Contents

1 Basic Test Results

```
Starting tests...
Thu Nov 20 15:27:46 IST 2014
110ab2c8b7aff49335f0c541986c313a38f639a4  -


-rw-r--r-- ransha/stud    2074 2014-11-20 15:24 README
-rw-r--r-- ransha/stud     421 2014-11-20 14:58 findLargest.py
-rw-r--r-- ransha/stud     848 2014-11-20 15:11 findSecondSmallest.py
-rw-r--r-- ransha/stud     443 2014-11-20 15:25 ithElementValue.py
-rw-r--r-- ransha/stud    1562 2014-11-20 15:18 twoDimensionalSeek.py
-rw-r--r-- ransha/stud     780 2014-11-20 15:12 totalWeight.py
-rw-r--r-- ransha/stud     685 2014-11-20 15:20 binaryToDecimal.py
-rw-r--r-- ransha/stud     685 2014-11-20 15:20 decimalToBinary.py
-rw-r--r-- ransha/stud     424 2014-11-20 14:58 decomposition.py

Testing README...
Done testing README...

Running presubmit tests...
result_code    ex3_example    10    1
Done running presubmit tests

Tests completed
```

2 README

```
ransha
203781000
Ran Shaham


=========================
=  README for ex3: Loops =
=========================



=================
=  Description:  =
=================
In this exercise I learned how to combine loops and conditions to solve
problems.

=============================
=  List of submitted files: =
=============================

README                  This file
findLargest.py        Finds the maximum in a given sequence of numbers
findSecondSmallest.py Finds the single number that is greater than the
                      minimum in a sequence of numbers
totalWeight.py        Sums the inputs until threshold or a keyword is reached
twoDimensionalSeek.py Calculates the absolute position in a plain
ithElementValue.py    Calculates the value of a chosen element in
                      Fibonacci sequence (general knowledge)
decomposition.py      Decomposes a chosen number to it's elements
                      from right to left
binaryToDecimal.py    Converts a chosen number from binary base to decimal
decimalToBinary.py    Converts a chosen number from decimal base to binary

========================
= Answers to question 9 =
========================

*After reading this question, I found that there is no significant
 difference between conversion between different bases, and therefore
 I re-wrote most of my code. The mathematical principle used to convert
 a number between bases is similar.

*The minimal change that needed to be done in the 7th or 8th question to
 solve the other one is swapping the 'to_base','from_base' variables values

*The new code I wrote can convert numbers from chosen base (smaller than 10)
 to another one, with the same changes that needed to be done for solving
 question 8 instead of 7 or the other way around. (base_source=from_base
                                                   base_target=to_base)
 For example, to change the 8th question code from decimal to binary
 to binary to decimal, the variable from_base needs to be changed to 2
 and the to_base variable needs to be changed to 10. That's it.
```

3 binaryToDecimal.py

```python
num_in= int(input("Insert number in binary representation:"))
#targest_base and source_base
FROM_BASE= 2
TO_BASE= 10

num_out= 0
stop= False
pos= 0 #holds the number of digit in the composed number
       #we are currently checking.

#a loop that goes through every figure in the input number
#to convert it to chosen base (decimal in this case)
while stop == False:
    if num_in // TO_BASE == 0: stop= True #finish iteration, then stop if
                                          #there are no more figures to check
    num_out+= (num_in % TO_BASE) * (FROM_BASE ** pos)
    num_in//= TO_BASE
    pos+= 1

print("The decimal value of the inserted binary number is", num_out)
```

4 decimalToBinary.py

```python
num_in= int(input("Insert number in decimal representation:"))
#targest_base and source_base
FROM_BASE= 10
TO_BASE= 2

num_out= 0
stop= False
pos= 0 #holds the number of digit in the composed number
       #we are currently checking.

#a loop that goes through every figure in the input number
#to convert it to chosen base (binary in this case)
while stop == False:
    if num_in // TO_BASE == 0: stop= True #finish iteration, then stop if
                                          #there are no more figures to check
    num_out+= (num_in % TO_BASE) * (FROM_BASE ** pos)
    num_in//= TO_BASE
    pos+= 1

print("The binary value of the inserted decimal number is", num_out)
```

5 decomposition.py

```python
gimli= int(input("Insert composed number:")) #gets Gimli’s input
stop= False
day= 0
goblets= 0
#this loop decomposes the input number for it’s decimal figures by division
#(reads the composed number, figure by figure and prints each figure)
while stop == False:
    if gimli// 10 == 0: stop= True
    goblets= gimli % 10
    gimli//= 10
    day+= 1
    print("The number of goblets Gimli drank on day", day, "was", goblets)
```

6 findLargest.py

```python
#a range for the loop
riders= range(int(input("Enter the number of riders:")))
high_hat= 0
gandalf_pos= 0

#This is the loop that goes through every hat size and
#checks which is the largest.
for rider in riders:
    height= float(input("How tall is the hat?"))
    if height > high_hat:
        high_hat= height
        gandalf_pos= rider + 1

print("Gandalf's position is:", gandalf_pos)
```

7 findSecondSmallest.py

```python
smallest= 0
smallest_pos= 0
second= 0
second_pos= 0     #this will be Pippin's position
NUM_OF_DANCERS= 10
dancers= range(NUM_OF_DANCERS)

#Loop that checks the age of each dancer and sorts.
for dancer in dancers:
    #Gets the age for each dancer (from input)
    age= int(input("What is the age of the current dancer?"))

    #Checks if current dancer is smaller then the smallest so far
    if age < smallest or smallest == 0: #or if it's the first run.
        second= smallest                #'second' gets the old 'smallest' values
        second_pos= smallest_pos
        smallest= age
        smallest_pos= dancer + 1

    #If not, Checks if it's the second smallest (or first run).
    elif age < second or second == 0:
        second= age
        second_pos= dancer + 1


#prints the output
print("Pippin is dancer number", second_pos)
```

8 ithElementValue.py

```python
orc=[]          #a list variable that holds the number of arrows needed to
                #kill an indexed orc. orc[num] = number_of_arrows
orc.append(1)
orc.append(1)
arrows= 0
orcs= int(input("Which Orc do you wish to confront?"))

#calculates the number of arrows for desired orc
for orc_num in range(2, orcs):
        orc.append(orc[orc_num - 1] + orc[orc_num - 2])

arrows= orc[orcs - 1]
print("The required number of arrows is", arrows)
```

9 totalWeight.py

```python
bag= 0
item= 0
GANDALF_MAX= 100
STOP_VALUE= -1    #the ring (my precious)

print("Insert weights one by one:")

#reads the input until a threshold or stop value is reached
while item != STOP_VALUE:
    item= int(input())
    #the stop value is the 'ring' and it's weight is -1
    if item == STOP_VALUE:
        continue

    elif item < 0:            #invalid input, prints error and continues
        print("Weights must be non-negative")

    else:                     #correct input, sums the weight
        bag+= item
        if bag > GANDALF_MAX: #checks if the threshold is reached
            print("Overweight! Gandalf will not approve.")
            break

#this is reached when the stop value is entered
else:
    print("The total packed weight is", bag)
```

10 twoDimensionalSeek.py

```python
position= [0,0]
heading= 0 #I will define 4 directions: 0,90,180,270
           #for forward,right,backward,left accordingly.
FORWARD= 0
BACKWARD= 180
RIGHT= 90
LEFT= 270
turn= ""
steps= 0
END_VALUE= "end"
#loop that runs until end value is entered
while turn != END_VALUE:
    turn= input("Next turn:")
    #checks if end value is reached
    if turn == END_VALUE: continue
    #decides how to change the heading (direction)
    if turn == "right":
        heading+= RIGHT
    elif turn == "left":
        heading+= LEFT #NOTE that it's the same as doing heading-= RIGHT

    heading%= 360  # 360=0,450=90 etc...
    steps= int(input("How many steps?"))

    #checks the direction to decide how to manipulate the position variable.
    #forward/backward: only y value is changed
    #left/right: only x value is changed
    if heading == FORWARD:
        position[1]+= steps
    elif heading == BACKWARD:
        position[1]-= steps
    elif heading == RIGHT:
        position[0]+= steps
    elif heading == LEFT:
        position[0]-= steps

gandalf_dest=["right","forward"]
#Checks what needed to be written on the output.
if position[0] >= 0:
    gandalf_dest[0]= "right"
else:
    gandalf_dest[0]= "left"

if position[1] >= 0:
    gandalf_dest[1]= "forward"
else:
    gandalf_dest[1]= "backward"

#changes the output to a positive number.
position[0]= abs(position[0])
position[1]= abs(position[1])

print("Gandalf should fly", position[0], "steps", gandalf_dest[0],\
                    "and" , position[1], "steps", gandalf_dest[1])
```