CPSC203: WEEK-2 LAB-2
PROBLEM SOLVING (Functions)

-Prepared By
Nashad Ahmed Safa
Graduate Student
Department of Computer Science

COURSE WEBSITE

http://wiki.ucalgary.ca/page/Courses/Computer_Science/CPSC_203/CPSC_203_Template

FUNCTIONS

- Jython uses the **def** statement to define functions. Functions related statements include:

  def, return

  The def statement creates a function object and assigns it to a name. The return statement sends a result object back to the caller. This is optional, and if it is not present, a function exits so that control flow falls off the end of the function body.

FUNCTIONS

global

- The global statement declares module-level variables that are to be assigned. By default, all names assigned in a function are local to that function and exist only while the function runs. To assign a name in the enclosing module, list functions in a global statement.

FUNCTIONS

- The basic syntax to define a function is the following:

```
def name (arg1, arg2, ... ArgN)
statements
    return value
```

where name is the name of the function being defined. It is followed by an open parenthesis, a close parenthesis and a colon. The arguments inside parenthesis include a list of parameters to the procedures.

FUNCTIONS

(Cond) The next line after the colon is the body of the function - a group of commands that form the body of the function. After you define a Jython function, it is used just like any of the built-in functions. For example:

```
def add(number1, number2):
        answer = number1 + number2
return answer
```

FUNCTIONS

To call the function above, use the following command:

```
x = 1
y = 2
add(x, y)
[output=> 3]
```

LISTS

Python knows a number of compound data types, used to group together other values. The most versatile is the list, which can be written as a list of comma-separated values (items) between square brackets. List items need not all have the same type.

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]
```

LISTS

List indices start at 0, and lists can be sliced, concatenated and so on:

```
>>> a[0]
'spam’
>>> a[0:3]
['spam', 'eggs', 100]
>>> a[:2] + ['bacon', 2*2]
['spam', 'eggs', 'bacon', 4]
```

LISTS

Here are some of the methods of list objects:

append(x): Add an item to the end of the list

insert(i,x): Insert an item at a given position.

remove(x): Remove the first item from the list whose value is *x*

index(x): Return the index in the list of the first item whose value is *x*

count(x): Return the number of times *x appears in the list.*

sort( ): Sort the items of the list, in place.

reverse(): Reverse the elements of the list

len(): Give the length of the list.

SOME USEFUL BUILT IN FUNCTIONS
(There are hundreds of them)

range([start,] stop[, step])

raw_input([prompt]) The function reads a line from input, converts it to a string, and returns that

int([x[, radix]]): Convert a string or number to a plain integer

type(object): Return the type of an object

str([object]): Return a string containing a nicely printable representation of an object.

FUNCTIONS

Example 1: Receives a alphabet grade as input from the user, and outputs the equivalent numerical grade.

```
def grades():
    strInput = raw_input('Enter your grade: ')
    grade = int(strInput)
    print 'grade is', grade
    if grade >= 90 and grade < 100:
            print 'A'
    elif grade >=80:
            print 'B'
     elif grade >= 70:
            print 'C'
    elif grade >=60:
            print 'D'   else:
    print 'F'
```

FUNCTIONS

Example 2: Finds the minimum element in a list of elements

```
def empty(S):
        return len(S) == 0
def min(S):
    if empty(S):
            return 'undefined'
    else:
            min_so_far = S[0]
            for i in range(1,len(S)):
                    if S[i] < min_so_far:
                            min_so_far = S[i]
    return min_so_far
```

FUNCTIONS

Example 3: Sorts a list of numbers using the selection sort algorithm

```
def selectionSort(S):
        sortedS = []
        for i in range(0,len(S)):
                minElement = min(S)
                S.remove(minElement)
                sortedS.append(minElement)
        return sortedS
```

DRAWING ON IMAGES

So we can display images now we will manipulating them. Lets start by making a red dot on a blank picture.

```
def makePic():
# Make a new picture
newPic = makeEmptyPicture(200, 100)
# Get a pixel from the middle of it
pixel = getPixel(newPic, 100, 50)
 # Make that pixel red
setColor(pixel, red)
# Display the result
show(newPic)
```

This is nice, but drawing a single pixel at a time is going to be a little tedious. We can also draw lines, rectangles, ovals, and text.

DRAWING ON IMAGES

The function addLine takes a color and four coordinates x1, y1, x2, y2, and draws a straight line in that color. Note that you call this in a slightly di□erent way than the other functions we've seen so far. Don't worry about the di□erence for now.

```
def makePic():
    # Make a new picture
    newPic = makeEmptyPicture(200, 100)
    # Make a black line running from upper-left #to lower-right
    newPic.addLine(black, 0, 0, 200, 100)
    # Display the result
    show(newPic)
```

DRAWING ON IMAGES

The function addRect takes a color and four integers x, y, width, and height. It draws a rectangle of the given width and height with the position (x, y) as the upper left corner

```
def myFunction():
        newPic = makeEmptyPicture(200, 100)
        # Make a blue 30x30 square whose upper-
  #left corner is at 10, 20
        newPic.addRect(blue, 10, 20, 30, 30)
        show(newPic)
```

DRAWING ON IMAGES

The function addOval takes a color and four integers x, y, width, and height. It draws an oval whose left edge touches x, whose top edge touches y, and with the given width and height. To make a circle, just make width and height equal.

```
# Make a red oval centered at 80, 80
newPic.addOval(red, 80, 80, 20, 10)
```

DRAWING ON IMAGES

The function makeColor takes three integers red, green, and blue, and returns a new color. This way you can make your own colors if you know their red, green, and blue intensities! (0, 0, 0) is black, (255, 255, 255) is white, and (255, 0, 0) is red. (Note: you can also pick a color using the pickAColor() function.)

```
# Make a new color of mostly red and green, with a little blue
pukeColor = makeColor(196, 181, 84)
```

DRAWING ON IMAGES

The function addText takes a color, two coordinates x and y, and a string. It then writes that string starting at those coordinates in that color.

```
# Print a friendly message in that new color
  newPic.addText(pukeColor, 30, 80, "Hello  world!")
```

DRAWING ON IMAGES

The function writePictureTo takes a picture and a filename and writes the picture to that file. Be sure to include the extension ".jpg" so that other programs can recognize the file type.

```
# Display the results
show(newPic)
# Save the results to a new file
writePictureTo(newPic, "test.jpg")
```