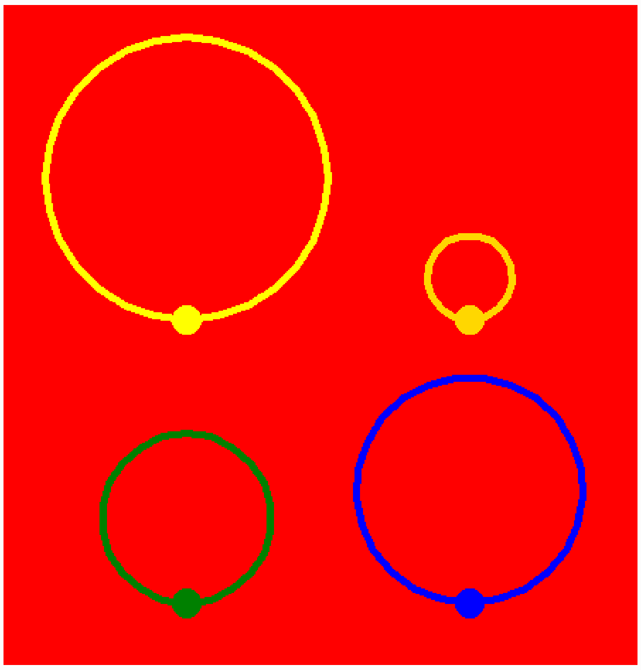
1. First three lines are common for turtle class which comes with Python soft Line #4 creates Circle subclass with turtle.
2. X and Y position of the circle on the plane are the parameters of the class Circle. Also, parameters of the Circle class are radius and color.
3. Line #14 allows to communicate turtle commands with class Circle declared with line #6.
4. Lines #16-22 determine main turtle commands, which are known from the previous lessons.
5. Line #24 specifies object circle with certain parameters.

We can add any numbers of circle objects to the end of this code, for example,

6. Example #6

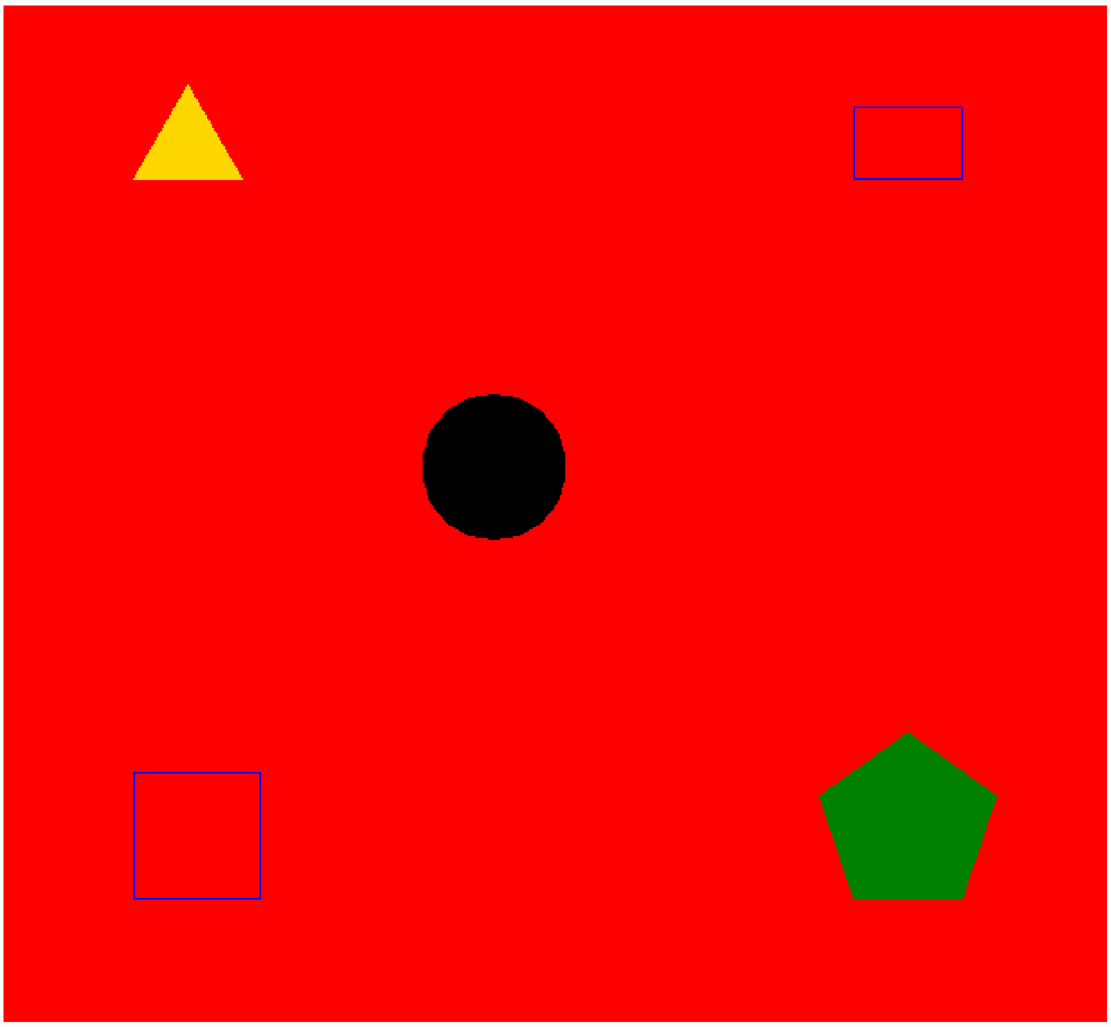
```
import turtle
wn=turtle.Screen()
wn.setup(800,800)
wn.bgcolor('red')

class Circle(turtle.Turtle):

    def __init__(self,X,Y,radius,color):

        self.X=X
        self.Y=Y
        self.radius=radius

        turtle.Turtle.__init__(self)
        #super().__init__()
        self.up()
        self.setposition(X,Y)
        self.down()
        self.pensize(5)
        self.color(color)
        self.shape('circle')
        self.circle(radius)

circle1=Circle(100,100,30,'gold')
circle1=Circle(100,-100,80,'blue')
circle1=Circle(-100,-100,60,'green')
circle1=Circle(-100,100,100,'yellow')
```

Output Result:



7. **Example #7** (Drawing: rectangle, circle, pentagon, and square using class concept)

Output Result

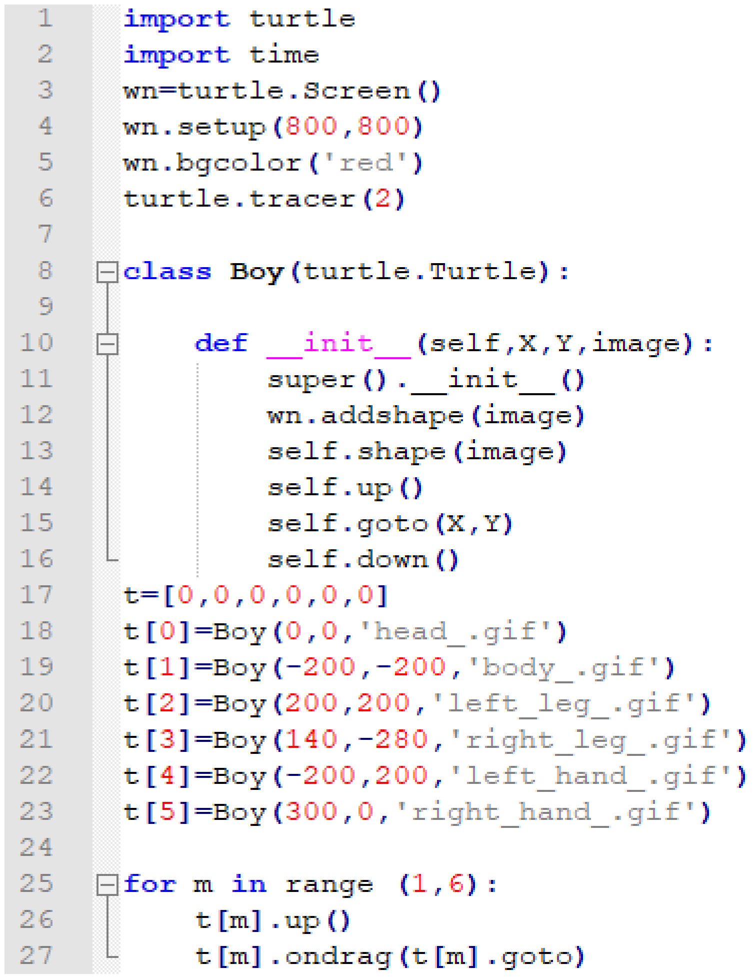


Code for example #7

```python
import turtle
wn=turtle.Screen()
wn.setup(800,800)
wn.bgcolor('red')
wn.tracer(5)
class Polygons(turtle.Turtle):

    def __init__(self):
        super().__init__()
        self.hideturtle()

    def draw_circle(self,R,color):
        self.down()
        self.shape('circle')
        self.color(color)
        self.begin_fill()
        self.circle(R)
        self.end_fill()
        self.hideturtle()

    def draw_rectangle(self,X,Y,width,length,color):
        self.shape('turtle')
        self.up()
        self.goto(X,Y)
        self.down()
        self.color(color)
        for i in range(2):
            self.fd(width)
            self.left(90)
            self.fd(length)
            self.left(90)
        self.hideturtle()

    def draw_triangle(self,X,Y,side,color):
        self.shape('turtle')
        self.up()
        self.goto(X,Y)
        self.down()
        self.color(color)
        self.begin_fill()
        for i in range(3):
            self.fd(side)
            self.left(120)
        self.end_fill()
        self.hideturtle()

    def draw_pentagon(self,X,Y,side,color):
        self.shape('turtle')
        self.up()
        self.goto(X,Y)
        self.down()
        self.color(color)
        self.begin_fill()
        for i in range(5):
            self.fd(side)
            self.left(72)
        self.end_fill()
        self.hideturtle()

tcircle=Polygons()
tcircle.draw_circle(40,'black')
trectangle=Polygons()
trectangle.draw_rectangle(200,200,60,40,'blue')
trectangle.draw_rectangle(-200,-200,70,70,'blue')
ttriangle=Polygons()
trectangle.draw_triangle(-200,200,60,'gold')
trectangle.draw_pentagon(200,-200,60,'green')
```

8. **Example #8** (Using Class function together with ondrag function)

Code:



```
1  import turtle
2  import time
3  wn=turtle.Screen()
4  wn.setup(800,800)
5  wn.bgcolor('red')
6  turtle.tracer(2)
7
8  class Boy(turtle.Turtle):
9
10     def __init__(self,X,Y,image):
11         super().__init__()
12         wn.addshape(image)
13         self.shape(image)
14         self.up()
15         self.goto(X,Y)
16         self.down()
17 t=[0,0,0,0,0,0]
18 t[0]=Boy(0,0,'head_.gif')
19 t[1]=Boy(-200,-200,'body_.gif')
20 t[2]=Boy(200,200,'left_leg_.gif')
21 t[3]=Boy(140,-280,'right_leg_.gif')
22 t[4]=Boy(-200,200,'left_hand_.gif')
23 t[5]=Boy(300,0,'right_hand_.gif')
24
25 for m in range (1,6):
26     t[m].up()
27     t[m].ondrag(t[m].goto)
```

Presented code contains class Boy (lines #8-16). These lines determine boy body parts, that are uploaded to the program with images (.gif file extension). Lines #18-23 are the turtle objects (head, body, left leg, right leg, left hand and right hand). Lines 25-27 include ondrag function for all boy body parts. So, we can create boy from five body parts.

Result: