Contents

1 Basic Test Results

```
Starting tests...
Tue Nov 18 18:22:42 IST 2014
16c7e0ffa97b95c81285c0145c677b9af8c0cbc7  -


-rw-r--r-- ransha/stud     410 2014-11-18 17:28 findLargest.py
-rw-r--r-- ransha/stud     763 2014-11-18 17:30 findSecondSmallest.py
-rw-r--r-- ransha/stud     785 2014-11-18 17:52 totalWeight.py
-rw-r--r-- ransha/stud    1535 2014-11-18 17:56 twoDimensionalSeek.py
-rw-r--r-- ransha/stud     465 2014-11-18 17:44 ithElementValue.py
-rw-r--r-- ransha/stud     423 2014-11-18 17:27 decomposition.py
-rw-r--r-- ransha/stud     542 2014-11-18 18:12 binaryToDecimal.py
-rw-r--r-- ransha/stud     542 2014-11-18 18:11 decimalToBinary.py
-rw-r--r-- ransha/stud    2072 2014-11-18 18:17 README

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 mathematic 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:"))

from_base= 2
to_base= 10

num_out= 0
stop= False
pos= 0 #holds 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
    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:"))

from_base= 10
to_base= 2

num_out= 0
stop= False
pos= 0 #holds 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
    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

```
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

```
smallest=0
smallest_pos=0
second=0
second_pos=0
num_of_dancers=10
dancers=range(num_of_dancers)

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 this 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

    #checks which is the smallest EXCLUDING the smallest (second smallest)
    if age<second and age>smallest or second<=smallest:
        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.append(1)   #for example: orc[0] and orc[1] takes 1 arrow to kill.
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:")

while item != stop_value:
    #reads the input until a threshold or stop value is reached
    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 choose 4 directions: 0,90,180,270
          #for forward,right,backward,left accordingly.
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
    if turn=="right":
        heading+=90
    elif turn=="left":
        heading-=90

    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 == 0:         #forward case
        position[1]+= steps
    elif heading == 180:     #backward case
        position[1]-= steps
    elif heading == 90:      #right case
        position[0]+= steps
    elif heading == 270:     #left case
        position[0]-= steps

#Gandalf's direction:[right/left,forward/backward]
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])
```