باب-7

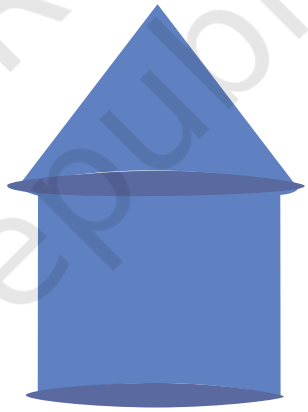
فنکشن
(FUNCTIONS)

5196CH07

7.1 تعارف

اب تک ہم نے کئی پروگرام تحریر کر لیے ہیں اور شاید اس حقیقت سے آشنا ہو چکے ہیں کہ جیسے جیسے مسئلہ پیچیدہ ہوتا جاتا ہے پروگرام میں لائنوں کی تعداد بڑھتی جاتی ہے جس کی وجہ سے پروگرام بہت زیادہ ضخیم نظر آتا ہے اور اس کے ساتھ کام کرنا مشکل ہو جاتا ہے۔ مندرجہ ذیل مسئلے پر غور کیجیے:

ایک کمپنی صارفین کی ضروریات کے مطابق خیمہ تیار کرتی ہے۔ خیمہ ایک ایسے استوانہ کی شکل میں ہے جس کا بالائی حصہ مخروطی ہے۔



شکل 7.1: خیمہ کی شکل

کمپنی ہر ایک خیمہ کی قیمت فروخت کا تعین کرنے کے لیے مندرجہ ذیل کاموں کو انجام دیتی ہے:

1۔ خیمے کے متعلق صارف کی ضروریات، مثلاً

a) اونچائی

b) نصف قطر

c) مخروطی حصے کی ترچھی اونچائی

2۔ استعمال کیے جانے والے کینوس کے رقبے کی تحسیب کیجیے۔

3۔ خیمہ بنانے میں استعمال کیے جانے والے کینوس کی قیمت کا حساب لگائیے۔

4۔ 18 فیصد ٹیکس شامل کرتے ہوئے خیمہ کے لیے صارف کے ذریعے ادا کی جانے والی کل رقم کی تحسیب کیجیے۔

''جیسے ہی آپ پیچیدہ الگورتم کے لیے پروگرام تحریر کرنے میں کامیاب ہو جاتے ہیں تو یہ پروگرام بہت تیز رفتار سے چلیں گے۔ کمپیوٹر کو یہ ضرورت نہیں ہوتی کہ وہ الگورتم کو سمجھے۔ اس کا کام صرف پروگراموں کو چلانا ہے۔''

— آر۔ ٹارجن
(R. Tarjan)

اس باب میں

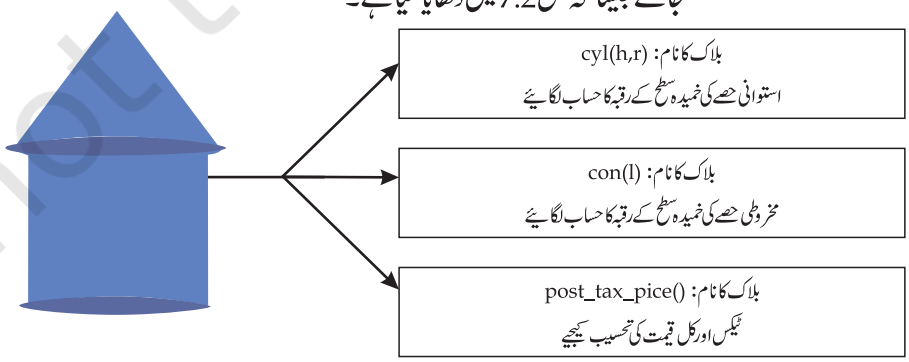
- فنکشن کا تعارف
- استعمال کنندہ کے ذریعے تعریف شدہ فنکشن
- متغیر کا اسکوپ
- پائتھن معیاری لائبریری

قابل ادا رقم کی فوری اور بالکل درست تخمیب کے لیے کمپنی نے ایک پروگرام کی تشکیل کی جیسا کہ پروگرام 7-1 میں دکھایا گیا ہے۔

پروگرام 7-1 خیمہ کے لیے قابل ادا رقم کی تخمیب کے لیے پروگرام

```
#Program 7-1
#Program to calculate the payable amount for the tent without
#functions

print( "Enter values for the cylindrical part of the tent in
meters\n")
h = float(input("Enter height of the cylindrical part: "))
r = float(input("Enter radius: "))

l = float(input("Enter the slant height of the conical part in
meters: "))
csa_conical = 3.14*r*l          #Area of conical part
csa_cylindrical = 2*3.14*r*h    #Area of cylindrical part

# Calculate area of the canvas used for making the tent
canvas_area = csa_conical + csa_cylindrical
print("The area of the canvas is",canvas_area,"m^2")

#Calculate cost of the canvas
unit_price = float(input("Enter the cost of 1 m^2 canvas: "))
total_cost= unit_price * canvas_area
print("The total cost of canvas = ",total_cost)

#Add tax to the total cost to calculate net amount payable by the
#customer
tax = 0.18 * total_cost;
net_price = total_cost + tax
print("Net amount payable = ",net_price)
```

مذکورہ بالا مسئلہ کو حل کرنے کا ایک دوسرا طریقہ یہ ہے کہ پروگرام کو کوڈ کے مختلف بلاکوں میں تقسیم کر لیا جائے جیسا کہ شکل 7.2 میں دکھایا گیا ہے۔



شکل 7.2: خیمہ کی لاگت کی تخمیب

ایک کمپیوٹر پروگرام کو مختلف ناموں اور مخصوص ضابطوں کے ساتھ کوڈ کے جداگانہ نوعیت والے آزاد بلاکوں یا ذیلی مسائل میں تقسیم کرنے کا طریقہ ماڈیولر پروگرامنگ کہلاتا ہے۔ اس باب میں ہم اس طریقۂ کار کے فوائد کا مطالعہ کریں گے۔

7.2 فنکشن (Functions)

پروگرامنگ میں فنکشن کا استعمال جداسازی (Modularity) اور بار بار استعمال کے جانے کی خصوصیت (Reusability) کے حصول کا ایک طریقہ ہے۔ فنکشن کی تعریف ان ہدایات کے گروپ نام کے طور پر کی جاسکتی ہے جو طلب کرنے پر ایک مخصوص کام کو انجام دیتی ہیں۔ فنکشن کو ایک مرتبہ اگر متعین کر دیا جائے تو اسے پروگرام کے مختلف مقامات پر بار بار طلب کیا جاسکتا ہے اور ہر مرتبہ فنکشن کے سبھی کوڈ کو تحریر کرنے کی ضرورت پیش نہیں آتی ہے۔ علاوہ ازیں اسے دوسرے فنکشن کے اندر سے محض فنکشن کا نام لکھ کر اور مطلوبہ پیرامیٹر (اگر کوئی ہے) کو پاس کر کے بھی طلب کیا جاسکتا ہے (سیکشن 7.3) پروگرام کوڈ تحریر کرنے کے دوران جتنے چاہے اتنے فنکشن متعین کرسکتا ہے۔ پروگرام 7-1 کو استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کر کے دوبارہ تحریر کیا گیا ہے جیسا کہ پروگرام 7-2 میں دکھایا گیا ہے۔

پروگرام 7-2 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کے قابلِ ادائیگی رقم کے تحسیب کا پروگرام

```
#Program 7-2
#Program to calculate the cost of tent
#function definition
def cyl(h,r):
     area_cyl = 2*3.14*r*h    #Area of cylindrical part
     return(area_cyl)

#function definition
def con(l,r):
     area_con = 3.14*r*l      #Area of conical part
     return(area_con)

#function definition
def post_tax_price(cost):     #compute payable amount for the tent
     tax = 0.18 * cost;
     net_price = cost + tax
     return(net_price)

print("Enter values of cylindrical part of the tent in meters:")
h = float(input("Height: "))
r = float(input("Radius: "))
 csa_cyl = cyl(h,r)  #function call

l = float(input("Enter slant height of the conical area in meters: "))
csa_con = con(l,r)  #function call
```

```
#Calculate area of the canvas used for making the tent
canvas_area = csa_cyl + csa_con
print("Area of canvas = ",canvas_area," m^2")

#Calculate cost of canvas
unit_price = float(input("Enter cost of 1 m^2 canvas in rupees: "))
total_cost = unit_price * canvas_area
print("Total cost of canvas before tax = ",total_cost)
print("Net amount payable (including tax) = ",post_tax_price(total_
cost))
```

اگر ہم پروگرام 7-1 اور پروگرام 7-2 کا موازنہ کریں تو صاف ظاہر ہے کہ پروگرام 7-2 زیادہ منظّم نظر آتا ہے اور اسے پڑھنا بھی نسبتاً آسان ہے۔

7.2.1 فنکشن کے فوائد (The advantages of Function)

فرض کیجیے کہ مذکورہ بالا کمپنی مزید دوسری قسم کا ایسا خیمہ بنانے کا فیصلہ کرتی ہے جس کا قاعدہ مستطیل نما ہے جب کہ بالائی حصہ وہی ہے۔ اس صورت حال میں فنکشن con(l,r) کو طلب کر کے موجودہ کوڈ کے کچھ حصے کو دوبارہ استعمال کیا جا سکتا ہے۔ اگر کمپنی دوسرے پروڈکٹ تیار کرتی ہے یا خدمات مہیا کرتی ہے، اور جہاں 18% ٹیکس کا اطلاق ہوتا ہے تو پروگرام فنکشن post_tax_price(cost) کو براہ راست استعمال کر سکتا ہے۔

چنانچہ پروگرام میں فنکشن کو استعمال کرنے کے فوائد مندرجہ ذیل ہیں:

- پروگرام کو بالخصوص طویل کوڈ کے معاملے میں پڑھنے میں آسانی کیوں کہ فنکشن کے استعمال سے پروگرام کو بہتر طور پر منظّم کیا جا سکتا ہے اور اسے سمجھنا بھی آسان ہے۔
- کوڈ کی لمبائی کم ہو جاتی ہے کیوں کہ پروگرام میں ایک ہی کوڈ کو مختلف جگہوں پر بار بار لکھنے کی ضرورت نہیں ہوتی ہے۔ اس کی وجہ سے پروگرام میں ہونے والی غلطیوں کو دور کرنا (Debugging) بھی آسان ہو جاتا ہے۔
- دوبارہ استعمال کی خصوصیت میں اضافہ ہوتا ہے، کیوں کہ فنکشن کو کسی دوسرے فنکشن یا دوسرے پروگرام سے کال (طلب) کیا جا سکتا ہے۔ چنانچہ ہم پہلے سے متعین (تعریف شدہ) فنکشن کو دوبارہ استعمال کر سکتے ہیں یا اس پر انحصار کر سکتے ہیں اور ایک ہی کوڈ کو بار بار لکھنے سے بھی بچا جا سکتا ہے۔
- کام کو ٹیم کے اراکین کے درمیان باسانی تقسیم کیا جا سکتا ہے اور کام کو ایک ساتھ مکمل کیا جا سکتا ہے۔

7.3 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن

فنکشن کی بازاستعمال کی خصوصیت سے استفادہ کرتے ہوئے، پائتھن میں اسٹینڈرڈ لائبریری (سیکشن 7.5) کے تحت فنکشن کی بہت بڑی تعداد پہلے ہی دست یاب ہے۔ ہم اپنے پروگرام میں ان فنکشن کو متعین کیے بغیر

براہ راست طلب کر سکتے ہیں۔ تاہم، اسٹینڈرڈ لائبریری فنکشن کے علاوہ ہم پروگرام لکھنے کے دوران خود اپنے فنکشن متعین کر سکتے ہیں۔ اس قسم کے فنکشن کو استعمال کنندہ کے ذریعے تعریف شدہ فنکشن (User defined functions) کہا جاتا ہے۔ چنانچہ ایسا فنکشن جسے پروگرامر کی ضروریات کے مطابق کسی کام کو انجام دینے کے لیے متعین کیا جاتا ہے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کہلاتا ہے۔

7.3.1 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کی تشکیل کرنا

فنکشن کی تعریف def [یعنی انگریزی زبان کے لفظ define کا مخفف] سے شروع ہوتی ہے۔ استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کی تشکیل کے لیے سنٹیکس مندرجہ ذیل ہے:

```
def<Function name> ([parameter 1, parameter 2,....]):
    set of instructions to be executed
    [return <value>]
```

فنکشن ہیڈر

فنکشن کا متن [فنکشن ہیڈر کے اندر انڈینٹیڈ ہونا چاہیے]

- "[ ]" میں درج کیے گئے آئٹم پیرامیٹر کہلاتے ہیں اور یہ اختیاری ہوتے ہیں۔ لہٰذا ایک فنکشن میں پیرامیٹر ہو بھی سکتے ہیں اور نہیں بھی۔ مزید یہ کہ فنکشن کسی ویلیو کو ظاہر کر بھی سکتا ہے اور نہیں بھی۔
- فنکشن کا ہیڈر ہمیشہ کولن (:) کے ساتھ ختم ہوتا ہے۔
- فنکشن کا نام منفرد نوعیت کا ہونا چاہیے۔ آئڈینٹی فائر (شناخت کنندہ) کے تسمیہ کے لیے جو قوانین ہیں ان کا اطلاق فنکشن کے تسمیہ پر بھی ہوتا ہے۔
- فنکشن انڈینٹیشن کے باہر درج کیے گئے بیانات کو فنکشن کا حصہ تصور نہیں کیا جاتا ہے۔

پروگرام 7-3 2 اعداد کو جمع کرنے اور ان کے حاصل جمع کو ظاہر کرنے کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن لکھیے۔

```
#Program 7-3
#Function to add two numbers
#The requirements are listed below:
  #1. We need to accept 2 numbers from the user.
  #2. Calculate their sum
  #3. Display the sum.

#function definition
def addnum():
    fnum = int(input("Enter first number: "))
    snum = int(input("Enter second number: "))
    sum = fnum + snum
    print("The sum of ",fnum,"and ",snum,"is ",sum)

#function call
addnum()
```

فنکشن addnum() پر عمل درآمد کرنے کے لیے ہمیں اسے کال (طلب) کرنے کی ضرورت پڑے گی۔ پروگرام میں فنکشن کو اس کے نام کے بعد ( ) لکھ کر کال کیا جا سکتا ہے جیسا کہ پروگرام 7-3 کی آخری لائن میں دکھایا گیا ہے۔

نتیجہ:

```
Enter first number: 5
Enter second number: 6
The sum of 5 and 6 is 11
```

7.3.2 آرگیومینٹ اور پیرامیٹر (Argument and Parameters)

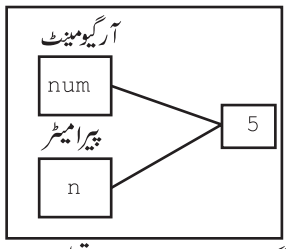
مذکورہ بالا مثال میں اعداد کو استعمال کنندہ سے فنکشن کے اندر ہی حاصل کیا گیا تھا، حالاں کہ استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کے معاملے میں یہ بھی ممکن ہے کہ قدروں (Values) کو اس وقت بھی حاصل کیا جا سکتا ہے جب فنکشن کو طلب کیا جا رہا ہو۔ فنکشن کو طلب کر کے دوران فنکشن کو پاس کی جانے والی وہ قدر (ویلیو) جسے فنکشن ہیڈر میں متعین کیے گئے نظیری پیرامیٹر میں حاصل کیا جاتا ہے آرگیومینٹ (Argument) کہلاتی ہے۔

پروگرام 7-4 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کو استعمال کرتے ہوئے ایک پروگرام لکھیے جو پہلے n فطری اعداد کا حاصل جمع ظاہر کرتا ہے جہاں n کو آرگیومینٹ کے طور پر پاس کیا گیا ہے۔

```
#Program 7-4
#Program to find the sum of first n natural numbers
#The requirements are:
  #1. n be passed as an argument
  #2. Calculate sum of first n natural numbers
  #3. Display the sum

#function header
def sumSquares(n):        #n is the parameter
     sum = 0
     for i in range(1,n+1):
          sum = sum + i
     print("The sum of first",n,"natural numbers is: ",sum)

 num = int(input("Enter the value for n: "))
 #num is an argument referring to the value input by the user
 sumSquares(num)          #function call
```

فرض کیجیے کہ پروگرام 7-4 پر عمل درآمد (Execution) کے دوران استعمال کنندہ عدد 5 داخل کرتا ہے۔ لہٰذا، متغیر num سے قدر 5 منسوب ہوجاتی ہے۔ اب اسے فنکشن میں آرگیومینٹ کے طور پر استعمال کیا جائے گا۔

```
sumSquares(num)
```



شکل 7.3: آرگیومینٹ اور پیرامیٹر دونوں کا تعلق ایک ہی قدر سے ہے

جیسے ہی فنکشن کو کال کیا جاتا ہے تو کنٹرول فنکشن پر عمل درآمد کے لیے منتقل ہوجاتا ہے۔

```
def sumSquares(n):
```

جہاں پیرامیٹر n کا تعلق بھی قدر 5 سے ہے جسے num سے منسوب کیا گیا ہے جیسا کہ شکل 7.3 میں دکھایا گیا ہے۔

چوں کہ num اور n دونوں سے ایک ہی قدر منسوب ہے لہٰذا دونوں کی شناخت ایک ہی ہے۔ ہم آرگیومینٹ اور پیرامیٹر سے منسوب آبجیکٹ کی شناخت کو معلوم کرنے کے لیے id() فنکشن کا استعمال کرسکتے ہیں۔ آیئے اسے مندرجہ ذیل مثال کی مدد سے سمجھنے کی کوشش کرتے ہیں۔

پروگرام 7-5 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کو استعمال کرتے ہوئے ایک پروگرام لکھیے جو ایک صحیح عدد کو حاصل کرتا ہے اور اس کی قدر میں 5 کا اضافہ کردیتا ہے۔ علاوہ ازیں آرگیومینٹ کی id (فنکشن کو طلب کرنے سے پہلے)، اضافہ کرنے سے پہلے اور بعد میں پیرامیٹر کی id کو ظاہر کرتا ہے۔

```
#Program 7-5
#Function to add 5 to a user input number
#The requirements are listed below:
  #1. Display the id()of argument before function call.
  #2. The function should have one parameter to accept the argument
  #3. Display the value and id() of the parameter.
  #4. Add 5 to the parameter
  #5. Display the new value and id()of the parameter to check
      #whether the parameter is assigned a new memory location or
      #not.
def incrValue(num):
    #id of Num before increment
    print("Parameter num has value:",num,"\nid =",id(num))
    num = num + 5
    #id of Num after increment
    print("num incremented by 5 is",num,"\nNow id is ",id(num))
number = int(input("Enter a number: "))
print("id of argument number is:",id(number))          #id of Number
incrValue(number)
```

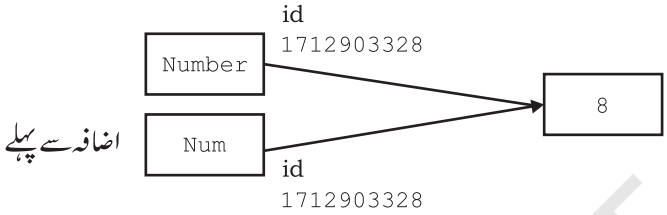
نتیجہ:

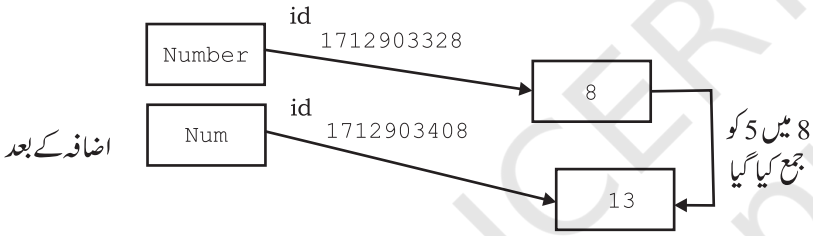
```
Enter a number: 8
id of argument number is: 1712903328
Parameter num has value: 8
id = 1712903328
num incremented by 5 is 13
Now id is 1712903408
```

Num اور Number دونوں کی id یکساں ہے

Num کی id تبدیل ہوسکتی ہے۔

آیئے مذکورہ بالا نتیجے کو ایک ڈائیگرام کی مدد سے سمجھتے ہیں (شکل 7.4 دیکھیے):





شکل 7.4: اضافہ کرنے سے پہلے اور بعد میں آرگیومنٹ اور پیرامیٹر کی ID

آرگیومینٹ اور پیرامیٹر دونوں کا ایک ہی نام ہوسکتا ہے جیسا کہ پروگرام 7-6 میں دکھایا گیا ہے۔

پروگرام 7-6 لسٹ میں ذخیرہ شدہ فلوٹنگ (ڈیٹا ٹائپ) قدروں کے اوسط کی تحسیب کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن myMean() کو استعمال کرتے ہوئے ایک پروگرام لکھیے۔

```
#Program 7-6
#Function to calculate mean
#The requirements are listed below:
  #1. The function should have 1 parameter (list containing floating
      #point values)
  #2. To calculate mean by adding all the numbers and dividing by
      #total number of elements

def myMean(myList): #function to compute means of values in list
     total = 0
     count = 0
     for i in myList:
```

```
            total = total + i            #Adds each element i to total
            count = count + 1            #Counts the number of elements
      mean = total/count                 #mean is calculated
      print("The calculated mean is:",mean)
myList = [1.3,2.4,3.5,6.9]
#Function call with list "myList" as an argument
myMean(myList)
```

نتیجہ:

```
The calculated mean is: 3.5250000000000004
```

پروگرام 7-7 آرگیومینٹ کے طور پر پاس کیے گئے عدد کے فیکٹوریل کی تحسیب اور اسے ظاہر کرنے کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن calcFact() کو استعمال کرتے ہوئے ایک پروگرام لکھیے۔

```
#Program 7-7
#Function to calculate factorial
#The requirements are listed below:
  #1. The function should accept one integer argument from user.
  #2. Calculate factorial. For example:
  #3. Display factorial

def calcFact(num):
      fact = 1
      for i in range(num,0,-1):
            fact = fact * i
      print("Factorial of",num,"is",fact)

num = int(input("Enter the number: "))
calcFact(num)
```

نتیجہ:

```
Enter the number: 5
Factorial of 5 is 120
```

نوٹ: چوں کہ ضرب کا عمل تقلیبی ہے لہٰذا 5! = 5*4*3*2*1 = 1*2*3*4*5

(A) اسٹرنگ بحیثیت پیرامیٹر (String as Parameters)

پروگرام 5-7 سے لے کر پروگرام 7-7 تک سبھی میں پاس کیے گئے آرگیومینٹ صرف عددی قسم کے ہیں۔ تاہم کچھ پروگراموں میں استعمال کنندہ کو آرگیومینٹ کے طور پر اسٹرنگ ویلیو کو پاس کرنے کی ضرورت پیش آسکتی ہے جیسا کہ پروگرام 8-7 میں دکھایا گیا ہے۔

پروگرام 7-8 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کو استعمال کرتے ہوئے ایک پروگرام لکھیے جو پہلا نام اور آخری نام بحیثیت آرگیومینٹ حاصل کرتا ہے، پورا نام حاصل کرنے کے لیے انھیں یکجا کرتا ہے اور نتیجے کو مندرجہ ذیل طریقے سے ظاہر کرتا ہے:

```
Hello full name
```

مثال کے طور پر اگر پہلا نام Gyan اور آخری نام Vardhan ہے تو نتیجہ مندرجہ ذیل ہونا چاہیے:

```
Hello Gyan Vardhan
```

```
#Program 7-8
#Function to display full name
#The requirements are listed below:
   #1. The function should have 2 parameters to accept first name and
       #last name.
   #2. Concatenate names using + operator with a space between first
       #name and last name.
   #3. Display full name.

   def fullname(first,last):
   #+ operator is used to concatenate strings
        fullname = first + " " + last
        print("Hello",fullname)
   #function ends here
   first = input("Enter first name: ")
   last = input("Enter last name: ")
   #function call
   fullname(first,last)
```

نتیجہ:

```
   Enter first name: Gyan
   Enter last name: Vardhan
   Hello Gyan Vardhan
```

(B) ڈیفالٹ پیرامیٹر (Default Parameter)

پائتھن میں یہ سہولت دست یاب ہے کہ پیرامیٹر کو ڈیفالٹ ویلیو تفویض کی جاسکتی ہے۔ ڈیفالٹ ویلیو وہ قدر ہے جو پہلے سے متعین ہوتی ہے اور اس وقت پیرامیٹر کو تفویض کر دی جاتی ہے جب فنکشن کال کا اپنا نظیری آرگیومینٹ نہیں ہوتا ہے۔

پروگرام 7-9 ایک پروگرام لکھیے جو کسری عدد کا شمار کنندہ اور نسب نما حاصل کرتا ہے اور بننے والی کسر واجب کسر نہیں ہے تو استعمال کنندہ کے ذریعے تعریف شدہ فنکشن

mixed Fraction()کو طلب (کال) کرتا ہے۔نسب نما کی ڈیفالٹ قدر 1 ہے۔فنکشن صرف اسی صورت میں مخلوط کسر کو ظاہر کرتا ہے جب پیرامیٹر کے ذریعے تشکیل کردہ کسر کی قدر مکمل عدد نہیں ہوتی ہے۔

```
#Program 7-9
#Function to display mixed fraction for an improper fraction
#The requirements are listed below:
  #1. Input numerator and denominator from the user.
  #2. Check if the entered numerator and denominator form a proper
      #fraction.
  #3. If they do not form a proper fraction, then call
      #mixedFraction().
  #4. mixedFraction()display a mixed fraction only when the fraction
      #does not evaluate to a whole number.

def mixedFraction(num,deno = 1):
     remainder = num % deno
#check if the fraction does not evaluate to a whole number
     if remainder!= 0:
          quotient = int(num/deno)
          print("The mixed fraction=", quotient,"(",remainder, "/",
deno,")")
     else:

          print("The given fraction evaluates to a whole number")
#function ends here
num = int(input("Enter the numerator: "))
deno = int(input("Enter the denominator: "))
print("You entered:",num,"/",deno)
if num > deno:    #condition to check whether the fraction is
improper
     mixedFraction(num,deno)       #function call
else:
     print("It is a proper fraction")
```

نتیجہ:

```
Enter the numerator: 17
Enter the denominator: 2
You entered: 17 / 2
The mixed fraction =  8 ( 1 / 2 )
```

مذکورہ بالا پروگرام میں، داخل کیا گیا نسب نما 2 ہے جو پیرامیٹر "deno" کو پاس کر دیا جاتا ہے لہٰذا آرگیومینٹ deno کی ڈیفالٹ ویلیو دوبارہ تحریر کر دی جاتی ہے۔

آیئے مندرجہ ذیل فنکشن کال پر غور کیجیے:

```
mixedFraction(9)
```

یہاں num کو قدر 9 تفویض کی جاتی ہے اور deno ڈیفالٹ ویلیو 1 کا استعمال کرے گا۔

نوٹ:

- ایک فنکشن آرگیومینٹ عبارت کی شکل میں بھی ہوسکتا ہے مثلاً

```
mixedFraction(num+5,deno+5)
```

اس معاملے میں فنکشن کو طلب کرنے سے پہلے آرگیومینٹ کی تحسیب کی جاتی ہے تاکہ پیرامیٹر کو درست قدر تفویض کی جاسکے۔

- پیرامیٹر اسی ترتیب میں ہونے چاہئیں جس ترتیب میں آرگیومینٹ ہوتے ہیں۔

ڈیفالٹ پیرامیٹر کو فنکشن ہیڈر میں تعاقبی پیرامیٹر (Trailing parameter) ہونا چاہیے جس کا مطلب ہے کہ اگر کسی پیرامیٹر کی کوئی ڈیفالٹ ویلیو ہے تو اس کے دائیں طرف والے دیگر سبھی پیرامیٹر بھی ڈیفالٹ قدروں کے حامل ہونے چاہئیں۔ مثال کے طور پر،

```
def mixedFraction(num,deno = 1)
def mixedFraction(num = 2,deno = 1)
```

آیئے کچھ اور فنکشن ڈیفینیشن ہیڈر پر غور کریں:

```
#incorrect as default must be the last
#parameter
def calcInterest(principal = 1000, rate,
time = 5):
#correct
def calcInterest(rate,principal = 1000,
time = 5):
```

7.3.3 فنکشن کے ذریعے ظاہر کی جانے والی قدر

فنکشن کو جب طلب کیا جاتا ہے تو وہ قدر کو ظاہر کر بھی سکتا ہے اور نہیں بھی۔ ریٹرن (Return) بیان فنکشن کی قدر کو ظاہر کر دیتا ہے۔ اب تک دی گئی مثالوں میں فنکشن تحسیبات کا عمل انجام دیتا ہے اور نتیجہ (یا نتائج) کو ظاہر کر دیتا ہے۔ یہ فنکشن کسی قدر کو ظاہر نہیں کرتے ہیں۔ اس قسم کے فنکشن کالعدم (Void) فنکشن کہلاتے ہیں۔ لیکن ایک ایسی صورت حال بھی پیدا ہوسکتی ہے جس میں ہمیں قدر (یا قدروں) کو فنکشن سے اس کے کالنگ فنکشن میں بھیجنے کی ضرورت پیش آسکتی ہے۔ اس کام کو رٹرن بیان کی مدد سے انجام دیا جاتا ہے۔

رٹرن بیان مندرجہ ذیل کاموں کو انجام دیتا ہے:

- کنٹرول، کالنگ فنکشن کو منتقل کر دیتا ہے۔
- قدر (یا قدروں) یا None کو ظاہر کرتا ہے۔

پروگرام 7-10 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن CalcPow() کا استعمال کرکے ایک پروگرام لکھیے جو اساس اور قوت نما کو آرگیومینٹ کے طور پر حاصل کرتا ہے اور قدر Baseexponent کو ظاہر کردیتا ہے جہاں اساس (Base) اور قوت نما (Exponent) صحیح اعداد ہیں۔

```
#Program 7-10
#Function to calculate and display base raised to the power exponent
#The requirements are listed below:
   #1. Base and exponent are to be accepted as arguments.
   #2. Calculate Base^exponent
   #3. Return the result (use return statement )
   #4. Display the returned value.
def calcpow(number,power):          #function definition
     result = 1
     for i in range(1,power+1):
          result = result * number
     return result

base = int(input("Enter the value for the Base: "))
expo = int(input("Enter the value for the Exponent: "))
answer = calcpow(base,expo)          #function call
print(base,"raised to the power",expo,"is",answer)
```

نتیجہ:

```
Enter the value for the Base: 5
Enter the value for the Exponent: 4
5 raised to the power 4 is 625
```

اب تک ہم نے یہ سیکھا ہے کہ ایک فنکشن میں پیرامیٹر ہو بھی سکتے ہیں اور نہیں بھی۔ اسی طرح ایک فنکشن قدر (یا قدروں) کو ظاہر کر بھی سکتا ہے اور نہیں بھی۔ پائتھن میں ہم اپنی ضروریات کے لحاظ سے مندرجہ ذیل کسی ایک طریقے سے فنکشن کو استعمال کرتے ہیں۔

- ایسا فنکشن جس میں کوئی آرگیومینٹ نہیں ہوتا اور یہ کوئی قدر ظاہر نہیں کرتا ہے
- ایسا فنکشن جس میں کوئی آرگیومینٹ نہیں ہوتا اور یہ کسی قدر کو ظاہر کرتا ہے
- ایسا فنکشن جس میں آرگیومینٹ ہوتے ہیں اور یہ کوئی قدر ظاہر نہیں کرتا ہے
- ایسا فنکشن جس میں آرگیومینٹ ہوتے ہیں اور یہ کسی قدر کو ظاہر کرتا ہے

7.3.4 عمل درآمد کا تسلسل (Flow of Execution)

عمل درآمد کے تسلسل کی تعریف اس ترتیب کے طور پر کی جاسکتی ہے جس کے تحت پروگرام کے بیانات ہدایات کی تعمیل کرتے ہیں۔ پائتھن انٹرپریٹر، پروگرام میں موجود ہدایات پر عمل درآمد پہلے بیان سے شروع کرتا ہے۔ بیانات پر عمل درآمد اوپر سے نیچے کی طرف ان کے ظاہر ہونے کی ترتیب میں یکے بعد دیگرے ہوتا ہے۔

جب انٹرپریٹر کسی فنکشن ڈیفنیشن پر پہنچتا ہے تو فنکشن کے اندر موجود بیانات پر اس وقت تک عمل درآمد نہیں کیا جاتا ہے جب تک کہ فنکشن کو طلب (کال) نہ کیا جائے۔ بعد میں جب انٹرپریٹر فنکشن کال پر پہنچتا ہے تو عمل درآمد کے تسلسل میں معمولی سا انحراف دیکھنے کو ملتا ہے۔ اس معاملے میں اگلے بیان پر جانے کے بجائے کنٹرول فنکشن طلب کیے گئے فنکشن پر پہنچ جاتا ہے اور اس فنکشن کے بیان پر عمل درآمد کیا جاتا ہے۔ اس کے بعد کنٹرول فنکشن کال پر واپس آتا ہے تاکہ پروگرام کے باقی بیانات پر عمل درآمد کیا جاسکے۔ چنانچہ جب ہم پروگرام کو پڑھتے ہیں تو ہمیں اسے فقط اوپر سے نیچے کی طرف نہیں پڑھنا چاہیے بلکہ ہمیں کنٹرول یا عمل درآمد کے تسلسل کا اتباع کرنا چاہیے۔ یہ بات بھی قابل غور ہے کہ پروگرام میں فنکشن کو اسے طلب (کال) کرنے سے پہلے متعین کیا جانا ضروری ہے۔

پروگرام 7-11 فنکشن کے استعمال سے عمل درآمد کے تسلسل کو سمجھنے کے لیے پروگرام

```
#Program 7-11
#print using functions
helloPython()                                  #Function Call

def helloPython():                             #Function definition
     print("I love Programming")
```

مذکورہ بالا کوڈ پر عمل درآمد کے نتیجے میں مندرجہ ذیل غلطی ظاہر ہوتی ہے:

```
Traceback (most recent call last):
  File "C:\NCERT\Prog 7-11.py", line 3, in <module>
    helloPython()                              #Function Call
NameError: name 'helloPython' is not defined
```

غلطی 'فنکشن متعین نہیں ہے (Function is not defined)' ظاہر ہوجاتی ہے حالاں کہ فنکشن تعریف شدہ ہے۔ جب کنٹرول کا ایک فنکشن کال سے سامنا ہوتا ہے تو یہ فنکشن ڈیفنیشن پر پہنچ جاتا ہے اور اسے ایگزیکیوٹ کرتا ہے۔ مذکورہ بالا پروگرام میں چوں کہ فنکشن کال، فنکشن ڈیفنیشن سے پہلے ہے لہٰذا انٹرپریٹر جب فنکشن ڈیفنیشن کو غیر موجود پاتا ہے تو غلطی ظاہر ہوجاتی ہے۔

اسی لیے فنکشن کو فنکشن کال سے پہلے متعین کیا جانا چاہیے جیسا کہ ذیل میں دکھایا گیا ہے:

```
def helloPython(): #Function definition
     print("I love Programming")

helloPython()      #Function Call
```

```
[2]     def Greetings(Name):          #Function Header
[3]          print("Hello "+Name)

[1]     Greetings("John")             #Function Call
[4]     print("Thanks")
```

```
[4]     def RectangleArea(l,b):       #Function Header
[5]     return l*b

[1]     l = input("Length: ")
[2]     b = input("Breadth: ")
[3][6]  Area = RectangleArea(l,b)     #Function Call
[7]     print(Area)
[8]     print("thanks")
```

شکل 7.5: بیانات کے عمل درآمد کی ترتیب

شکل 7.5 میں دو پروگراموں کے لیے عمل درآمد کے تسلسل کی وضاحت کی گئی ہے۔ اسکوائر بریکٹ میں درج اعداد بیانات کے ایگزیکیوٹ ہونے کی ترتیب کو ظاہر کرتے ہیں۔

بعض اوقات کسی فنکشن کو کئی قدریں ظاہر کرنی پڑسکتی ہیں جنھیں ٹپل کا استعمال کرکے ظاہر کیا جاسکتا ہے۔ پروگرام 7-12 میں ایک فنکشن کو دکھایا گیا ہے جو ٹپل کا استعمال کرکے دو قدروں یعنی مستطیل کے رقبے اور احاطے کو ظاہر کرتا ہے۔

پروگرام 7-12 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کرتے ہوئے ایک پروگرام لکھیے جو مستطیل کی لمبائی اور چوڑائی کو حاصل کرتا ہے اور مستطیل کے رقبے اور احاطے کو ظاہر کرتا ہے۔

```
#Program 7-12
#Function to calculate area and perimeter of a rectangle
#The requirements are listed below:
  #1. The function should accept 2 parameters.
  #2. Calculate area and perimeter.
  #3. Return area and perimeter.

def calcAreaPeri(Length,Breadth):
     area = length * breadth
     perimeter = 2 * (length + breadth)
     #a tuple is returned consisting of 2 values area and perimeter
     return (area,perimeter)

l = float(input("Enter length of the rectangle: "))
b = float(input("Enter breadth of the rectangle: "))
#value of tuples assigned in order they are returned
area,perimeter = calcAreaPeri(l,b)
print("Area is:",area,"\nPerimeter is:",perimeter)
```

نتیجہ :

```
Enter Length of the rectangle: 45
Enter Breadth of the rectangle: 66
Area is: 2970.0
Perimeter is: 222.0
```

پائتھن میں متعدد قدروں کو ٹپل کے ذریعے ظاہر کیا جاتا ہے۔(باب 10)

پروگرام 7-13 ایک پروگرام لکھیے جو ٹریفک لائٹ کی نقل کرتا ہے۔ پروگرام میں مندرجہ ذیل باتیں شامل ہونی چاہئیں۔

1۔ استعمال کنندہ کے ذریعے تعریف شدہ فنکشن trafficLight() جو استعمال کنندہ سے ان پُٹ حاصل کرتا ہے اور اگر استعمال کنندہ RED، YELLOW اور GREEN کے علاوہ کچھ اور داخل کرتا ہے تو یہ غلطی صادر ہوجانے کا پیغام ظاہر کردیتا ہے۔ فنکشن Light() کو طلب کیا جاتا ہے اور Light() کے ذریعے ظاہر کی جانے والی قدر کی بنیاد پر مندرجہ ذیل کو ظاہر کرتا ہے۔

a) ''رکیے۔ آپ کی زندگی بیش قیمت ہے'' اگر Light() کے ذریعے ظاہر کی جانے والی قدر 0 ہے۔

b) ''براہ مہربانی لائٹ کے ہرا ہونے تک انتظار کیجیے'' اگر Light() کے ذریعے ظاہر کی جانے والی قدر 1 ہے۔

c) ''چلیے! صبر کرنے کے لیے شکریہ'' اگر Light() کے ذریعے ظاہر کی جانے والی قدر 2 ہے۔

2۔ استعمال کنندہ کے ذریعے تعریف شدہ فنکشن Light( ) جو ان پُٹ کے طور پر اسٹرنگ حاصل کرتا ہے اور اگر ان پُٹ RED ہے تو 0 ظاہر کرتا ہے، اگر ان پُٹ YELLOW ہے تو 1 ظاہر کرتا ہے اور اگر ان پُٹ GREEN ہے تو 2 ظاہر کرتا ہے۔ ان پُٹ کو آرگیومینٹ کے طور پر پاس کیا جانا چاہیے۔

3۔ فنکشن trafficLight( ) پر عمل درآمد ہونے کے بعد ''SPEED THRILLS BUT KILLS'' ڈسپلے کیجیے۔

```
#Program 7-13
#Function to simulate a traffic light
#It is required to make 2 user defined functions trafficLight() and
#light().

def trafficLight():
     signal = input("Enter the colour of the traffic light: ")
     if (signal not in ("RED","YELLOW","GREEN")):
          print("Please enter a valid Traffic Light colour in
CAPITALS")
     else:
          value = light(signal)           #function call to light()
```

```
if (value == 0):
     print("STOP, Your Life is Precious.")
elif (value == 1):
     print ("PLEASE GO SLOW.")
else:
               print("GO!,Thank you for being patient.")
#function ends here

def light(colour):
     if (colour == "RED"):
          return(0);
     elif (colour == "YELLOW"):
          return (1)
     else:
          return(2)
#function ends here

trafficLight()
print("SPEED THRILLS BUT KILLS")
```

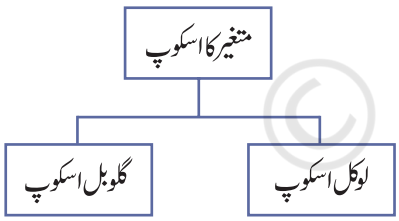
نتیجہ:

```
Enter the colour of the traffic light: YELLOW
PLEASE GO SLOW.
SPEED THRILLS BUT KILLS
```

7.4 متغیر کا اسکوپ (SCOPE OF A VARIABLE)

ایک فنکشن کے اندر تعریف شدہ متغیر کو اس کے باہر سے ایکسس نہیں کیا جا سکتا ہے۔ ہر ایک متغیر کا ایکسس واضح طور پر متعین ہوتا ہے۔ پروگرام کا وہ حصہ جہاں کسی متغیر کو ایکسس کیا جا سکتا ہے اس متغیر کا اسکوپ کہلاتا ہے۔ متغیر کا اسکوپ مندرجہ ذیل میں سے کوئی ایک ہو سکتا ہے:

وہ متغیر جو گلوبل اسکوپ کا حامل ہوتا ہے اسے گلوبل متغیر اور لوکل اسکوپ کے حامل متغیر کو لوکل متغیر کہتے ہیں۔



شکل 7.6: متغیر کا اسکوپ

(A) گلوبل متغیر (Global Variable)

پائتھن میں، وہ متغیر جسے کسی فنکشن یا کسی بلاک کے باہر متعین کیا جاتا ہے اسے گلوبل متغیر کہا جاتا ہے۔ اس متغیر کو آگے متعین کیے گئے کسی بھی فنکشن سے ایکسس کیا جا سکتا ہے۔ گلوبل متغیر میں کی گئی کسی بھی قسم کی تبدیلی پروگرام میں موجود ان سبھی فنکشن کو متاثر کرے گی جہاں اس متغیر کو ایکسس کیا جا سکتا ہے۔

(B) لوکل متغیر (Local Variable)

وہ متغیر جسے کسی فنکشن یا کسی بلاک کے اندر متعین کیا جاتا ہے اسے لوکل متغیر کہا جاتا ہے۔ اس متغیر کو صرف اسی فنکشن یا بلاک میں ایکسس کیا جا سکتا ہے جہاں اسے متعین کیا گیا ہے۔ یہ صرف فنکشن کی تعمیل ہونے تک موجود رہتا ہے۔

پروگرام 7-14 فنکشن کے باہر موجود کسی متغیر کو ایکسس کرنے کے لیے پروگرام

```
#Program 7-14
#To access any variable outside the function
num = 5
def myFunc1( ):
    y = num + 5
    print("Accessing num -> (global) in myFunc1, value = ",num)
    print("Accessing y-> (local variable of myFunc1) accessible, value=",y)

myFunc1()
print("Accessing num outside myFunc1 ",num)
print("Accessing y outside myFunc1 ",y)
```

نتیجہ:

```
Accessing num -> (global) in myFunc1, value =  5
Accessing y-> (local variable of myFunc1) accessible, value =  10
Accessing num outside myFunc1  5
Traceback (most recent call last):
  File "C:\NCERT\Prog 7-14.py", line 9, in <module>
    print("Accessing y outside myFunc1 ",y)
NameError: name 'y' is not defined
```

y کو اگر myfunc1() کے باہر سے ایکسس کیا جاتا ہے تو یہ غلطی ظاہر کرتا ہے

گلوبل متغیر کا نتیجہ ← لوکل متغیر کا نتیجہ

نوٹ:

- گلوبل متغیر میں کی جانے والی تبدیلی مستقل نوعیت کی ہوتی ہے اور یہ ان سبھی فنکشن کو متاثر کرتی ہے جہاں اسے استعمال کیا جاتا ہے۔
- اگر کسی متغیر کو فنکشن کے باہر اسی نام سے متعین کیا جاتا ہے جو گلوبل متغیر کا ہے تو اسے اس فنکشن کے لیے لوکل تصور کیا جاتا ہے اور گلوبل متغیر مخفی ہو جاتا ہے۔
- اگر گلوبل متغیر کی ترمیم شدہ قدر کو فنکشن کے باہر استعمال کیا جانا ہو تو فنکشن میں کلیدی لفظ global کو متغیر کے نام کے ساتھ سابقہ کے طور پر استعمال کیا جانا چاہیے۔

پروگرام 7-15 فنکشن کے باہر کسی متغیر کو ایکسس کرنے کے لیے پروگرام لکھیے۔

```
#Program 7-15
#To access any variable outside the function
num = 5
def myfunc1():
    #Prefixing global informs Python to use the updated global
```

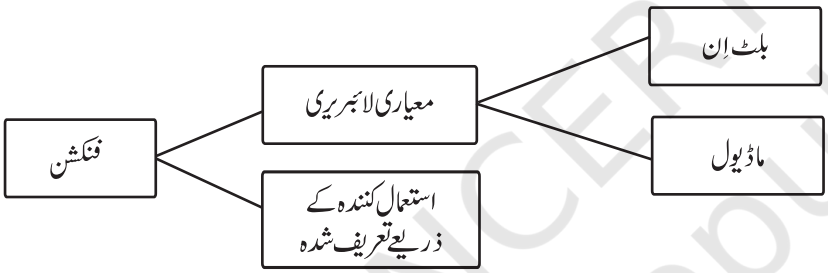
```
    #variable num outside the function
    global num
    print("Accessing num =",num)
    num = 10
    print("num reassigned =",num)
#function ends here

myfunc1()
print("Accessing num outside myfunc1",num)
```

نتیجہ:

```
Accessing num = 5 ⟵ گلوبل متغیر num ایکسس کیا گیا ہے کیوں کہ ابہام کو اس میں گلوبل سابقہ لگا کر حل کر لیا گیا ہے
num reassigned = 10
Accessing num outside myfunc1 10
```

7.5 پائتھن معیاری لائبریری (Python Standard Library)

پائتھن میں ایک انتہائی وسیع معیاری لائبریری موجود ہے۔ یہ ایسے متعدد بلٹ ان فنکشن کا مجموعہ ہے جنھیں پروگرام میں حسب ضرورت طلب کیا جاسکتا ہے چنانچہ عام طور سے استعمال ہونے والے ان فنکشن کی بار بار تشکیل میں صرف ہونے والے وقت کی بچت ہوتی ہے۔



شکل 7.7: فنکشن کی اقسام

7.5.1 بلٹ ان فنکشن (Built-in functions)

پائتھن میں بلٹ ان فنکشن پہلے سے تیار شدہ ایسے فنکشن ہیں جنھیں پروگرام میں بار بار استعمال کیا جاتا ہے۔
آیئے مندرجہ ذیل پائتھن پروگرام کا معائنہ کریں:

```
#Program to calculate square of a number
a = int(input("Enter a number: ")
b = a * a
print(" The square of ",a ,"is", b)
```

مندرجہ بالا پروگرام میں input()، int() اور print() بلٹ ان فنکشن ہیں۔ مذکورہ بلٹ ان فنکشن کے لیے ہدایات کے جس سیٹ پر عمل درآمد کیا جاتا ہے وہ پائتھن انٹرپریٹر میں پہلے ہی سے متعین ہوتا ہے۔
پائتھن پروگرام کے مندرجہ ذیل بیان پر غور کیجیے جس میں ایک بلٹ ان فنکشن کے لیے فنکشن کال دیا ہوا ہے۔ اس بیان سے متعلق سوالوں کے جواب دیجیے:

```
fname = input("Enter your name: ")
```

زیر استعمال فنکشن کا کیا نام ہے؟

- `input()`

کیا مذکورہ فنکشن کسی آرگیومینٹ یا قدر کو حاصل کرتا ہے؟

- ہاں، کیوں کہ قوسین "()" میں اسٹرنگ "Enter your name" موجود ہے۔

کیا یہ فنکشن ویلیو کو ظاہر کرتا ہے؟

- ہاں، کیوں کہ فنکشن کے نام سے پہلے اسائنمینٹ (=) آپریٹر موجود ہے، اس کا مطلب ہے کہ یہ فنکشن fname متغیر میں اسٹور کی گئی قدر کو ظاہر کرے گا۔

چنانچہ، فنکشن input() قدر کو حاصل کرتا ہے اور اسے ظاہر کرتا ہے۔

اب بلٹ ان فنکشن int() اور print() پر غور کیجیے اور مندرجہ ذیل سوالوں کے جواب دیجیے:

- کیا یہ فنکشن قدر یا آرگیومینٹ کو حاصل کرتا ہے؟
- کیا یہ فنکشن ویلیو کو ظاہر کرتا ہے؟

پائتھن میں بار بار استعمال ہونے والے کچھ بلٹ ان فنکشن کی زمرہ وار فہرست ذیل میں دی گئی ہے:

بلٹ ان فنکشن

ان پُٹ یا آؤٹ پُٹ	ڈیٹا ٹائپ کی تبدیلی	ریاضیاتی فنکشن	دیگر فنکشن
input()	bool()	abs()	__import__()
print()	chr()	divmod()	len()
	dict()	max()	range()
	float()	min()	type()
	int()	pow()	
	list()	sum()	
	ord()		
	set()		
	str()		
	tuple()		

ہم ان میں سے کچھ بلٹ ان فنکشن کو پہلے ہی استعمال کر چکے ہیں۔

آیئے ان میں سے کچھ فنکشن کا مطالعہ کرتے ہیں جنھیں جدول 7.1 میں دیا گیا ہے۔

جدول 7.1 عام طور سے استعمال ہونے والے بلٹ ان فنکشن

فنکشن سنٹیکس	آرگیومینٹ	ظاہر ہونے والی قدر	آؤٹ پٹ کی مثال

abs(x)	x صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہوسکتا ہے	x کی مطلق قدر	`>>> abs(4)`
			`4`
			`>>> abs(-5.7)`
			`5.7`

divmod(x,y)	x اور y صحیح اعداد ہیں	ٹپل: (خارج قسمت، باقی)	>>> divmod(7,2)
			(3, 1)
			>>> divmod(7.5,2)
			(3.0, 1.5)
			>>> divmod(-7,2)
			(-4, 1)

max(sequence)
or
max(x,y,z,...)
	x، y، z،... صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہو سکتے ہیں	سلسلے کا سب سے بڑا عدد / دو یا دو سے زیادہ آرگیومینٹ میں سب سے بڑا	>>> max([1,2,3,4])
			4
			>>> max("Sincerity")
			'y' #Based on ASCII value
			>>> max(23,4,56)
			56

min(sequence)
or
min(x,y,z,...)
	x، y، z،... صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہو سکتے ہیں	سلسلے کا سب سے چھوٹا عدد / دو یا دو سے زیادہ آرگیومینٹ میں سب سے چھوٹا	>>> min([1,2,3,4])
			1
			>>> min("Sincerity")
			'S'
			#Uppercase letters have lower ASCII values than lowercase letters.
			>>> min(23,4,56)
			4

pow(x,y[,z])	x، y، z،... صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہو سکتے ہیں	x^y (x کی قوت y) اگر z دیا ہوا ہے تو: (xy) % z	>>> pow(5,2)
			25.0
			>>> pow(5.3,2.2)
			39.2
			>>> pow(5,2,4)
			1

sum(x[,num])	x ایک عددی سلسلہ ہے اور num ایک اختیاری آرگیومینٹ ہے	سلسلے میں بائیں سے دائیں طرف سبھی عناصر کا حاصل جمع اگر دیا ہوا پیرامیٹر num کو حاصل جمع میں جوڑا جاتا ہے	>>> sum([2,4,7,3])
			16
			>>> sum([2,4,7,3],3)
			19
			>>> sum((52,8,4,2))
			66

len(x)	x ایک سلسلہ یا ڈکشنری ہو سکتا ہے	x میں موجود عناصر کو شمار کرتا ہے	>>> len("Patience")
			8
			>>> len([12,34,98])
			3
			>>> len((9,45))
			2 >>>len({1:"Anuj",
			2:"Razia",
			3:"Gurpreet",
			4:"Sandra"})
			4

7.5.2 ماڈیول (Module)

بلٹ ان فنکشن کے علاوہ، پائتھن اسٹینڈرڈ لائبریری میں بہت سے ماڈیول بھی شامل ہیں۔ایک طرف جہاں فنکشن، ہدایات کا مجموعہ ہے وہیں دوسری طرف ماڈیول فنکشن کا مجموعہ ہے۔جیسا کہ ہم جانتے ہیں کہ جب پروگرام بہت بڑا ہو جاتا ہے تو کوڈ کو آسان بنانے اور تکرار سے بچنے کے لیے فنکشن کا استعمال کیا جاتا ہے۔ پیچیدہ مسئلوں کے معاملے میں کوڈ کو ایک ہی فائل میں منظّم کرنا آسان نہیں ہوتا ہے۔لہٰذا اس صورت میں پروگرام کو مختلف سطحوں کے تحت مختلف حصوں میں تقسیم کر دیا جاتا ہے، جنھیں ماڈیول کہتے ہیں۔فرض کیجیے کہ ہم نے ایک پروگرام میں کچھ فنکشن تشکیل دیے ہیں اور ہم انھیں کسی دوسرے پروگرام میں استعمال کرنا چاہتے ہیں۔اس صورت میں ہم ان فنکشن کو ماڈیول کے تحت محفوظ کر سکتے ہیں اور انھیں دوبارہ استعمال کر سکتے ہیں۔ماڈیول کی تشکیل پائتھن (.py) فائل کے طور پر ہوتی ہے جو فنکشن کی تعریفات کے مجموعہ پر مشتمل ہوتی ہے۔

ماڈیول کو استعمال کرنے کے لیے ہمیں ماڈیول کو امپورٹ (Import) کرنا ہوگا، ہم اس ماڈیول سبھی فنکشن کو براہ راست استعمال کرتے ہیں۔ماڈیول کو امپورٹ کرنے کے لیے استعمال کیا جانے والا بیان درج ذیل ہے:

```
import modulename1 [,modulename2, …]
```

اس بیان کی مدد سے ہم ماڈیول کے سبھی فنکشن کو ایکسس کر سکتے ہیں۔ماڈیول کے فنکشن کو طلب کرنے کے لیے فنکشن کے نام سے پہلے ماڈیول کا نام لکھا جاتا ہے اور یہ دونوں ایک دوسرے سے ایک نقطہ (.) کے ذریعے علاحدہ رہتے ہیں۔
سنٹیکس کو ذیل میں دکھایا گیا ہے:

```
modulename.functionname()
```

(A) بلٹ ان ماڈیول (Built-in Modules)

پائتھن لائبریری میں متعدد بلٹ ان ماڈیول ہیں جو پروگرامر کے لیے انتہائی مفید ہیں۔آیئے عام طور سے استعمال ہونے والے کچھ ماڈیول اور بار بار استعمال ہونے والے ایسے فنکشن پر غور کریں جو ان ماڈیول میں پائے جاتے ہیں:

- میتھ (Math)
- رینڈم (Random)
- اسٹیٹسٹکس (Statistics)

یاد رہے کہ پائتھن میں انگریزی کے چھوٹے اور بڑے حروف کا مفہوم مختلف ہے۔سبھی ماڈیول کے نام چھوٹے حروف میں ہیں۔

1۔ ماڈیول کا نام: میتھ (math)

یہ مختلف اقسام کے ریاضیاتی فنکشن پر مشتمل ہوتا ہے۔اس ماڈیول میں موجود اکثر فنکشن فلوٹ ویلیو کو ظاہر کرتے ہیں۔math ماڈیول میں عام طور سے استعمال ہونے والے کچھ فنکشن جدول 7.2 میں دیے گئے ہیں۔میتھ ماڈیول کو استعمال کرنے کے لیے ہمیں مندرجہ ذیل بیان کو استعمال کرتے ہوئے اسے امپورٹ کرنا ہوگا۔

```
import math
```

جدول 7.2: math ماڈیول میں عام طور سے استعمال ہونے والے کچھ فنکشن

فنکشن سنٹیکس	آرگیومینٹ	ظاہر ہونے والی قدر	آؤٹ پٹ کی مثال

math.ceil(x)	x صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہوسکتا ہے	x کی زیادہ سے زیادہ قدر	>>> math.ceil(-9.7)
			-9
			>>> math.ceil (9.7)
			10
			>>> math.ceil(9)
			9

math.floor(x)	x صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہوسکتا ہے	x کی کم سے کم قدر	>>> math.floor(-4.5)
			-5
			>>> math.floor(4.5)
			4
			>>> math.floor(4)
			4

math.fabs(x)	x صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہوسکتا ہے	x کی مطلق قدر	>>> math.fabs(6.7)
			6.7
			>>> math.fabs(-6.7)
			6.7
			>>> math.fabs(-4)
			4.0

math.factorial(x)	x is a positive integer	factorial of x	>>> math.factorial(5) 120

math.fmod(x,y)	x اور y صحیح اعداد یا فلوٹنگ پوائنٹ نمبر ہوسکتے ہیں	x کے نشان کے ساتھ y % (x)	>>> math.fmod(4,4.9)
			4.0
			>>> math.fmod(4.9,4.9)
			0.0
			>>> math.fmod(-4.9,2.5)
			-2.4
			>>> math.fmod(4.9,-4.9)
			0.0

math.gcd(x,y)	x اور y مثبت صحیح اعداد ہیں	x اور y کا سب سے بڑا مشترک قاسم (gcd)	>>> math.gcd(10,2) 2

math.pow(x,y)	x اور y صحیح اعداد یا فلوٹنگ پوائنٹ نمبر ہو سکتے ہیں	x^y (x کی قوت y)	>>> math.pow(3,2)
			9.0
			>>> math.pow(4,2.5)
			32.0
			>>> math.pow(6.5,2)
			42.25
			>>> math.pow(5.5,3.2)
			233.97

math.sqrt(x)	x صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہوسکتا ہے	x کا جذر المربع	>>> math.sqrt(144)
			12.0
			>>> math.sqrt(.64)
			0.8

math.sin(x)	x صحیح عدد یا فلوٹنگ پوائنٹ نمبر ہوسکتا ہے	x کی sine قدر (ریڈین میں)	>>> math.sin(0)
			0
			>>> math.sin(6)
			-0.279

2۔ ماڈیول کا نام: رینڈم (Random)

یہ ماڈیول ایسے فنکشن پر مشتمل ہے جن کا استعمال بے ترتیب اعداد (Random number) کی تشکیل کے لیے کیا جاتا ہے۔ Random ماڈیول میں عام طور سے استعمال ہونے والے کچھ فنکشن جدول 7.3 میں دیے گئے ہیں۔ اس ماڈیول کو استعمال کرنے کے لیے ہم مندرجہ ذیل بیان کو استعمال کرتے ہوئے اسے امپورٹ کر سکتے ہیں۔

```
import random
```

جدول 7.3 رینڈم ماڈیول میں عام طور سے استعمال ہونے والے کچھ فنکشن

فنکشن سنٹیکس	آرگیومینٹ	ظاہر ہونے والی قدر	آؤٹ پٹ کی مثال
random.random()	کوئی آرگیومنٹ نہیں (خالی)	0.0 تا 1.0 کی رینج میں رینڈم حقیقی اعداد (فلوٹنگ) کو ظاہر کرتا ہے	>>> random.random() 0.65333522

random.randint(x,y)	x اور y صحیح اعداد ہیں اس طرح کہ x <= y	x اور y کے درمیان رینڈم صحیح اعداد	>>> random.randint(3,7)
			4
			>>> random.randint(-3,5)
			1
			>>> random.randint(-5,-3)
			-5.0

random.randrange(y)	اختتامی قدر کا اظہار کرتے ہوئے y مثبت صحیح عدد ہے	0 اور y کے درمیان رینڈم صحیح اعداد	>>> random.randrange(5) 4
random.randrange(x,y)	ابتدائی اور اختتامی قدر کا اظہار کرتے ہوئے x اور y مثبت صحیح اعداد ہیں	0 اور y کے درمیان رینڈم صحیح اعداد	>>> random.randrange(2,7) 2

3۔ ماڈیول کا نام: اسٹیٹسٹکس (Statistics)

یہ ماڈیول عددی (حقیقی قدر والے) اعداد و شمار سے متعلق شماریات کی تحسیب کے لیے فنکشن مہیا کرتا ہے۔ statistics ماڈیول میں عام طور سے استعمال ہونے والے کچھ فنکشن جدول 7.4 میں دیے گئے ہیں۔ اس

ماڈیول کواستعمال کرنے کے لیے،ہم مندرجہ ذیل بیان کواستعمال کرتے ہوئے اسے امپورٹ کر سکتے ہیں۔

```
import statistics
```

جدول 7.3 Statistics ماڈیول میں عام طور سے استعمال ہونے والے کچھ فنکشن

فنکشن سنٹیکس	آرگیومینٹ	ظاہر ہونے والی قدر	آؤٹ پٹ کی مثال
statistics.mean(x)	x ایک عددی سلسلہ ہے	حسابی درمیانہ (اوسط)	>>> statistics.mean([11,24,32,45,51]) 32.6
statistics.median(x)	x ایک عددی سلسلہ ہے	x کا وسطانیہ (وسطی قدر)	>>>statistics.median([11,24,32,45,51]) 32
statistics.mode(x)	x ایک سلسلہ ہے	بہتاتیہ (سب سے زیادہ مرتبہ دہرائی جانے والی قدر)	>>> statistics.mode([11,24,11,45,11]) 11 >>> statistics.mode(("red","blue","red")) 'red'

نوٹ:

- import بیان کو پروگرام میں کسی بھی جگہ پر لکھا جا سکتا ہے۔
- ماڈیول کو صرف ایک مرتبہ امپورٹ کیا جانا چاہیے۔
- پائتھن میں موجود ماڈیول کی فہرست حاصل کرنے کے لیے، ہم مندرجہ ذیل بیان کو استعمال کر سکتے ہیں:

```
>>> help("module")
```

- ماڈیول کے مواد (مثلاً math) کو دیکھنے کے لیے مندرجہ ذیل کو ٹائپ کیجیے:

```
>>> help("math")
```

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> help("math")
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.
```

شکل 7.8: ماڈیول ''math'' کا مواد

- اسٹینڈرڈ لائبریری میں موجود ماڈیول کو پائتھن کے Lib فولڈر میں دیکھا جا سکتا ہے۔

(B) From اسٹیٹمنٹ (From Statement)

ایک ماڈیول کو امپورٹ کر کے سبھی فنکشن کو میموری میں لوڈ کرنے کے بجائے ماڈیول میں موجود صرف مطلوبہ فنکشن کو ایکسس کرنے کے لیے from اسٹیٹمنٹ کا استعمال کیا جا سکتا ہے۔ یہ ماڈیول کے سبھی فنکشن کو لوڈ کرنے کے بجائے صرف معینہ فنکشن کو ہی لوڈ کرتا ہے۔ اس کا سنٹیکس مندرجہ ذیل ہے:

```
>>> from modulename import functionname [,
functionname,...]
```

from اسٹیٹمنٹ کی مدد سے امپورٹ کر کے فنکشن کو استعمال کرنے کے لیے ہمیں اس کے پہلے ماڈیول کا نام لکھنے کی ضرورت نہیں ہے۔ ہم سبھی فنکشن کو براہ راست طلب (کال) کر سکتے ہیں جیسا کہ مندرجہ ذیل مثالوں میں دکھایا گیا ہے:

مثال 7.5

```
>>> from random import random
>>> random()       #Function called without
the module name
```

نتیجہ:

```
0.9796352504608387
```

پروگرامنگ سے متعلق بہترین عادت:
ماڈیول کو امپورٹ کرنے کے بجائے صرف مطلوبہ فنکشن کو استعمال کرنے سے میموری کی بچت ہوتی ہے۔

مثال 7.6

```
>>> from math import ceil,sqrt
>>> value = ceil(624.7)
>>> sqrt(value)
```

نتیجہ:

```
25.0
```

مثال 7.2 میں، 624.7 کی زیادہ سے زیادہ قدر (ceil value) متغیر ''value'' میں اسٹور ہو جاتی ہے اور اس کے بعد متغیر ''value'' پر sqrt فنکشن کا اطلاق ہوتا ہے۔ مذکورہ بالا مثال کو دوبارہ مندرجہ ذیل طریقے سے لکھا جا سکتا ہے:

```
>>> sqrt(ceil(624.7))
```

فنکشن sqrt() کی تعمیل کا انحصار ceil() فنکشن کے نتیجے پر ہوتا ہے۔

اگر ہم 624.7 کے صحیح عددی حصے کو علاحدہ کرنا چاہتے ہیں (ہم میتھ ماڈیول کے trunc() کا استعمال کریں گے) تو ہم مندرجہ ذیل بیانات کو استعمال کر سکتے ہیں۔

```
#ceil and sqrt already been imported above
>>> from math import trunc
>>> sqrt(trunc(625.7))
```

نتیجہ :

```
25.0
```

پروگرام کا وہ بیان جس میں فنکشن یا عبارتیں (Expressions) کسی نتیجے کے حصول کے لیے ایک دوسرے پر منحصر ہوں اسے کمپوزیشن (Composition) کہتے ہیں۔ کمپوزیشن کی کچھ اور مثالیں ذیل میں دی گئی ہیں۔

- `a = int(input("First number: "))`
- `print("Square root of ",a ," = ",math.sqrt(a))`
- `print(floor(a+(b/c)))`
- `math.sin(float(h)/float(c))`

پائتھن اسٹینڈرڈ لائبریری میں دست یاب ماڈیول کے علاوہ ہم اپنے خود کے فنکشن پر مشتمل ماڈیول تیار کر سکتے ہیں۔

پروگرام 7-6 استعمال کنندہ کے ذریعے تعریف شدہ ماڈیول basic_math کی تشکیل کیجیے جو استعمال کنندہ کے ذریعے تعریف شدہ مندرجہ ذیل فنکشن پر مشتمل ہو۔

1۔ دو اعداد کو جمع کرنا اور ان کے حاصل جمع کو ظاہر کرنا۔
2۔ دو اعداد کو گھٹانا اور ان کے حاصل فرق کو ظاہر کرنا۔
3۔ دو اعداد کو ضرب کرنا اور ان کے حاصل ضرب کو ظاہر کرنا۔
4۔ دو اعداد کو تقسیم کرنا اور ان کے خارج قسمت کو ظاہر کرنا اور اگر نسب نما صفر ہے تو غلطی "Division by Zero" کو پرنٹ کرنا۔
5۔ ماڈیول کی وضاحت کے لیے docstring کو شامل کرنا۔ ماڈیول کی تشکیل کے بعد فنکشن کو امپورٹ اور ایگزیکیوٹ کیجیے۔

"""Docstrings""" کو پائتھن دستاویز سازی اسٹرنگ بھی کہتے ہیں۔ یہ کئی سطروں پر تبصرہ مشتمل ہے جسے ماڈیول، فنکشن وغیرہ کو بیان کرنے کے لیے شامل کیا جاتا ہے۔ انھیں عام طور سے 3 دوہرے واوین کا استعمال کر کے پہلی سطر میں شامل کیا جاتا ہے۔

```
#Program 7-16
#The requirement is:
  #1. Write a docstring describing the module.
  #2. Write user defined functions as per the specification.
  #3. Save the file.
  #4. Import at shell prompt and execute the functions.

  """
          basic_math Module
        ********************
  This module contains basic arithmetic operations
  that can be carried out on numbers

  """
  #Beginning of module
```

```
def addnum(x,y):
    return(x + y)
def subnum(x,y):
    return(x - y)
def multnum(x,y):
    return(x * y)
def divnum(x,y):
    if y == 0:
        print ("Division by Zero Error")
    else:
        return (x/y)          #End of module
```

نتیجہ:

```
#Statements for using module basic_math
>>> import basic_math
#Display descriptions of the said module
>>> print(basic_math.__doc__)

        basic_math Module
       ******************
This module contains basic arithmetic operations
that can be carried out on numbers

>>> a = basic_math.addnum(2,5)  #Call addnum() function of the
>>> a                           #basic_math module
7
>>> a = basic_math.subnum(2,5)  #Call subnum() function of the
>>> a                           #basic_ math module
-3
>>> a = basic_math.multnum(2,5) #Call multnum() function of the
>>> a                           #basic_math module
10
>>> a = basic_math.divnum(2,5)  #Call divnum() function of the
>>> a                           #basic_math module
0.4
>>> a = basic_math.divnum(2,0)  #Call divnum() function of the
Zero Divide Error               #basic_math module
```

__doc__ متغیر docstring کو اسٹور کرتا ہے۔ ماڈیول کے docstring کو ظاہر کرنے کے لیے ہمیں ماڈیول کو امپورٹ کرنا ہوگا۔ مندرجہ ذیل کو ٹائپ کیجیے۔

```
print(<modulename>.__doc__)          #__ are 2 underscore without space
```

نوٹ

خلاصہ

- پروگرامنگ میں فنکشن کو جداسازی (modularity) اور بار بار استعمال کے جانے کی خصوصیت (reusability) کے حصول کے لیے استعمال کیا جاتا ہے۔
- فنکشن کی تعریف ان ہدایات کے گروپ نام کے طور پر کی جاسکتی ہے جو طلب کرنے پر ایک مخصوص کام کو انجام دیتی ہیں۔ پروگرامر اپنے خود کے فنکشن تحریر کرسکتا ہے۔ اس قسم کے فنکشن کو استعمال کنندہ کے ذریعے تعریف شدہ فنکشن (user defined functions) کہا جاتا ہے۔
- پائتھن انٹرپریٹر میں بہت سے فنکشن پہلے ہی سے موجود ہوتے ہیں۔ یہ ایسے فنکشن ہیں جنھیں پروگرام میں بار بار استعمال کیا جاتا ہے۔ اس قسم کے فنکشن بِلٹ اِن فنکشن (built-in functions) کہلاتے ہیں۔
- فنکشن کال کے دوران فنکشن کو پاس کی جانے والی وہ قدر (ویلیو) جسے فنکشن ہیڈر میں متعین کیے گئے پیرامیٹر میں حاصل کیا جاتا ہے آرگیومینٹ (argument) کہلاتی ہے۔
- پائتھن میں یہ سہولت دست یاب ہے کہ پیرامیٹر کو ڈیفالٹ ویلیو تفویض کی جاسکتی ہے۔
- ایک فنکشن return بیان کا استعمال کرکے کالنگ فنکشن کو قدر (یا قدریں) بھیجتا ہے۔
- پائتھن میں ایک سے زیادہ قدروں کو ٹپل کا استعمال کرکے ظاہر کیا جاسکتا ہے۔
- عمل درآمد کے تسلسل (flow of execution) کی تعریف اس ترتیب کے طور پر کی جاسکتی ہے جس کے تحت پروگرام کے بیانات ہدایات کی تعمیل کرتے ہیں۔
- پروگرام کا وہ حصہ جہاں کسی متغیر کو ایکسس کیا جاسکتا ہے اس متغیر کا اسکوپ کہلاتا ہے۔
- وہ متغیر جسے کسی مخصوص فنکشن یا بلاک کے باہر متعین کیا جاتا ہے اسے گلوبل متغیر کہا جاتا ہے۔ اس متغیر کو پروگرام میں کہیں سے بھی ایکسس کیا جاسکتا ہے۔
- وہ متغیر جسے کسی فنکشن یا کسی بلاک کے اندر متعین کیا جاتا ہے اسے لوکل متغیر کہا جاتا ہے۔ اس متغیر کو صرف اسی فنکشن یا بلاک میں ایکسس کیا جاسکتا ہے جہاں اسے متعین کیا گیا ہے۔ یہ صرف فنکشن کے ایگزیکیوٹ ہونے یا اس کے فعال رہنے تک موجود رہتا ہے۔
- پائتھن معیاری لائبریری میں فنکشن اور ماڈیول کا ایسا وسیع تر مجموعہ ہے جو پروگراموں کو تیزی سے فروغ دینے میں پروگرامر کی مدد کرتا ہے۔
- ماڈیول ایک پائتھن فائل ہے جس میں متعدد فنکشن کی تعریفات موجود ہوتی ہیں۔
- پروگرام میں ماڈیول کو امپورٹ کرنے کے لیے امپورٹ اسٹیٹمنٹ کا استعمال کیا جاتا ہے۔

نوٹ

- ماڈیول کو صرف ایک مرتبہ لوڈ کیا جاتا ہے۔
- پروگرام میں کسی ماڈیول سے مخصوص فنکشن کو امپورٹ کرنے کے لیے from اسٹیٹمنٹ کا استعمال کیا جاسکتا ہے۔

مشق

1۔ مندرجہ ذیل پروگراموں کا بغور مشاہدہ کیجیے اور ان میں موجود اغلاط کی نشاندہی کیجیے۔

a)
```
def create (text, freq):
    for i in range (1, freq):
        print text
create(5)           #function call
```

b)
```
from math import sqrt,ceil
def calc():
    print cos(0)
calc()              #function call
```

c)
```
mynum = 9
def add9():
    mynum = mynum + 9
    print mynum
add9()               #function call
```

d)
```
def findValue( vall = 1.1, val2, val3):
    final = (val2 + val3)/ vall
    print(final)
findvalue()          #function call
```

e)
```
def greet():
    return("Good morning")
greet() = message  #function call
```

2۔ math.ceil(89.7) اور math.floor (89.7) کے درمیان کیا فرق ہے؟

3۔ 1 اور 5 کے درمیان بے ترتیب اعداد (Random Numbers) کی تشکیل کے لیے ہمیں random() اور randint() میں سے کون سے فنکشن کا استعمال کرنا چاہیے؟

4۔ بلٹ ان فنکشن pow()، فنکشن math.pow() سے کس طرح مختلف ہے؟ مثال کی مدد سے وضاحت کیجیے۔

5۔ ایک مثال کی مدد سے دکھایئے کہ پائتھن میں ایک فنکشن متعدد قدروں کو کس طرح ظاہر کرسکتا ہے؟

6۔ ایک مثال کی مدد سے مندرجہ ذیل کے درمیان فرق بتایئے۔

نوٹ

a) آرگیومنٹ اور پیرامیٹر

b) گلوبل اور لوکل متغیر

7۔ کیا فنکشن ہمیشہ ہی کوئی قدر ظاہر کرتا ہے؟ ایک مثال کی مدد سے وضاحت کیجیے۔

سرگرمی پر مبنی سوالات (ACTIVITY-BASED QUESTIONS)

نوٹ: پروگرام لکھنے سے مراد ہے:

- تبصروں کو دستاویز سازی کے حصے کے طور پر شامل کرنا
- فنکشن ڈیفینیشن لکھنا
- فنکشن کو فنکشن کال کے ذریعے ایگزیکیوٹ کرنا

1۔ اپنے کھاتے کو محفوظ رکھنے کے لیے، خواہ وہ ای میل اکاؤنٹ ہو، آن لائن بینک اکاؤنٹ ہو یا کوئی اور اکاؤنٹ ہو، یہ ضروری ہے کہ ہم تصدیق کاری کا استعمال کریں۔ اپنی پروگرامنگ مہارتوں کا استعمال کرتے ہوئے ایک پروگرام کی تشکیل کیجیے اور اس میں استعمال کنندہ کے ذریعے تعریف شدہ فنکشن (جسے login نام دیا گیا ہو) کا استعمال کیجیے جو استعمال کنندہ کی آئی ڈی اور پاس ورڈ کو پیرامیٹر (login(uid,pwd)) کے طور پر قبول کرتا ہے اور تین غلط کوششوں کے نتیجے میں ایک پیغام ''account blocked'' ظاہر کرتا ہے۔ اگر استعمال کنندہ یوزر ID کے لیے ''ADMIN'' اور پاس ورڈ کے لیے ''St0rE@1'' داخل کرتا ہے تو لاگن کامیاب ہو جاتا ہے۔ کامیاب لاگن پر ''login successful'' پیغام کو ظاہر کرتا ہے۔

2۔ ایک XYZ اسٹور تہوار کے موقع پر اپنے گاہکوں کو رعایت دینے کا منصوبہ بنا رہا ہے۔ اسٹور انتظامیہ نے مندرجہ ذیل معیارات کی بنیاد پر رعایت دینے کا فیصلہ کیا ہے۔

خریداری کی رقم	رعایت کی پیش کش
>=500 and <1000	5%
>=1000 and <2000	8%
>=2000	10%

جو گاہک اسٹور کے ممبر ہیں انھیں 5% کی اضافی رعایت دی جاتی ہے جو اسٹور کے ممبر ہیں۔ استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کر کے ایک پروگرام تشکیل دیجیے جو خریداری کی رقم کو پیرامیٹر کے طور پر قبول کرتا ہے اور مندرجہ ذیل شرائط کی بنیاد پر رعایت اور خالص قابل ادا رقم کی تحسیب کرتا ہے۔

رعایت - خریداری کی کل رقم = خالص قابل ادا رقم

نوٹ

3۔ 'کھیلیں اور سیکھیں' ایک ایسی حکمت عملی ہے جس کی مدد سے بچے تصورات کو تفریحی انداز میں سیکھ سکتے ہیں۔ ایک سینئر طالب علم ہونے کے ناطے آپ نے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کر کے ایک پروگرام تیار کرنے کی ذمے داری لی ہے تا کہ بچوں کو انگریزی حروف تہجی کا استعمال کر کے دو یا تین حرفی الفاظ بنانے اور ایک ہندسی اعداد کو جمع کرنے میں مہارت حاصل ہو سکے۔ اس بات کو یقینی بنائیں کہ آپ نے ان سوالات کی قسم کا بغور تجزیہ کیا ہے جنھیں عمر اور درسیات کے مطابق شامل کیا جا سکتا ہے۔

4۔ اعداد کے مندرجہ ذیل سلسلے پر غور کیجیے۔

1, 1, 2, 3, 5, 8, 13, 21, 34, 55…

پیٹرن بنانے کے لیے، 1 اور 1 لکھ کر شروع کریں۔ انھیں جمع کر کے 2 حاصل کیجیے۔ آخری دو اعداد کو جمع کیجیے: 1+2 = 3۔ سلسلہ کا اگلا عدد حاصل کرنے کے لیے پچھلے دو اعداد کو جمع کرتے جائیے۔ یہ اعداد مشہور فیبو ناکی سلسلہ کی تشکیل کرتے ہیں: نیا عدد حاصل کرنے کے لیے سابقہ دو اعداد کو جمع کیا جاتا ہے۔

5۔ مندرجہ ذیل تحسیبات کو انجام دینے والے کیلکولیٹر کے نفاذ کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کر کے مینو کی مدد سے چلنے والے پروگرام کی تشکیل کیجیے۔

(a بنیادی حسابی عمل $(+,-,*,/)$

(b $\log_{10}(x), \sin(x), \cos(x)$

لیب کے لیے مجوزہ مشقیں (SUGESTED LAB EXERCISES)

1۔ کسی عدد کی 7 سے تقسیم پذیری کی جانچ کرنے کے لیے ایک پروگرام لکھیے جو استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کو پیرامیٹر کے طور پر پاس کیا جاتا ہے۔

2۔ ایک پروگرام تحریر کیجیے جو استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کرتا ہے اور یہ فنکشن نام اور جنس (مرد کے لیے M اور خاتون کے لیے F) کو قبول کرتا ہے اور جنس کی بنیاد پر نام کے ساتھ Mr/Ms کو بطور سابقہ استعمال کرتا ہے۔

3۔ ایک پروگرام لکھیے جو متغیرات میں دو درجی مساوات کے ضریبوں کو قبول کرنے کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن پر مشتمل ہے اور اس کے ڈٹرمنینٹ کی تحسیب کرتا ہے۔ مثلاً: اگر ضریبوں کو متغیرات a,b,c میں اسٹور کیا جاتا ہے تو ڈٹرمنینٹ کی تحسیب b^2-4ac کے طور پر کیجیے۔ مثبت، صفر اور منفی ڈٹرمنٹ کی جانچ کرنے کے لیے مناسب شرط لکھیے اور درست نتیجہ ظاہر کیجیے۔

4۔ ABC اسکول نے اپنی سالانہ تقریب کے موقع پر سبھی والدین کو لکی ڈرا میں سہولت فراہم کرنے کے لیے منفرد ٹوکن ID (1 تا 600) جاری کیے ہیں۔ جیتنے والے کو ایک خصوصی انعام دیا جائے گا۔ پانچھن

نوٹ

کا استعمال کر کے ایک پروگرام لکھیے جو اس کام کو خود کار انداز میں انجام دینے میں مدد کر سکے (اشارہ: رینڈم ماڈیول کا استعمال کیجیے)

5۔ ایک پروگرام لکھیے جو استعمال کنندہ کے ذریعے تعریف شدہ ایسے فنکشن کا استعمال کرتا ہے جو مرکب سود کی تحسیب اور اسے ظاہر کرنے کے لیے اصل زر (P)، شرح (R)، مدت (T)، جتنی مرتبہ سود کی تحسیب ہوتی ہے اس کی تعداد (NT) کو قبول کرتا ہے۔

(اشارہ: $CI=P*(1+r/n)^{nt}$)

6۔ ایک پروگرام لکھیے جس میں اعداد کو پیرامیٹر کے طور پر قبول کرنے کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن ہے اگر عدد 1، عدد 2 سے چھوٹا ہے تو اعداد کو آپس میں بدلا اور ریٹرن کیا جاتا ہے، یعنی عدد 2 کو عدد 1 کی جگہ ریٹرن کیا جاتا ہے اور عدد 1 کو عدد 2 کی جگہ، بصورت دیگر اسی ترتیب میں ریٹرن کیا جاتا ہے۔

7۔ ایک پروگرام لکھیے جو مربع، مستطیل، مثلث، دائرہ، استوانہ جیسی مختلف اشکال کا رقبہ، احاطہ یا سطحی رقبہ (جو بھی قابل اطلاق ہے) کی تحسیب کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن پر مشتمل ہے۔ استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کو تحسیب کے لیے پیرامیٹر کے طور پر قدروں کو قبول کرنا چاہیے اور تحسیب شدہ قدر کو ریٹرن کرنا چاہیے۔ ماڈیول کو امپورٹ کریں اور مناسب فنکشن کو استعمال کریں۔

8۔ ایک پروگرام لکھیے جو ایک ایسے GK کوئز کی تشکیل کرتا ہے جس میں آپ کی پسند کے کوئی بھی پانچ سوال موجود ہوں۔ سوالات کو بلا منصوبہ طور پر (randomly) ظاہر کیا جانا چاہیے۔ کوئز کے اسکور کی تحسیب کے لیے استعمال کنندہ کے ذریعے تعریف شدہ فنکشن score() کی تشکیل کیجیے۔ ایک اور استعمال کنندہ کے ذریعے تعریف شدہ فنکشن remark (scorevalue) کی تشکیل کیجیے جو مندرجہ ذیل کے مطابق ریمارک کو ظاہر کرنے کے لیے آخری اسکور کو قبول کرتا ہے۔

نمبر	ریمارکس
5	نمایاں
4	شاندار
3	اچھا
2	زیادہ اسکور کرنے کے لیے اور زیادہ پڑھیں۔
1	دلچسپی لینے کی ضرورت ہے۔
0	معلومات عامہ ہمیشہ آپ کے لیے معاون ثابت ہوگی۔ اسے سنجیدگی سے لیں۔

نوٹ

نظیری مطالعہ پر مبنی سوالات (CASE STUDY-BASED QUESTION)

باب 6 میں SMIS کے لیے آیئے مندرجہ ذیل کاموں کو انجام دیں۔

1۔ 7.1 استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا استعمال کرکے باب 5 اور 6 میں سبھی فعلیت (Functionality) تبدیل کیجیے۔

2۔ 7.2 اس بات کی جانچ کرنے کے لیے کہ آیا کسی طالب علم کی حاضری مقررہ تعداد سے کم ہے یا نہیں، مذکورہ بالا مینو میں ایک اور استعمال کنندہ کے ذریعے تعریف شدہ فنکشن کا اضافہ کیجیے۔ یہ فنکشن ایک ماہ میں کام کے دنوں کی کل تعداد کو قبول کرے گا اور فارمولے کی مدد سے اس کی حاضری کا حساب لگا کر اس بات کی جانچ کرے گا کہ آیا طالب علم ڈیفالٹر ہے یا نہیں: طالب علم کے حاضر رہنے کی دنوں کی تعداد یا کام کے دنوں کی کل تعداد شمار کریں۔ اگر تحسیب شدہ حاضری % 78 سے کم ہے تو فنکشن کو کم حاضری کا اظہار کرتے ہوئے 1 ریٹرن کرنا چاہیے بصورت دیگر فنکشن کو حاضری کم نہ ہونے کا اظہار کرتے ہوئے 0 ریٹرن کرنا چاہیے۔

آیئے باب 5 کے آخر میں ''دستاویز سازی کے لیے مشورے'' (Documentation Tips) کے تحت دیے گئے پیرامیٹر کی بنیاد پر دوسرے طلبا کے نظیری مطالعہ پر نظر ثانی کریں اور انھیں بازرسی فراہم کریں۔