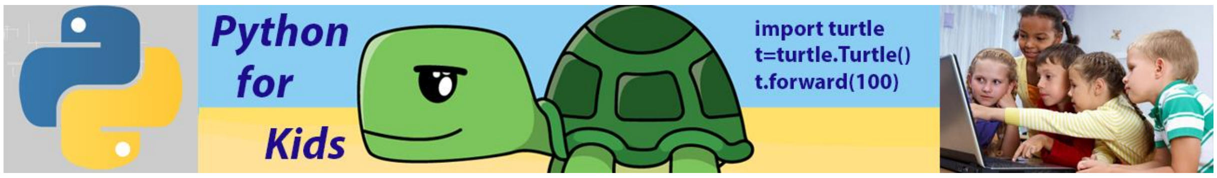


Lesson 11: Functions

Programmers like to make writing code easier. One of the most common shortcuts is to give a name to a block of code that does a repeatable job. Then, instead of having to type out the whole block each time, you need it, you can simply type its name. These named blocks of code are called functions.

Functions

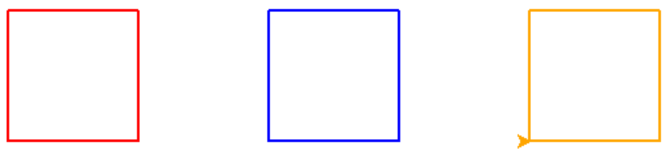
Python language and each library Python module (for example, Turtle module, Random module Math module and so on) contains lots of useful build in functions for performing certain tasks. Before using a module and all library module functions you have to tell the computer to import it so it can be used by your program. After you import certain module all module build in functions (commands) are getting available for you. However very often you need to create your own function to make your code much shorter and elegant. Let's consider that you need to draw several squares in a turtle graphics program.

1. **Example #1** (A few squares)

Create a code:

a.

```
import turtle

t = turtle.Turtle()
t.color('red')
t.pensize(2)

t.up()
t.goto(-200,0)
t.down()

for i in range (4):
  t.fd(100)
  t.left(90)

t.up()
t.goto(0,0)
t.color('blue')
t.down()

for i in range (4):
  t.fd(100)
  t.left(90)

t.up()
t.goto(200,0)
t.color('orange')
t.down()

for i in range (4):
  t.fd(100)
  t.left(90)
```



This program contains three pieces of reusable code:

```
for i in range (4):
  t.fd(100)
  t.left(90)
```

Of course, you copy and paste this code to each place in your program where you want to draw a square. In this example reusable code include only three line, therefore it is not a problem to copy and paste it. However, if this part has 10, 20 or more lines it could be a problem: program becomes too long. In such situations there is exist a solution: to define your own function and call it when it necessary. So, function is a chunk of code that tell Python to do something again and again. A function can be used to avoid entering the same lines of code more than once. The definition of a function will always have the keyword *def* and the function's name at the beginning of the code. In our case it could be

A colon marks the end of the function's name and the start of the code it contains

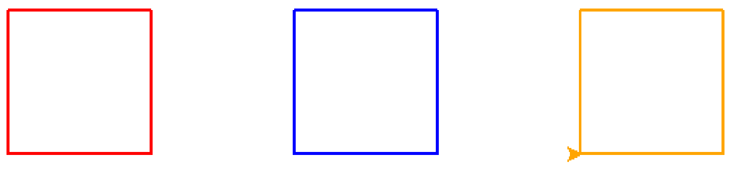
```
def square():

  for i in range (4):
    t.fd(100)
    t.left(90)
```

This is the code within the function

After defining a function, we have to call it in our program using the function's name followed by parenthesis: square()
Here's the code with a function:

b.

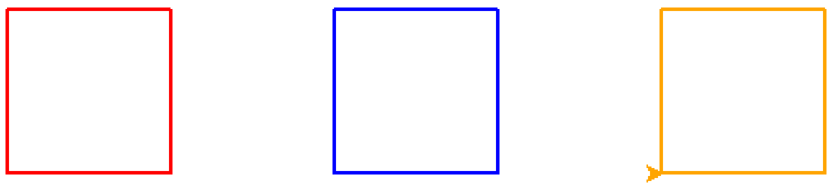
```
import turtle

t = turtle.Turtle()
t.color('red')
t.pensize(2)

def square():
  for i in range (4):
    t.fd(100)
    t.left(90)

t.up()
t.goto(-200,0)
t.down()
square()

t.up()
t.goto(0,0)
t.color('blue')
t.down()
square()

t.up()
t.goto(200,0)
t.color('orange')
t.down()
square()
```



As you can see we call our function three times to get three squares. This program is easier to read but it is still long. Now we try to make our program shorter using function with parameters. You can notice that these three drawn squares have different coordinates of the screen: first square starts from X=-200,Y=0; second square → from X=0,Y=0 and third square → from X=200,Y=0. Therefore, X position of square could be parameter in our new function. Parameters allow us to send information to the function by passing values to it as arguments inside its parentheses. Now we can modify our code:

c.

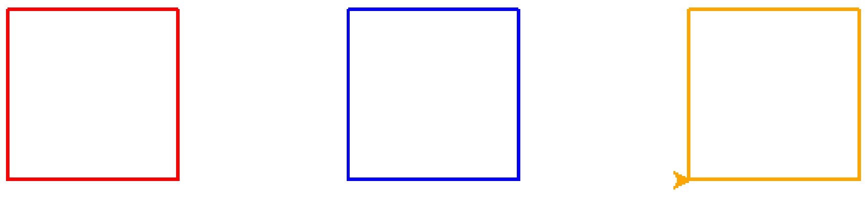
```
import turtle
t = turtle.Turtle()
t.pensize(2)

def square(X):
  t.up()
  t.goto(X,0)
  t.down()

  for i in range (4):
    t.fd(100)
    t.left(90)

t.color('red')
square(-200)
t.color('blue')
square(0)
t.color('orange')
square(200)
```



To make our program shorter we can introduce second parameter colour. Name this parameter clr. Let's try

d.

```
import turtle
t = turtle.Turtle()
t.pensize(2)

def square(X,clr):
  t.up()
  t.goto(X,0)
  t.down()
  t.color(clr)

  for i in range (4):
    t.fd(100)
    t.left(90)

square(-200, 'red')
square(0,'blue')
square(200,'orange')
```

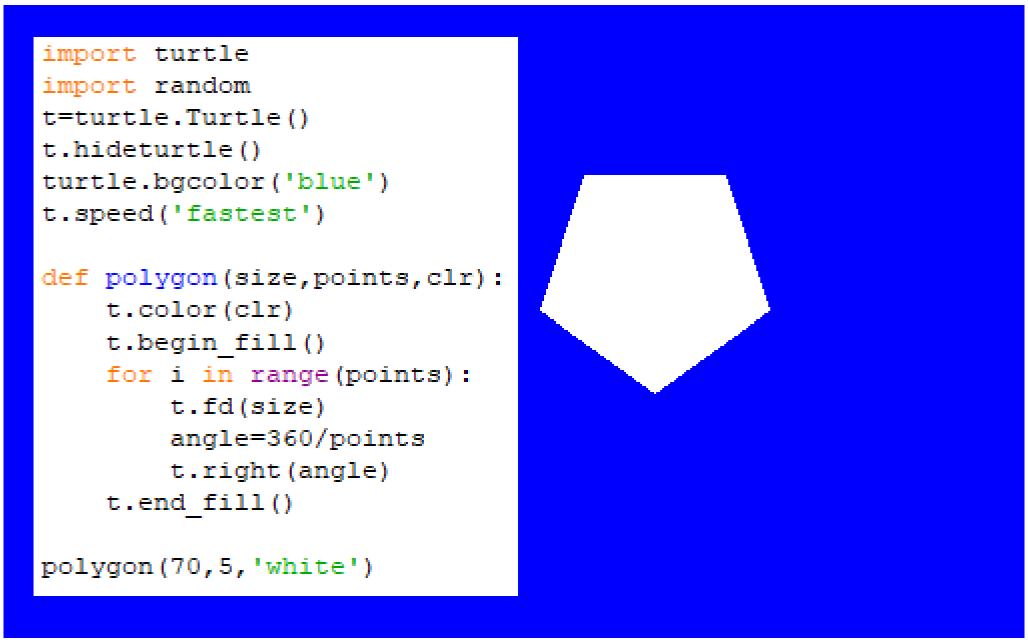


You can compare initial code (a) and final code (d). Version (a) consist of 24 lines, version (d)→ 14 lines which is significantly shorter. Bellow we would like to show a few Python programs that effectively use functions.

2. **Example #2** (Polygons)

Let’s draw a few colour polygons with the following parameters: size, points, clr. Size means the polygon size, points→number of corners, clr→colour of polygon.

Code and Result:

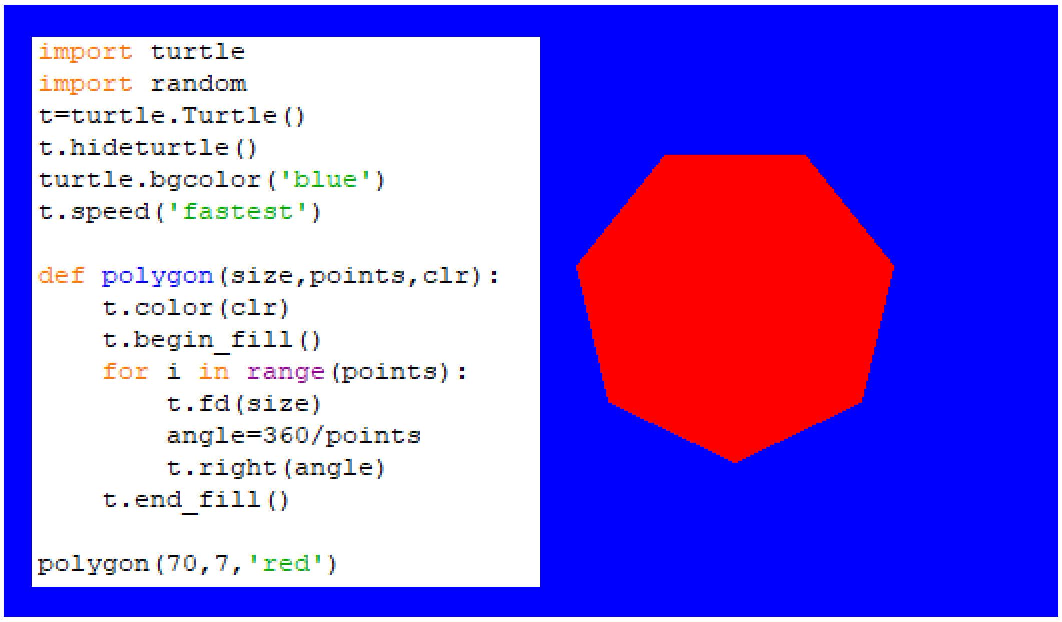


```
import turtle
import random
t=turtle.Turtle()
t.hideturtle()
turtle.bgcolor('blue')
t.speed('fastest')

def polygon(size,points,clr):
    t.color(clr)
    t.begin_fill()
    for i in range(points):
        t.fd(size)
        angle=360/points
        t.right(angle)
    t.end_fill()

polygon(70,5,'white')
```

Here’s polygon size is 50 pixels, number of corners is equal 5, and polygon colour is white. To calculate the turn angle for polygon with number of corners equal points (in our code) we use formula

`angle=360/points`

Below it is shown code and result for polygon with seven corners and size equal 70 pixels



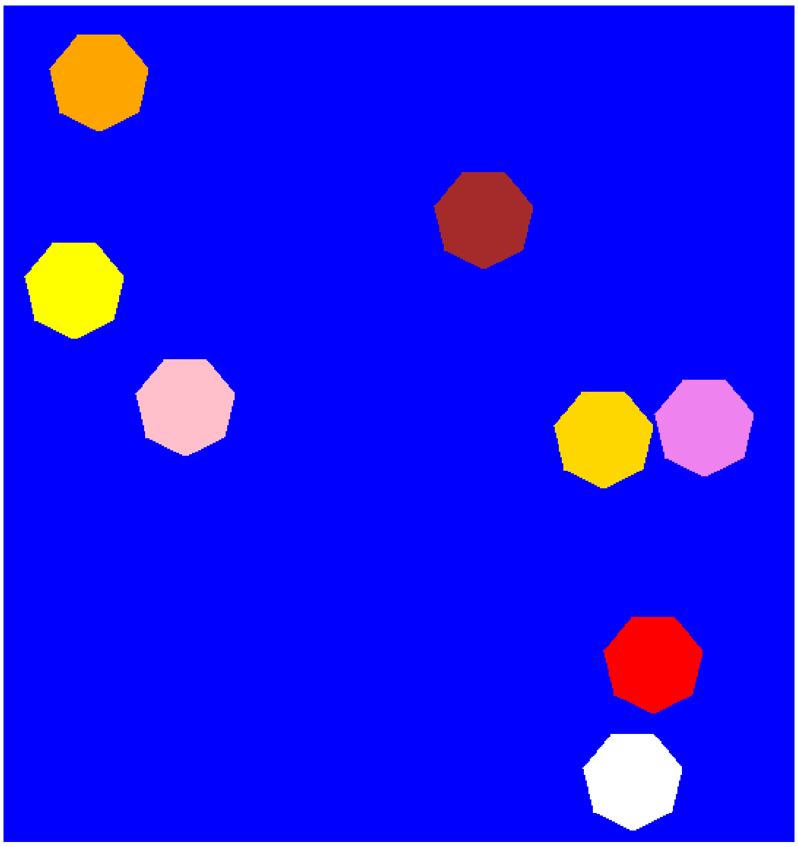
```
import turtle
import random
t=turtle.Turtle()
t.hideturtle()
turtle.bgcolor('blue')
t.speed('fastest')

def polygon(size,points,clr):
    t.color(clr)
    t.begin_fill()
    for i in range(points):
        t.fd(size)
        angle=360/points
        t.right(angle)
    t.end_fill()

polygon(70,7,'red')
```

Now we would like to draw a few different polygons on the window screen placed on the randomly located positions.

Code:

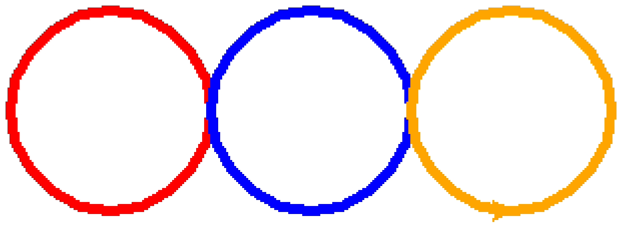
```python
import turtle
import random
t=turtle.Turtle()
t.hideturtle()
turtle.bgcolor('blue')
t.speed('fastest')
clr=['red','blue','gold','brown','violet','white','pink','orange','yellow']

def polygon(size,points,clr):
    t.color(clr)
    t.begin_fill()
    for i in range(points):
        t.fd(size)
        angle=360/points
        t.right(angle)
    t.end_fill()


for i in range(9):
    t.penup()
    t.goto(random.randint(-300,300),random.randint(-300,300))
    polygon(30,7,clr[i])
```

Result:



3. **Example #3** (Three circles)

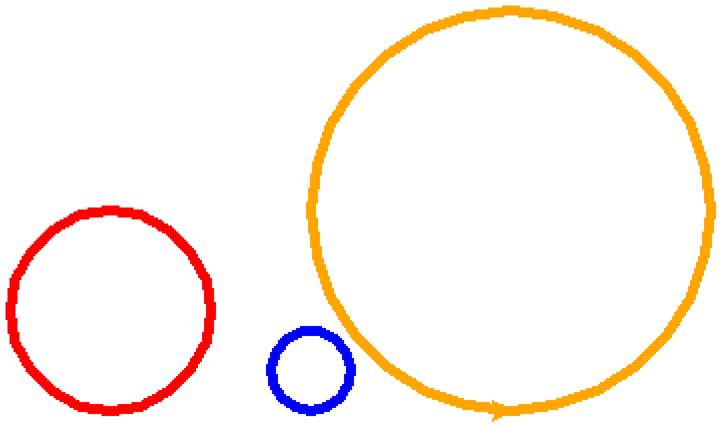
```
import turtle
t = turtle.Turtle()
t.pensize(5)

def circle(X,clr):
  t.up()
  t.goto(X,0)
  t.down()
  t.color(clr)
  t.circle(50)


circle(-100, 'red')
circle(0,'blue')
circle(100,'orange')
```



Here’s parameters: circle position and colour

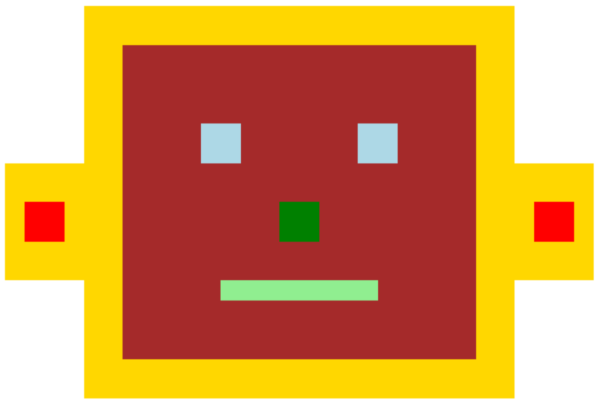
4. **Example #4** (Three circles with three parameters)

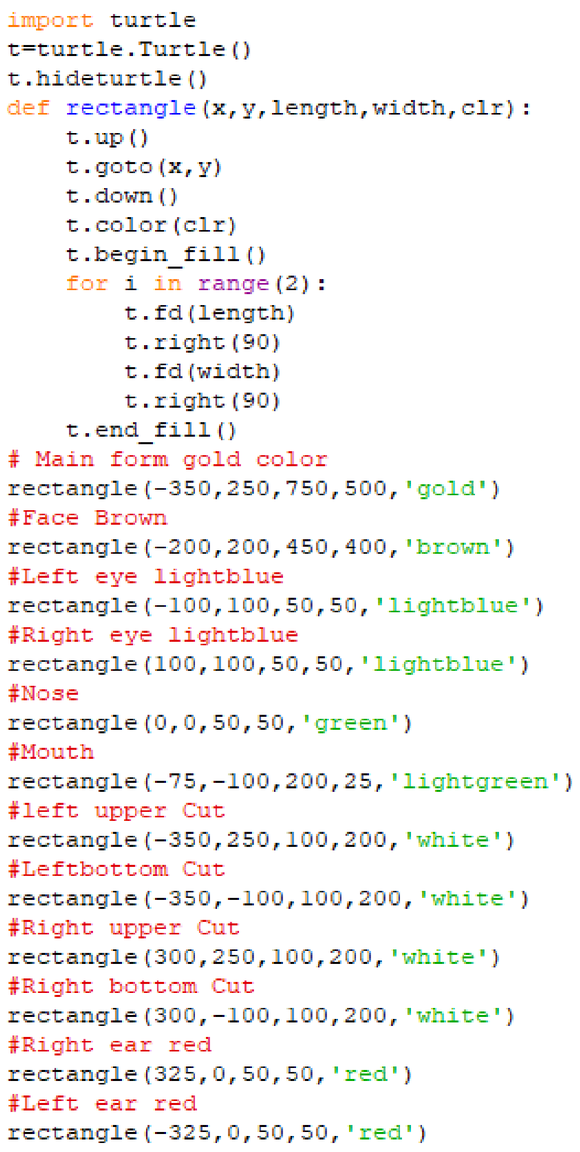
```
import turtle
t = turtle.Turtle()
t.pensize(5)

def circle(X,R,clr):
  t.up()
  t.goto(X,0)
  t.down()
  t.color(clr)
  t.circle(R)


circle(-100, 50,'red')
circle(0,20,'blue')
circle(100,100,'orange')
```



Here’s parameters: circle position, circle radius and colour.

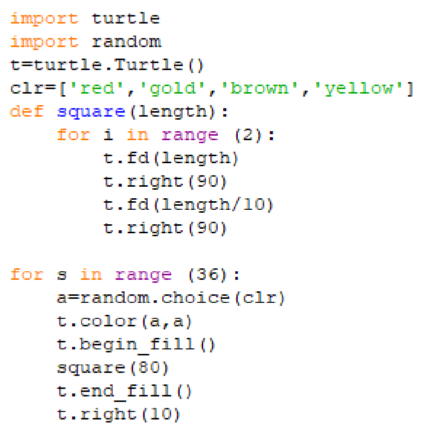
5. **Example #5** (Face created with function)



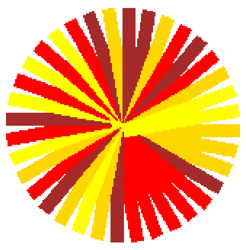


```python
import turtle
t=turtle.Turtle()
t.hideturtle()
def rectangle(x,y,length,width,clr):
    t.up()
    t.goto(x,y)
    t.down()
    t.color(clr)
    t.begin_fill()
    for i in range(2):
        t.fd(length)
        t.right(90)
        t.fd(width)
        t.right(90)
    t.end_fill()
# Main form gold color
rectangle(-350,250,750,500,'gold')
#Face Brown
rectangle(-200,200,450,400,'brown')
#Left eye lightblue
rectangle(-100,100,50,50,'lightblue')
#Right eye lightblue
rectangle(100,100,50,50,'lightblue')
#Nose
rectangle(0,0,50,50,'green')
#Mouth
rectangle(-75,-100,200,25,'lightgreen')
#left upper Cut
rectangle(-350,250,100,200,'white')
#Leftbottom Cut
rectangle(-350,-100,100,200,'white')
#Right upper Cut
rectangle(300,250,100,200,'white')
#Right bottom Cut
rectangle(300,-100,100,200,'white')
#Right ear red
rectangle(325,0,50,50,'red')
#Left ear red
rectangle(-325,0,50,50,'red')
```

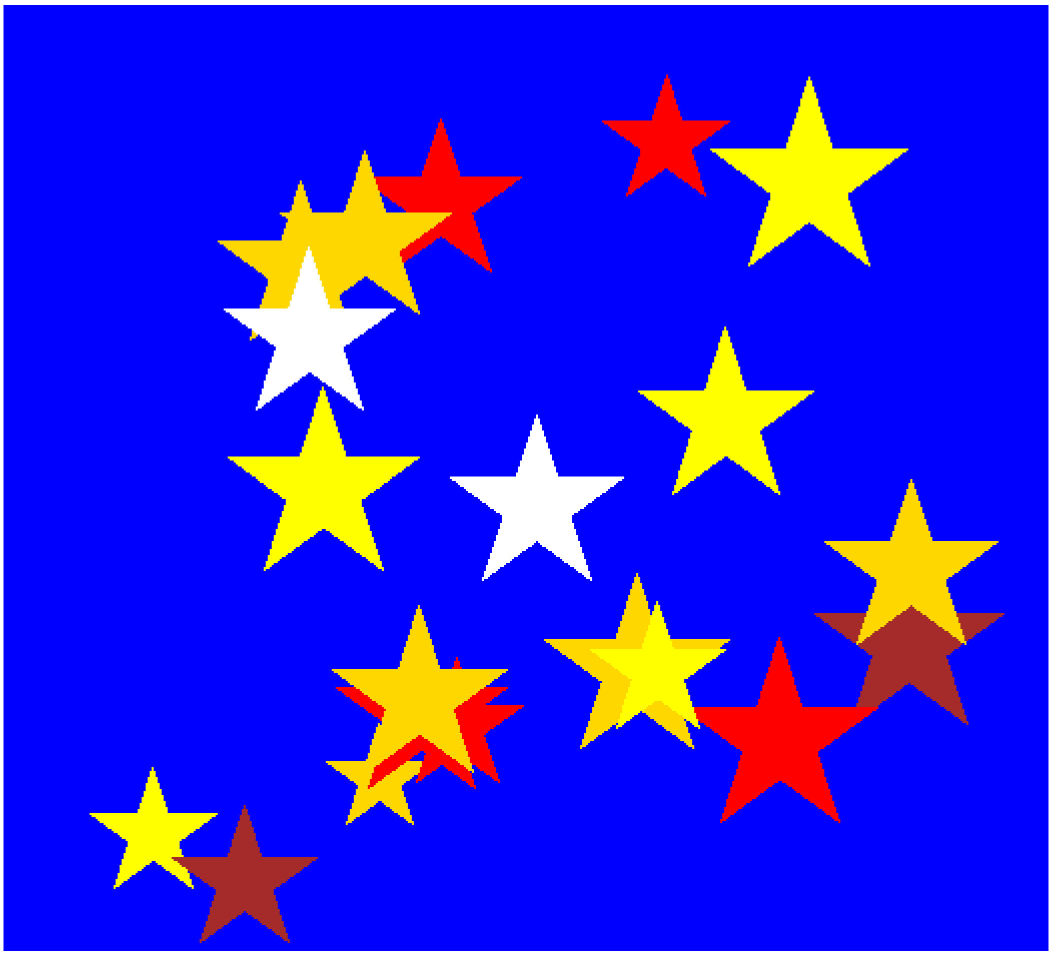
This code uses rectangle function with the following parameters::
x and y specify rectangle position on the screen;
third and fourth parameters length and width specify length and width of the rectangle;
Colour parameter clr determines the colour of the rectangle.

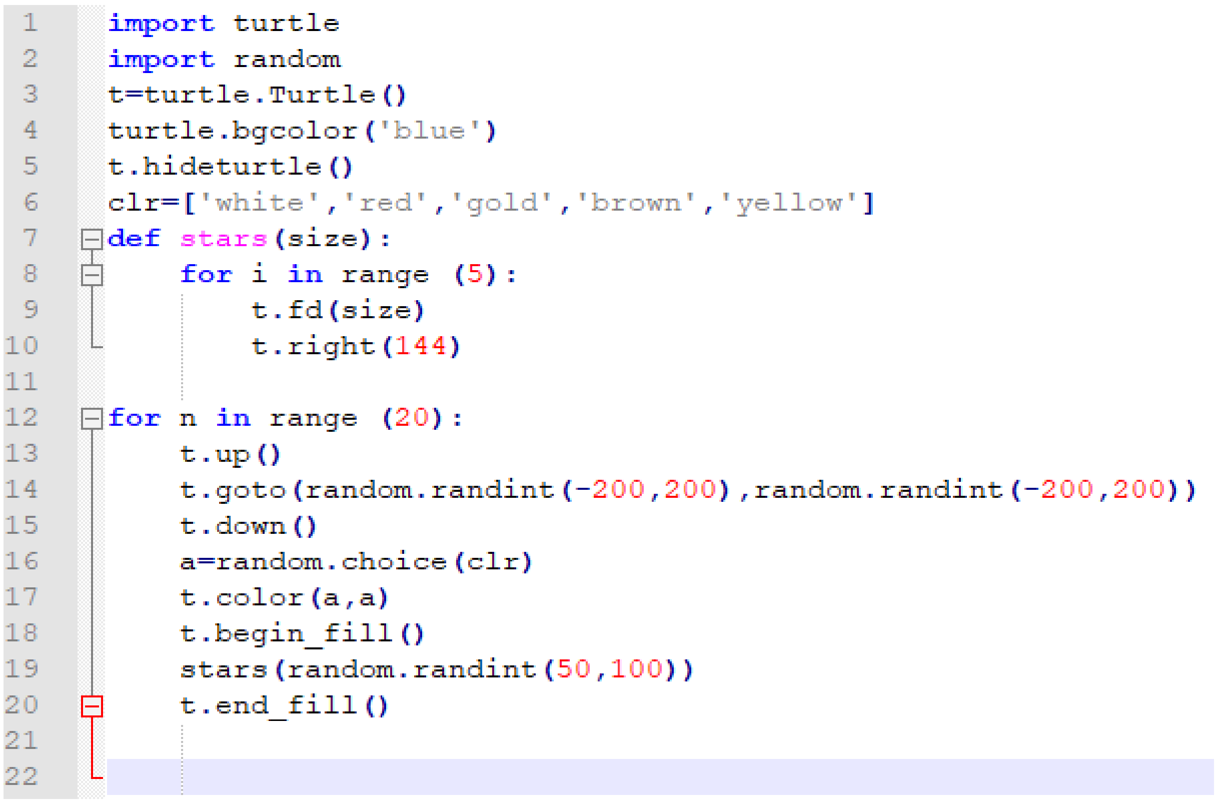


```python
import turtle
import random
t=turtle.Turtle()
clr=['red','gold','brown','yellow']
def square(length):
    for i in range (2):
        t.fd(length)
        t.right(90)
        t.fd(length/10)
        t.right(90)

for s in range (36):
    a=random.choice(clr)
    t.color(a,a)
    t.begin_fill()
    square(80)
    t.end_fill()
    t.right(10)
```



6. **Example #6** (Random position of random size stars on the screen)



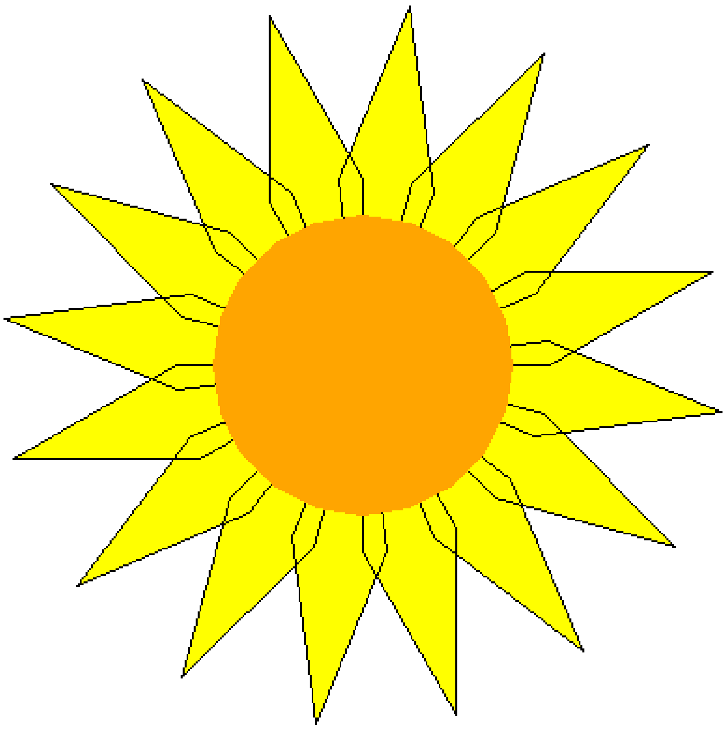


```python
import turtle
import random
t=turtle.Turtle()
turtle.bgcolor('blue')
t.hideturtle()
clr=['white','red','gold','brown','yellow']
def stars(size):
    for i in range (5):
        t.fd(size)
        t.right(144)

for n in range (20):
    t.up()
    t.goto(random.randint(-200,200),random.randint(-200,200))
    t.down()
    a=random.choice(clr)
    t.color(a,a)
    t.begin_fill()
    stars(random.randint(50,100))
    t.end_fill()

```

This code has created function (lines 7 to 10) which specifies stars with size parameter (Size parameter a random, line19). Code draws stars in random position of the screen (line14) with a random colour choice (lines16,17).

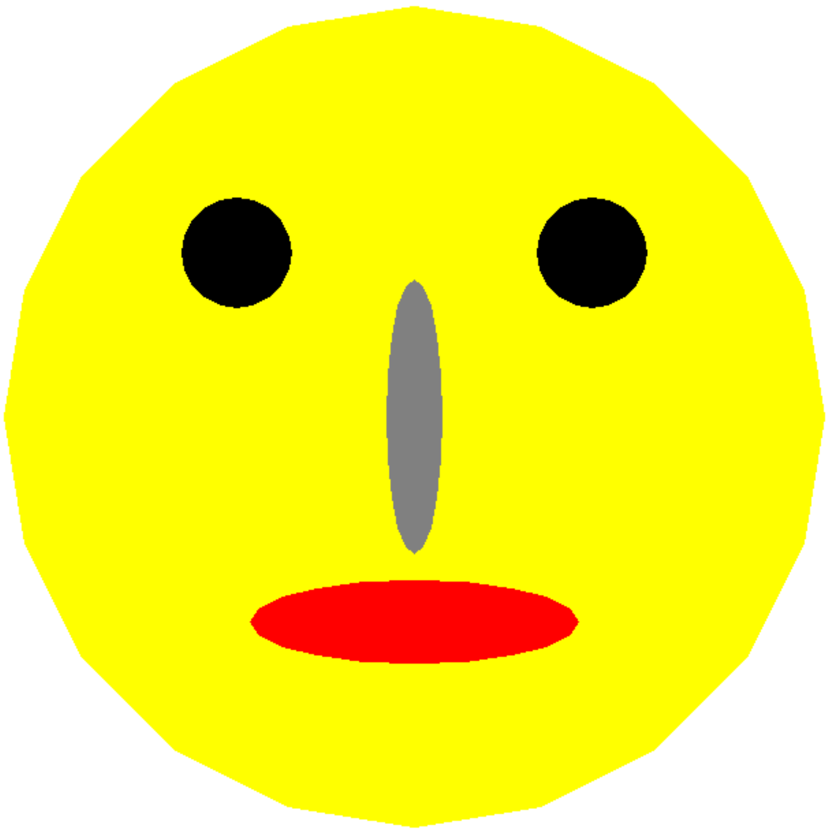
7. **Example #7** (Sun Flower)

```python
import turtle
t=turtle.Turtle('turtle')
t.color('black','yellow')
t.setheading(0)
def petal(r,angle):
    t.fd(r)
    t.left(angle)
    t.fd(r)
    t.left(180-angle)
    t.fd(r)
    t.left(angle)
    t.fd(r)

t.begin_fill()
for q in range (16):
    t.setheading(22.5*q)
    petal(100,30)
t.end_fill()
t1=turtle.Turtle('circle')
t1.color('orange')
t1.shapesize(8)
```



8. **Example #8** (Face from circles with functions)

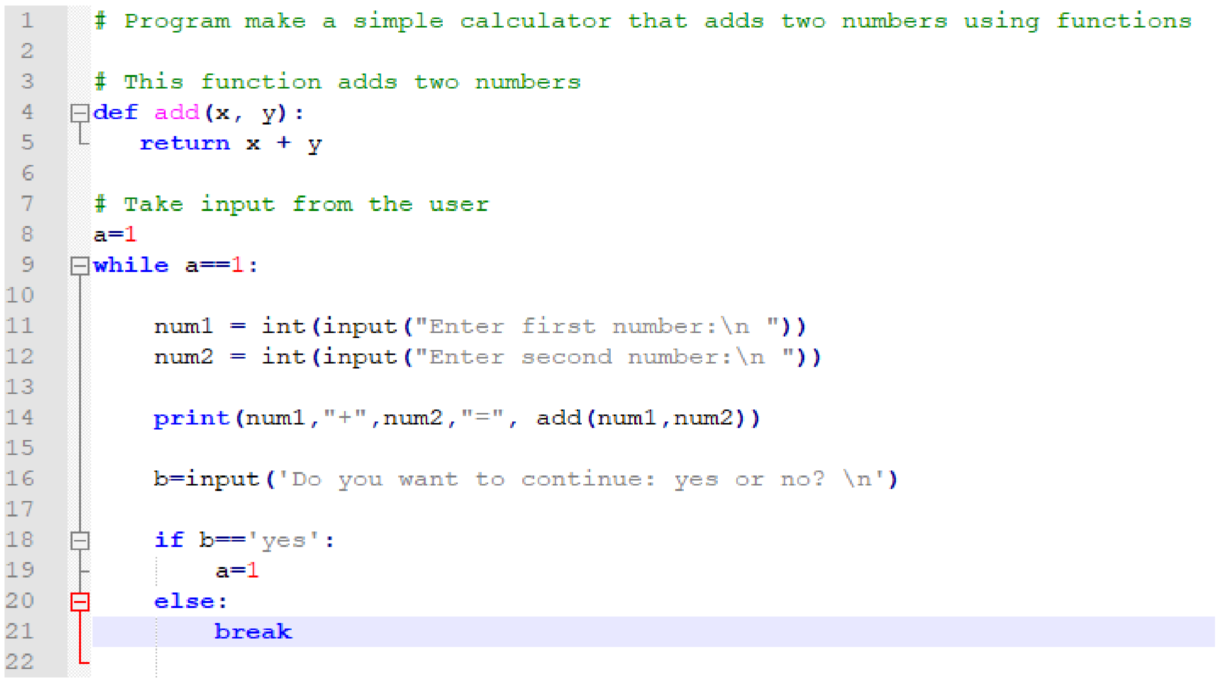
```python
import turtle
t=turtle.Turtle('circle')
t.setheading(0)

def face(clr,size1,size2,X,Y):
    t.color(clr)
    t.up()
    t.shapesize(size1,size2)
    t.goto(X,Y)
    t.stamp()

face('yellow',30,30,0,0)
face('red',3,12,0,-150)
face('black',4,4,-130,120)
face('black',4,4,130,120)
face('gray',10,2,0,0)
```



Functions in Math Calculations

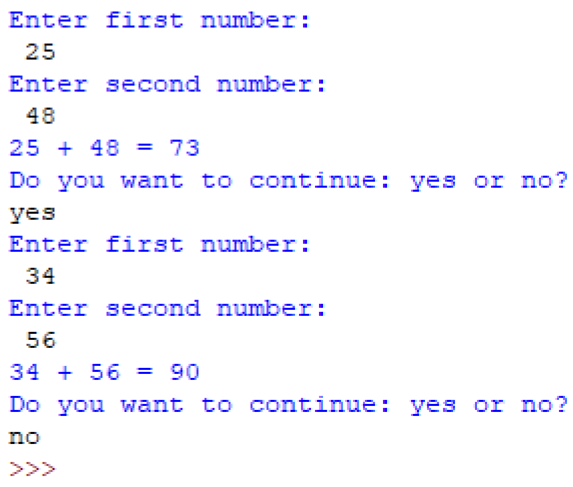
9. **Example #9** (simple calculator adds two numbers)

Code:



```
1   # Program make a simple calculator that adds two numbers using functions
2
3   # This function adds two numbers
4   def add(x, y):
5       return x + y
6
7   # Take input from the user
8   a=1
9   while a==1:
10
11      num1 = int(input("Enter first number:\n "))
12      num2 = int(input("Enter second number:\n "))
13
14      print(num1,"+",num2,"=", add(num1,num2))
15
16      b=input('Do you want to continue: yes or no? \n')
17
18      if b=='yes':
19          a=1
20      else:
21          break
22
```

Result:

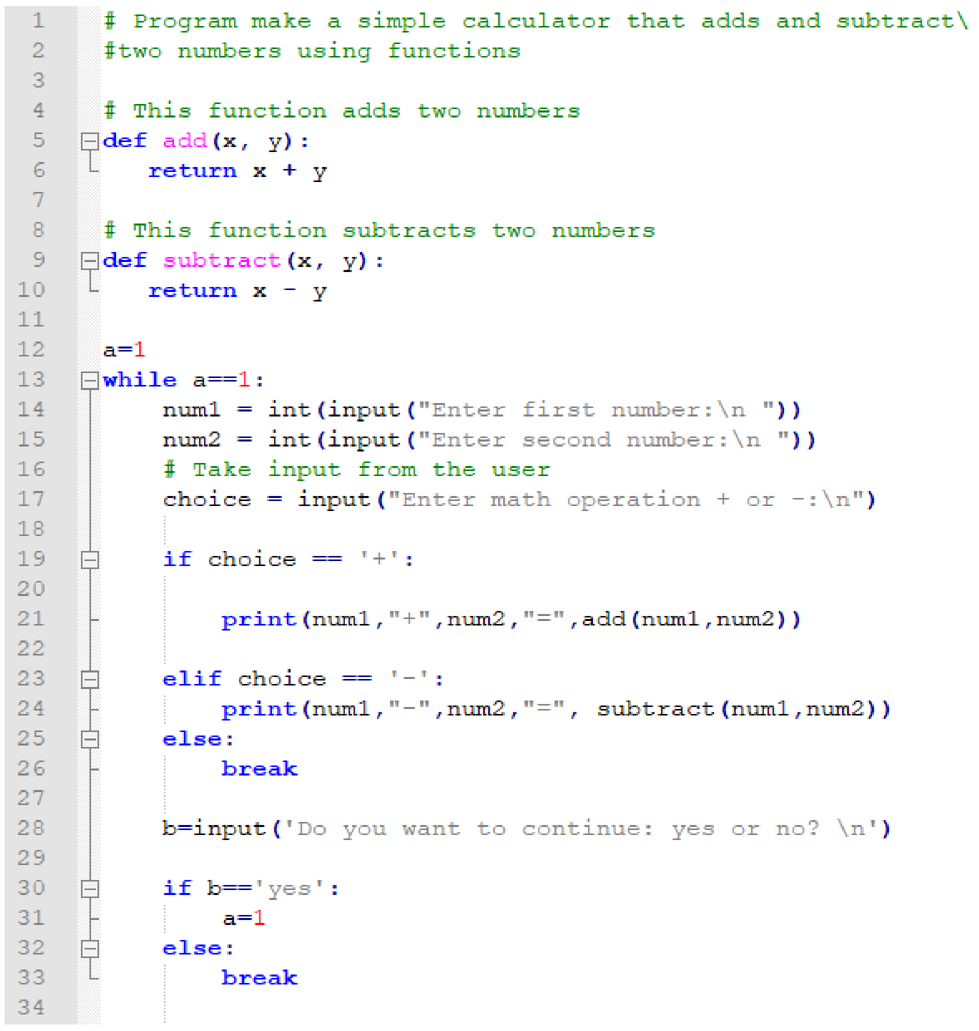


```
Enter first number:
 25
Enter second number:
 48
25 + 48 = 73
Do you want to continue: yes or no?
yes
Enter first number:
 34
Enter second number:
 56
34 + 56 = 90
Do you want to continue: yes or no?
no
>>>
```

In this code line 4 specifies function that adds two numbers, in line 14 we call this function.

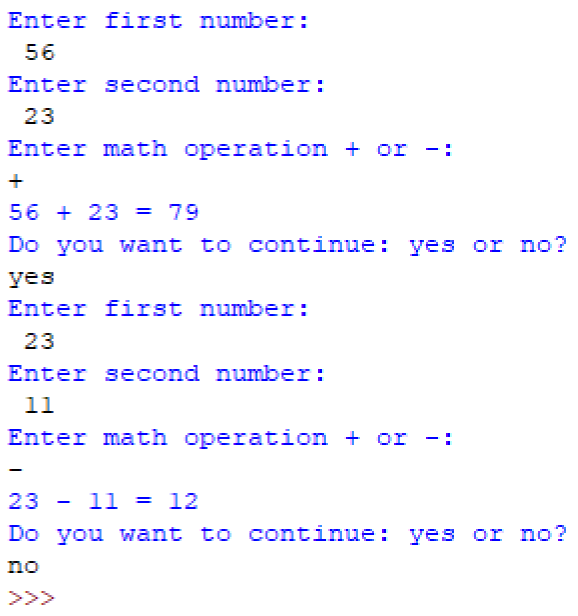
10. **Example #10** (Addition and Subtraction with function)

Code:



```python
# Program make a simple calculator that adds and subtract\
#two numbers using functions

# This function adds two numbers
def add(x, y):
    return x + y

# This function subtracts two numbers
def subtract(x, y):
    return x - y

a=1
while a==1:
    num1 = int(input("Enter first number:\n "))
    num2 = int(input("Enter second number:\n "))
    # Take input from the user
    choice = input("Enter math operation + or -:\n")

    if choice == '+':

        print(num1,"+",num2,"=",add(num1,num2))

    elif choice == '-':
        print(num1,"-",num2,"=", subtract(num1,num2))
    else:
        break

    b=input('Do you want to continue: yes or no? \n')

    if b=='yes':
        a=1
    else:
        break

```

Result:



```
Enter first number:
 56
Enter second number:
 23
Enter math operation + or -:
+
56 + 23 = 79
Do you want to continue: yes or no?
yes
Enter first number:
 23
Enter second number:
 11
Enter math operation + or -:
-
23 - 11 = 12
Do you want to continue: yes or no?
no
>>>
```

Here’s we use two functions: one for addition of two numbers and another one for subtraction of two numbers. Loop while is to provide call functions a lot of times, till you want to make calculations.