New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python

Homework, Session 2

Procedure to follow when writing exercise solutions:

1. The exercises are based entirely on material from the slides.
2. Do not attempt to google a solution. If stuck, refer to the exercise solution as a guide, make sure it is clear to you, then try to recode from scratch.
3. Be completely clear on your code -- clarity means understanding the type and value of every object in the code, and what led up to its type and value.
4. Test as you write -- insert print statements that make each object type and value clear. (Be careful not to let your output jumble together and create more confusion.)
5. Remember that the primary goal is to understand how Python thinks, not just to make a solution work -- so hacking together or guessing your way to a solution subverts the purpose of this effort.

Procedure to follow when writing homework solutions:

1. The homework solutions are made up of techniques explored in the exercises**.**
2. Consider the overall objectives of the script and the steps you might take. Reflect on what you learned in the slides and exercises. Try to visualize in your mind how the data should be processed -- what objects would hold what values, how they would need to be converted or changed. This may not be easy at first, but it should be attempted. Eventually you'll learn how to think this way.
3. Write an outline, in **#** comments, of the steps the script will take, without writing any code. If you really feel at a loss as to what steps to take, feel free to review the "homework discussion" document, which includes an outline. (Please *do remove comments before submitting*.)
4. Referring closely to the reference exampes for the features you think you will need, compose code to satisfy the first step in the outline.
5. ***Test the code*** by printing the resulting object(s) and their type(s).
6. ***Don't proceed to the next step without testing the first step!*** Sometimes seeing the outcome of the first step can give you clues as to the proper next step, or cause you to modify your outline.
7. If you're not sure what to do, please email me before attempting to search on google. Google searches are problematic for beginners. We want to learn how to develop a strategy, not try to apply code solutions found on the web, which in the end will only take you so far -- that is, not very far in really understanding how to code.
8. Before submitting, review the Code Quality requirements and conform your code to proper style and code organization. I will quickly return solutions that ignore CQ. :)

EXERCISES

Ex. 2.1 Write a program that takes user input with **input()** and prints whatever the user typed.

Sample output:

```
please enter some text:  hey what up
you just wrote:  "hey what up"
```

Ex. 2.2 Again using **input()**, request the user to enter a number and test to see whether it is all digits (hint: use the **str isdigit()** method in an **if** test). Simply print a success message if all digits, a failure message if no. (Remember that **if** can be used with **else** to have Python do one thing or another depending on the result of the test.)

Sample program runs:

```
$ python ex2.3.py
please enter an integer:  hello
that was NOT an integer!

$ python ex2.3.py
please enter an integer:  55.5
that was NOT an integer!

$ python ex2.3.py
please enter an integer:  "55"
that was NOT an integer!

$ python ex2.3.py
please enter an integer:  55
THANKS FOR THE INTEGER!
```

Ex. 2.3 Based on the above program, place your whole program inside a **while True:** block. If the input is all digits (i.e., an integer), include the print statement indicating success, and follow it with a **break** statement to leave the loop. As you know, if you do not **break**, at the end of the block **while**'s "automatic return" will return to the top of the block automatically and your block will take **input()** again.

Sample program runs:

```
$ python ex2.4.py
please enter an integer:  hello
that was NOT an integer!
please enter an integer:  one?
that was NOT an integer!
please enter an integer:  5
THANKS FOR THE INTEGER!
```

Ex. 2.4 Use a while loop to count from 1 to 10. Before the loop begins, initialize an integer counter (**count**) to 1. The **while** test should test to see if the counter is less than or equal to 10. Inside the loop, "increment" **count** with **count = count + 1** (i.e., it will be incremented upon each iteration of the loop). Print each new value of the integer. (Note that the slides contain this solution, so your job is to be able to code a program like this from scratch. Repeat it until it is second nature if you can!)

Expected output:

```
1
2
3
4
5
6
7
8
9
10
```

Ex. 2.5 Similar to the above program, use a **while** loop and a counter to print "Happy birthday to you!" 10 times (do not print the integer count in this version).

Expected output:

```
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
```

Ex. 2.6 Modify the above program to take user input with **input()** and print the message that many times. (Hint: since we know that **input()** returns a string, the value will have to be converted to an **int** before it can be used in the **while** test with the loop count value.)

Sample program run:

```
$ python ex2.7.py
how many times should I greet you?  3
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!
```

Ex. 2.7 Modify the above program to test the user input to make sure it is a usable integer. (Hint: as we did earlier, use **str.isdigit()**). If the input is not all digits, you can use **exit('error message')** to exit the program.

Sample program runs:

```
$ python ex2.8.py
how many times should I greet you?  three
error:  please enter an integer          # i.e., exit('error: please...')

$ python ex2.8.py
how many times should I greet you?  3
Happy birthday to you!
Happy birthday to you!
Happy birthday to you!

$ python ex2.8.py
how many times should I greet you?  2
Happy birthday to you!
Happy birthday to you!
```

Ex. 2.8 Use the string **count()** method to count the number of times 'or' appears in the following text: **I am happy or sad or angry or mad or generous or stingy**.

Starter code: start your program with this code

```
msg = 'I am happy or sad or angry or mad or generous or stingy.'
```

Expected output:

```
$ python ex2.9.py
5
```

Ex. 2.9 Modifying the above program, take user input of a string to count, and then report the number of occurrences in the sentence.

Starter code: same as above

Sample program runs:

```
$ python ex2.10.py
please enter a string to search:  or
5

$ python ex2.10.py
please enter a string to search:  m
2

$ python ex2.10.py
please enter a string to search:  x
0
```

Ex. 2.10 Taking a user input substring, replace the substring in the string with 'x's

Starter code: same as above

Sample program runs:

```
$ python ex2.12.py
I am happy or sad or angry or mad or generous or stingy.
please enter a character to replace:  a
I xm hxppy or sxd or xngry or mxd or generous or stingy.

$ python ex2.12.py
I am happy or sad or angry or mad or generous or stingy.
please enter a character to replace:  or
I am happy x sad x angry x mad x generous x stingy.
```

HOMEWORK

Feel free to refer to the discussion document for further detail on how to proceed (including a code outline).

2.1 Count from 0 up to 100 by 2's (or 3's, or 4's…). Use a **while True:** loop and **input()** to take one input from the user: an integer "step" value. Test to ensure that the value is an integer value, and keep taking input until correct. Then print out all the values from 0 up to 100 (or less than 100 if the step value doesn't divide evenly into 100).

Sample program runs:

```
$ python homework_2-1.py

please enter an integer:  x
sorry, 'x' is not a valid integer



please enter an integer:  3
counting from 0 to 100 by 3's

0
3
6
9
12
15  ... (etc., up to 99)


$ python homework_2-1.py

please enter an integer: 25
counting from 0 to 100 by 25's
```

```
0
25
50
75
100
```

2.2 Sum a sequence of integers. As in previous, use a **while True:** loop to take a *positive* integer value (i.e., reject non-integers as well as values < 1. Now sum up all integers between 0 and the submitted value.

Sample program runs (user input in bold):

```
$ python dblaikie_2.2.py
please enter a positive integer:  hey
sorry, 'hey' is not a valid positive integer
please enter a positive integer:  0
sorry, '0' is not a valid positive integer
please enter a positive integer:  1
sum from 1 to 1 is 1

$ python dblaikie_2.2.py
please enter a positive integer:  2
sum from 1 to 2 is 3

$ python dblaikie_2.2.py
please enter a positive integer:  3
sum from 1 to 3 is 6

$ python dblaikie_2.2.py
please enter a positive integer:  4
sum from 1 to 4 is 10

$ python dblaikie_2.2.py
please enter a positive integer:  5
sum from 1 to 5 is 15
```

2.3 Text replacement utility. Complete this program so that it can replace any word or phrase in a text file, and then output the file text after the replacements have been made. It can also report the number of replacements that were made.

Take two strings using **input()**: target text and replace text. The supplied starter code will open and read the file out to a string; complete the program by replacing the target text with the replace text, and then report on the number of replacements made.

If you would like to apply this to a file, comment out the last line in this sample and uncomment the line before it. Add an **input()** statement to the other two that prompts for the filename.

Sample run:

```
python homework_2-3.py

please enter search text:  yourself
please enter replace text:  your Corgi

And since you know you cannot see your Corgi,
so well as by reflection, I, your glass,
will modestly discover to your Corgi,
that of your Corgi which you yet know not of.

3 replacements made.
```

Starter code:

```
sample_text = """And since you know you cannot see yourself,
so well as by reflection, I, your glass,
will modestly discover to yourself,
that of yourself which you yet know not of."""

def read_file(fname):
    try:
        text = open(fname).read()
    except IOError:
        print 'file {} not found or could not be read.  '.format(fname)
        print 'Using sample text instead.'
        text = sample_text
    return text

# to use a file instead of sample text, add a comment mark (#)
# to start of next line, and remove comment mark from the line after

# text = read_file(input('please enter a filename:  '))
text = sample_text

# [YOUR CODE GOES HERE]
```

[continued]

SUPPLEMENTARY (EXTRA CREDIT) EXERCISES

2.4 Reverse Number Guessor. Create a program that selects a random number between 1 and 100 (see starter code below), but keeps it secret. It then tells the user whether his or her guess is correct, is greater than or is less than the user's guess.

Starting Code:

```
import random

number_to_guess = random.randint(1,10)
```

Sample program run (user input in bold):

```
I am thinking of a number from 1 to 100.   Try to guess it and I'll give
you hints about it.

Your guess?  (q to quit):  50
your guess is LOWER than the number I'm thinking of.

Your guess?  (q to quit):  75
your guess is LOWER than the number I'm thinking of.

Your guess?  (q to quit):  87
your guess is HIGHER than the number I'm thinking of.

Your guess?  (q to quit):  81
you got it!  You guessed it in 4 tries.
```

2.5 Number Guessor. Write a program that attempts to guess a number from 1 to 100 that the user has chosen and is keeping secret. To guess the number, the program asks the user a series of guesses and questions, narrowing down the possibilities by half each time. The program will:

a. Ask the user to think of a number from 1 to 100
b. Ask the user if the number is 50 (i.e., halfway between 1 and 100
c. If the user answers "no", ask the user if the number is higher or lower than 50
d. If lower, ask the user if the number is 25 (i.e., halfway between 1 and 50)
e. If higher, ask the user if the number is 75 (i.e., halfway between 50 and 100)
f. Return to question c -- if the guess is incorrect, ask the user if the number is higher or lower than 25 (or 75)
g. Continue halving the possibiltiies and guessing the number until the user reports that the guess is correct.

[continued]

Sample program run (user input in bold):

```
Think of a number between 100 and 0, and I will try to guess it.  Hit
[Enter] to start.
is it 50 (yes/no/quit)?  no
is it higher or lower than 50?  higher
is it 76 (yes/no/quit)?  no
is it higher or lower than 76?  lower
is it 63 (yes/no/quit)?  no
is it higher or lower than 63?  lower
is it 57 (yes/no/quit)?  no
is it higher or lower than 57?  lower
is it 54 (yes/no/quit)?  no
is it higher or lower than 54?  lower
is it 52 (yes/no/quit)?  no
is it higher or lower than 52?  lower
is it 51 (yes/no/quit)?  yes
I knew I could do it!
```

2.6 Find prime numbers in a range of numbers. Ask the user for an integer, and display all prime numbers between 2 and the user's number.

Sample program run (user input in bold):

```
$ python primes.py

* Prime Numbers *
Please enter max integer:  5
2
3
5

$ python primes.py

* Prime Numbers *
Please enter max integer:  20
2
3
5
7
11
13
19
```

2.7 Chicken, Fox and Grain game. This is rather lengthy assignment using **while**, **break** and **continue**, as well as numerous **if** tests comparing strings and numbers. (In fact, it requires a great deal more if/then/else than might be needed if we were to use some upcoming Python features. But, it provides an interesting challenge using this week's features.)

The program is an implementation of a well-known logic puzzle:

A man has to transport a fox, a chicken and a sack of corn across a river using a rowboat, but the rowboat has room for only one of the items besides himself. Therefore when crossing the river he usually has to leave two of the items alone on one of the shores. The problem is that if he leaves the fox alone with the chicken, the fox will eat the chicken, and if he leaves the chicken alone with the grain, the chicken will eat the grain. How can he move all three items across the river without losing any of them?

Sample program runs (user input in bold):

```
$ python david_blaikie_2.7.py


**THE FOX, THE CHICKEN AND THE GRAIN**

You are on the east bank of a river.
You must transport a fox, a chicken
and a bag of grain to the west bank.
However, you can only carry one item
at a time.  Good luck!

you now are on the east bank.
the chicken is on the east bank.
the fox is on the east bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  xx

sorry, I don't recognize 'xx'.  Try again.

you now are on the east bank.
the chicken is on the east bank.
the fox is on the east bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  grain

oops, the fox got the chicken!


$ python david_blaikie_2.7.py

**THE FOX, THE CHICKEN AND THE GRAIN**

You are on the east bank of a river.
You must transport a fox, a chicken
and a bag of grain to the west bank.
However, you can only carry one item
at a time.  Good luck!
```

```
you now are on the east bank.
the chicken is on the east bank.
the fox is on the east bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  fox

oops, the chicken got the grain!


$ python david_blaikie_2.7.py


**THE FOX, THE CHICKEN AND THE GRAIN**

You are on the east bank of a river.
You must transport a fox, a chicken
and a bag of grain to the west bank.
However, you can only carry one item
at a time.  Good luck!

you now are on the east bank.
the chicken is on the east bank.
the fox is on the east bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  chicken

...carrying the chicken to the west bank...

you now are on the west bank.
the chicken is on the west bank.
the fox is on the east bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  [pressed Enter, meaning take back nothing]

...traveling to the east bank...

you now are on the east bank.
the chicken is on the west bank.
the fox is on the east bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  fox

...carrying the fox to the west bank...

you now are on the west bank.
the chicken is on the west bank.
the fox is the west bank.
the grain is on the east bank.

What would you carry in the seat of the boat ([Enter] for nothing, q to
quit)?  [pressed Enter, meaning take back nothing]

oops, the fox got the chicken!
```