Functions

Neeraj Jain

September 11, 2021

1 Working on the Problem: Correct below code with minimal changes

[46]:

```
num=input('enter a number:')
try:
    i=1
    fac=1
    num=int(num)
    if num>0:
        while num>=i:
            fac=fac*i
            print('factorial=', fac)
            i=i+1
    elif num<0:
        print('the factorial is not possible')
except:
    print('only integers are allowed')
if num==0:
    print('the factorial is 1')
```

```
enter a number:4
factorial= 1
factorial= 2
factorial= 6
factorial= 24
```

[2]:

```
#Solution by Aditi
num=input('enter a number:')
try:
    i=1
    fac=1
    num=int(num)
    if num>0:
        while num>=i:
            fac=fac*i
            i=i+1
        print('factorial=', fac) #move print out of while loop
    elif num<0:
        print('the factorial is not possible')
except:
    print('only integers are allowed')
if num==0:
    print('the factorial is 1')
```

```
enter a number:0
the factorial is 1
```

2 Functions

[4]:

```
import math
```

[5]:

```
math.sqrt(34)
```

[5]:

```
5.830951894845301
```

[10]:

```
math.pi
```

[10]:

```
3.141592653589793
```

[9]:

```
b = 5
b
```

[9]:

```
5
```

[11]:

```
pi = 3.141592653589793
pi
```

[11]:

```
3.141592653589793
```

[12]:

```
#so math.pi() will return error
math.pi() #will be treated as function by python
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-12-6027e8555373> in <module>
          1 #so math.pi() will return error
    ----> 2 math.pi() #will be treated as function by python


        TypeError: 'float' object is not callable
```

[16]:

```
math.factorial(3)
```

[16]:

```
6
```

[17]:

```
math.factorial(8)
```

[17]:

```
40320
```

3 Defining own functions

There are three components of a function: 1. Name of the function: addtwo 2. Arguments: Inputs entered by user : a, b 3. Body of the function

see code below for function `addtwo`:

[ ]:

```
def addtwo(a, b):
    a + b
```

[20]:

```
addtwo(12, 13)
```

No output will be produced unless used `print` or `return` in the function.

[48]:

```
def addtwo(a, b):
    print(a + b)
```

[49]:

```
addtwo(12, 13)
```

```
25
```

Output produced using `print` can not be saved in a variable. For example try:

[51]:

```
a = addtwo(12, 13)
a #no output produced. Check second line in output box in empty
type(a)
```

```
25
```

[51]:

```
NoneType
```

Better to use `return` It is advisable to use `return` in the function not `print` to save output to a new variable. See example:

[52]:

```
def addtwo(a, b):
    c = a + b
    return(c)
```

[53]:

```
e = addtwo(12, 42)
e
```

[53]:

```
54
```

Calling value of pi To get the value of pi use math library in python. The value of the pi can be called using `math.pi`. Donot use `math.pi()`. Pi value is stored in variable name `pi` in math variable. Using bracket after pi means you are calling pi as a function.

[24]:

```
pi12 = addtwo(12, math.pi)
```

```
15.141592653589793
```

Using more than one return in a function If used more than one `return` in a function, function will return the value of first `return`, and no further evaluation takes place. For exmple, in below code, function `ret` will return the value of a.

[33]:

```
def ret(a, b):
    return(a)
    return(b)
```

[34]:

```
ret(10, 12)
```

[34]:

```
10
```

Practice: Normal Density Define function for normal density, that takes 3 inputs `x, mu, and sigma`. The formula for the normal density is:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \qquad \sigma > 0 \tag{1}$$

[43]:

```
import math
def dnorm(x, mu, sigma):
    if sigma <= 0:
        return("Sigma should be positive")
    a = 1 / (sigma * (2 * math.pi)**0.5)
    b = -0.5 * ((x - mu) / sigma)**2
    result = math.exp(b) * a
    return(result)
```

[45]:

```
dnorm(12, 12, 5)
```

[45]:

```
0.07978845608028655
```

Homework

1. Define function `iseven` that return `True` if number is even, otherwise `False`. Try to put error message for all types of possible incorret argumetns (like, negative values, non-integers, strings (like three) etc.

2. Define function `isprime` that return `True` if number is prime, otherwise `False`. Try to put error message for all types of possible incorret argumetns (like, negative values, non-integers, strings (like three) etc.