Hi guys, this is Akshansh (+91 8384891269, akshanshofficial@gmail.com (mailto:akshanshofficial@gmail.com)) and we are here to discuss lecture 08; a continuation of lecture07 that talked about the function in python. We discussed about the basics of the function in that lecture note. Here we will learn something new called fruitful function.

many of the python functions we have used, such as the math functions, produce return values. But the functions we've written are all void: they have an effect, like printing a value or song. They do not return value. In this lecture we are going to learn how to write fruitful functions.

## return values

Calling a function gererates a return value, which we usually assign to a variable or use as part of an expression.

In [16]:

```
import math
e=math.exp(1.0)
```

In this lecture, we are finally going to write some fruitful functins. The first example is area, which returns the area of circle with the given radius-

In [17]:

```
def area(radius):
    area=math.pi*radius**2
    return Area
```

In a fruitful function the return statement includes an expression. This statement meeans: Return immediately from this function and use the following expression as a return value.

code written above can be written as following too-

In [18]:

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

In [19]:

```
area(5)
```

Out[19]:

```
78.53981633974483
```

Sometimes it is useful to have multiple return statements, one in each branch of a conditional :

Look at the following code, it will print absolute value. If x<0 that means negative, it will make it positive with negative signs becuase two negatives is positive. Else, it will print same result

In [20]:

```
def absolute_value(x):
    if x<0:
        return -x
    else:
        return x
```

In [21]:

```
absolute_value(-3)
```

Out[21]:

```
3
```

In [22]:

```
absolute_value(3)
```

Out[22]:

```
3
```

In [23]:

```
absolute_value(0)
```

Out[23]:

```
0
```

since these return statements are in an alternative conditional, only one has run. As soon as a return statement runs, function gets terminated and rest of the code is called dead code in this case.

## Incremental development

As you write larger functions, you might find yourself spending more time debuggin. To deal with increasingly complex program, you might want to try a process called incremental development.

The goal od incremental development is to avoid long debuggin sessions by adding and testing onlu a small amount of code at a time.

Suppose you want to know the distance between two points or cordinates. You remeber the distance formula, do you?

### distance = {(x2-x1)^2 +(y2-y1)^2}^1/2

in this you have to take 4 parameteres or arguments. It makes program more complicated. What you can do is - check the program in each step. Syntax should be correct. We will not make whole program in just once but we will be making in steps and checking each steps if it is correct or not.

let's check if it is takingf arguments more than one. If it runs succesfully that means we can move ahead -

In [24]:

```python
def distance(x1,y1,x2,y2):
    return 0.0
```

In [25]:

```python
distance(1,2,3,5)
```

Out[25]:

```
0.0
```

see it returned 0.0 that means it run proprely. Program is good to go. Now move ahead for adding new things to calculate distance

In [26]:

```python
def distance(x1,y1,x2,y2):
    dx=x2-x1
    dy=y2-y1
    print('dx is ', dx)
    print('dy is ', dy)
    return 0.0
```

In [27]:

```python
distance(1,2,3,5)
```

```
dx is  2
dy is  3
```

Out[27]:

```
0.0
```

it ran properly that means it is syntatically correct. Now we can move ahead .

In [28]:

```python
def distance(x1,y1,x2,y2):
    dx=x2-x1
    dy=y2-y1
     #squaring both of these and adding them
    dsquaredTotal = dx**2 + dy**2
     #taking squareroot of the dsqauredTotal
    result=math.sqrt(dsquaredTotal)
     #returning reult
    return result
```

let's give it a run

In [29]:

```
distance(1,2,3,4)
```

Out[29]:

```
2.8284271247461903
```

see this can be done to avoid complicated debugging. You don't need to follow the same pattern, it is just my suggestion to keep it in this way. If you wan't to you can do it, or if not comfortable, don't do this.

## composition

Do you know that you can call one func within anothere fucntion. As an example, we'll write a func that takes two points, the center of the circle and a point on the perimeter, and then computes the area of the circle.

suppose we have a point on centre and coordinates are storesd as xc and yc. Another point is on perimeter and cordinates of the same is stoerd as xp, yp. Distance between these two points will be radius of the circle because if you join perimeter to the center of the circle, you get the radius-

radius = distance(xc,yc,xp,yp)

> result=area(radius)

Encapsulating these steps in a function, we get

In [31]:

```
def circle_area(xc,yc,xp,yp):
    radius=distance(xc,yc,xp,yp)
    result=area(radius)
    return result
```

see that we created another fucntion to obtain the area of the circle and named it circle_area. In this function we used two other functions, which were created earlier in this/previous lecture. (distance and area are those function). This is called composing the function with each others.

You can write the above program more concisely

In [33]:

```
def circle_area(xc,yc,xp,yp):
    return area(distance(xc,yc,xp,yp))
```

let's give it a test run, suppose (3,4) is at center and (6,3) is at perimeter

In [34]:

```
circle_area(3,4,6,3)
```

Out[34]:

```
31.41592653589794
```

## Random Numbers

A good program is always predictable and it provides tha output as desired. But sometimes unpredictivity is also required. Think about the game if O/P will remain same, players will get bored of it so easily so we need to genreate random results.

random module provides functions that generate pseudorandom numbers. It returns between 0 to 1.00

In [37]:

```
import random
for num in range(10):
    random_num=random.random()
    print(random_num)
```

```
0.6342059164310964
0.9407585410270621
0.4654872599496107
0.1146842923126804
0.7857086066926336
0.5808546799847509
0.3977396398312927
0.4825185116909737
0.495301695391942
0.10467220319179538
```

The functions randint takes parameters low and high and returns an integer between those(includin both)

In [38]:

```
random.randint(5,10)
```

Out[38]:

```
8
```

In [45]:

```
random.randint(5,10)
```

Out[45]:

```
5
```

In [40]:

```
random.randint(5,10)
```

Out[40]:

```
7
```

you can choose a list/tuble/set from where you want to pick the number but in random order. To do this, you will be using choice

In [46]:

```
t=[1,5,8,3,6]
```

In [47]:

```
random.choice(t)
```

Out[47]:

```
1
```

In [48]:

```
random.choice(t)
```

Out[48]:

```
3
```

In [ ]:

```

```