Functional Programming

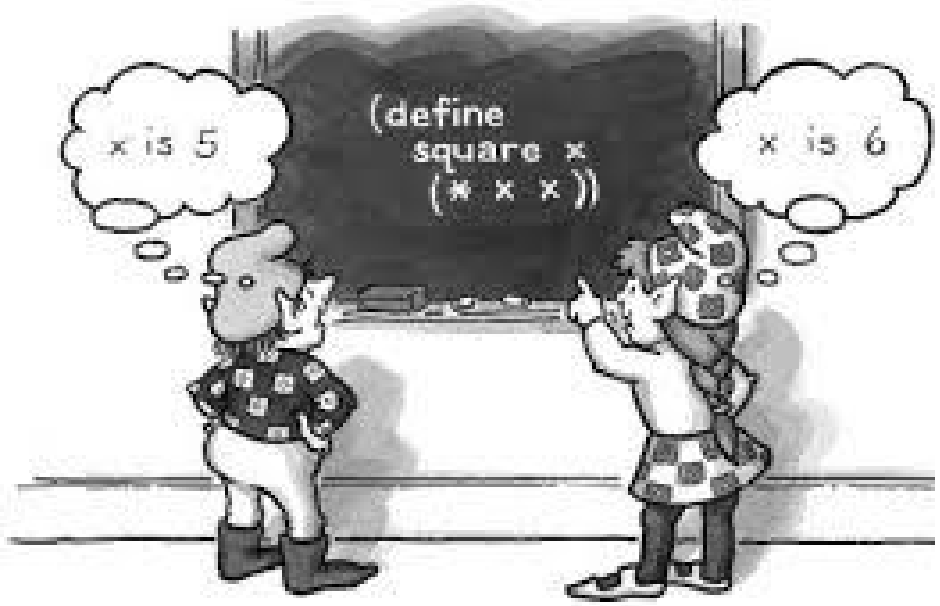
Pure Functions

Kasun de Zoysa
kasun@ucsc.cmb.ac.lk

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

Function

Functional programming languages are geared to support the creation of highly reusable and composable functions and to help developers organize their code base around them.



My First Scala Program

You type

```
scala> 1+1
res2: Int = 2
scala>
```

Scala shows the result at REPL.

```
scala> def sum(x:Int,y:Int):Int=x+y
sum: (x: Int, y: Int)Int

scala> sum(1,1)
res2: Int = 2
```

Scala Function

A function **declaration** has the following form:

- `def functionName ([list of parameters]): [return type]`

A function **definition** has the following form:

- `def functionName ([list of parameters]) : [return type] = { function body return [expr]}`

Last Week

ºF =ºC * 1.8000 + 32.00

```
scala> 35*1.8+32
res0: Double = 95.0
```

The volume of a sphere with radius r is 4/3 Pi r3.
What is the volume of a sphere with radius 5?

```
scala> val r=3.0
r: Double = 3.0
Scala> 4.0/3.0*math.Pi*r*r*r
res1: Double = 113.09733552923255
```

Last Week

ºF =ºC * 1.8000 + 32.00

```
def F(x:Double)=x*1.8+32.0
```

The volume of a sphere with radius r is 4/3 Pi r3.
What is the volume of a sphere with radius 5?

```
def volume(r:Double)=4/3*math.Pi*r*r*r
```

Return value of a function is the result of the body expression ({} are optional in this case)

Pure Function

Functional programming languages are geared to support the creation of highly reusable and composable functions and to help developers organize their code base around them.

Function is one that:

- Has one or more input parameters
- Performs calculations using only the input parameters
- Returns a value
- Always returns the same value for the same input
- Does not use or affect any data outside the function
- Is not affected by any data outside the function

Problems

- I run 2 km at an easy pace (8 min per km), then 3 km at Tempo (7 min per km) and 2 km at easy pace again to reach home. What is the total running time?

- Suppose the cover price of a book is Rs. 24.95, but bookstores get a 40% discount. Shipping costs Rs. 3 for the first 50 copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

Running Time

Running time for easy phase:

```
def easy(x:Int):Int=x*8
```

Running time for tempo phase:

```
def tempo(x:Int):Int=x*7
```

The total running time:

```
scala> easy(2)+tempo(3)+easy(2)
res24: Int = 53
```

Book Price

The total amount for 60 books:

```
def bookPrice(x:Int):Double=x*24.95
```

Discount:

```
def discount(amount:Double):Double=
amount*.4
```

Shipping Cost:

```
def shippingCost(x:Int):Double=3*x+(x-50)*.75
```

The total wholesale cost for 60 books:

```
bookPrice(60)-discount(bookPrice(60))
+shippingCost(60)
```

Why it is worth the trouble to divide a program into functions?

There are several reasons:

•Creating a new function gives you an opportunity to name a group of statements, which makes your program easier to read and debug.

•Functions can make a program smaller by eliminating repetitive code. Later, if you make a change, you only have to make it in one place.

•Dividing a long program into functions allows you to debug the parts one at a time and then assemble them into a working whole.

•Well-designed functions are often useful for many programs. Once you write and debug one, you can reuse it.

Problem

Area of a disk with radius r is Pi r*r. What is the area of a disk with radius 5?

- A good name for a function is **`areaOfDisk`**.
- Using this name, we would express the function for computing the area of a disk.

areaOfDisk is a functions, that it consumes a single INPUT, called r, and that the result, or OUTPUT, is going to be Pi*r*r.

Area of Disk

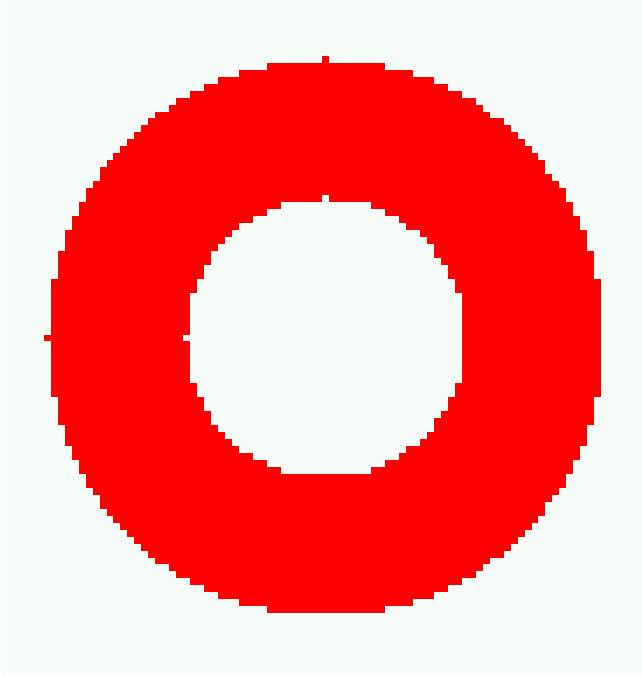
The application of a defined operation is evaluated by copying the function named areaOfDisk and by replacing the variable (r) with the number we supplied (5):

```
scala> def areaOfDisk(x:Double):Double=r*r*math.Pi
areaOfDisk: (x: Double)Double

scala> areaOfDisk(5)
res17: Double = 63.61725123519331
```

The Area of a Ring

Many programs consume more than one input. Say we wish to define a program that computes the area of a ring, that is, a disk with a hole in the center:



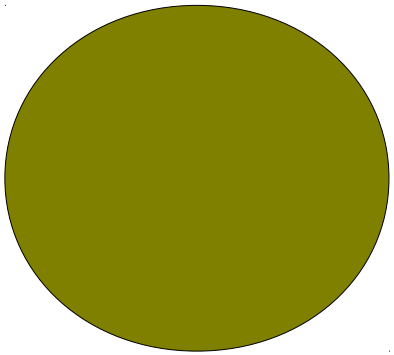
The Area of a Ring

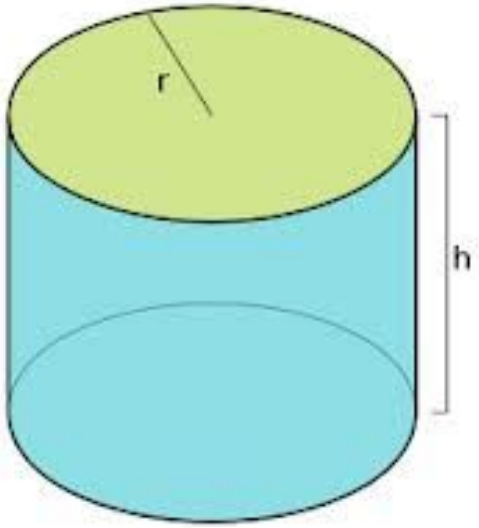
The area of the ring is that of the outer disk minus the area of the inner disk, which means that the program requires two unknown quantities: the outer and the inner radii. Then the program that computes the area of a ring is defined as follows:

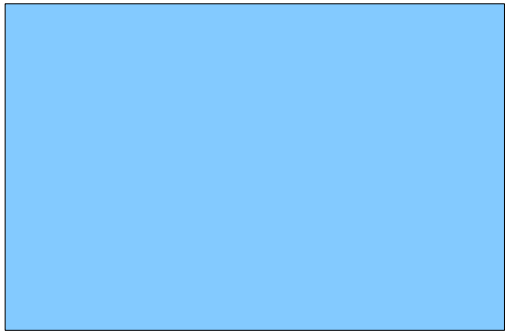
```
scala>def areaOfRing(x:Double,y:Double):Double=
     | areaOfDisk(x)-areaOfDisk(y)
areaOfRing: (x: Double, y: Double)Double

scala> areaOfRing(10,5)
res3: Double = 235.61944901923448
```

Problems : Area of Cylinder

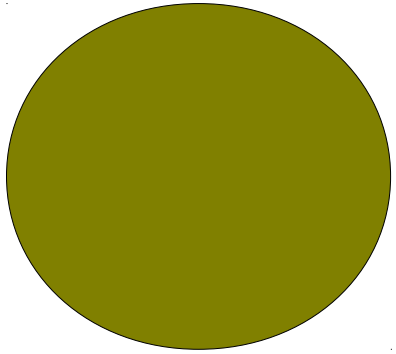


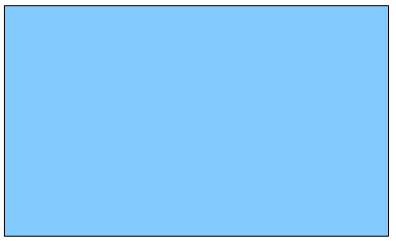




Scala Functions

```
//Calculate the area of disk
//by giving the radius:
scala> def areaOfDisk(r:Double):
Double=r*r*math.Pi
areaOfDisk: (r: Double)Double
```





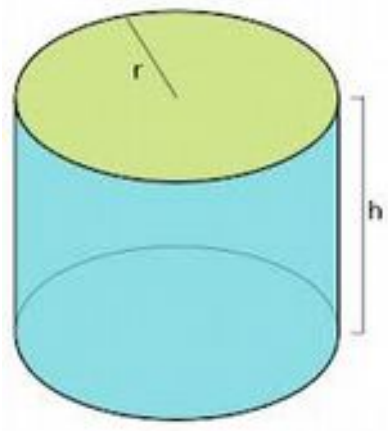
```
//Calculate the area of rectangle by
//giving the width and hight:
scala> def areaOfRec(width:Double,hight:Double)
:Double=width*hight
areaOfRec: (width: Double, hight: Double)Double
```

Scala Functions

```
//Calculate the area of Cylinder
//by giving the radius and hight:
scala> def areaOfCylinder(radius:Double,
hight:Double):Double =
areaOfDisk(radius)*2 +
areaOfRec(2*math.Pi*radius,hight)
areaOfCylinder: (radius: Double, hight:
Double)Double

scala> areaOfCylinder(5,6)
res0: Double = 345.5751918948772

scala> printf("%.2f",areaOfCylinder(5,6))
345.58
```



Consider the following problem:

Company XYZ & Co. pays all its employees Rs.150 per normal working hour and Rs. 75 per OT hour. A typical employee works 40 (normal) and 20(OT) hours per week has to pay 10% tax. Develop a program that determines the take home salary of an employee from the number of working hours and OT hours given.

How many functions???

We have following functions:

1. wage=hours*150
2. ot = hours*75
3. income=wage+ot
4. tax = income* .1
5. takeHome=income-tax

Scala functions:

1. wage=hours*150

```
def wage(hours:Int):Int=hours*150
```

2. ot = hours*75

```
def ot(hours:Int):Int=hours*75
```

3. income=wage+ot

```
def income(h1:Int,h2:Int):Int=wage(h1)+ot(h2)
```

4. tax = income* .1

```
def tax(income:Int):Int=income*.1
```

5.takeHome=income-tax

```
def takeHome(h1:Int,h2:Int):Double=
income(h1,h2)-tax(income(h1,h2))
```

Consider the following problem:

Imagine the owner of a movie theater who has complete freedom in setting ticket prices. The more he charges, the fewer the people who can afford tickets.

In a recent experiment the owner determined a precise relationship between the price of a ticket and average attendance.

At a price of Rs 15.00 per ticket, 120 people attend a performance. Decreasing the price by 5 Rupees increases attendance by 20 and increasing the price by 5 Rupees decreases attendance by 20.

Unfortunately, the increased attendance also comes at an increased cost. Every performance costs the owner Rs.500. Each attendee costs another 3 Rupees.

The owner would like to know the exact relationship between profit and ticket price so that he can determine the price at which he can make the highest profit.

Simplifying the problem:

When we are confronted with such a situation, it is best to tease out the various dependencies one at a time:

1)Profit is the difference between revenue and costs.

2)The revenue is exclusively generated by the sale of tickets. It is the product of ticket price and number of attendees.

3)The costs consist of two parts: a fixed part (Rs.500) and a variable part (Rs. 3 per attendee) that depends on the number of attendees.

4)Finally, the problem statement also specifies how the number of attendees depends on the ticket price.

Identifying the functions:

1. Calculate the number of attendees by giving the ticket price:

```
attendees= 120 + (15-Ticket price)/5*20
```

2. Calculate the revenue by giving the ticket price:

```
Revenue=attendees*price
```

3. Calculate the cost by giving the ticket price:

```
cost = 500 + 3 * attendees
```

4. Calculate the profit by giving the ticket price:

```
Profit= Revenue - Cost
```

Scala functions:

Calculate the number of attendees by giving the ticket price: **attendees= 120 + (15-Ticket price)/5*20**

```
def attendees(price:Int):Int=120+(15-price)/5*20
```

Calculate the revenue by giving the ticket price:**Revenue=attendees*price**

```
def revenue(price:Int):Int = attendees(price)*
price
```

Calculate the cost by giving the ticket price: **cost = 500 + 3 * attendees**

```
def cost(price:Int):Int=500+attendees(price)
```

Scala functions:

```
Calculate the profit by giving the ticket price:
Profit= Revenue - Cost
def profit(price:Int):Int =
revenue(price)- cost(price)


Scala> print(profit(5), profit(10), profit(15),
profit(20))
(140,760,1180,1400)

scala> print(profit(25), profit(30), profit(35),
profit(40))
(1420,1240,860,280)
```

Discussion