SY201 Six Week Exam Study Guide
09/19/18

General Guidance

- The exam is designed to be 1-hour long (approximately)

- It will be given during our regular 2-hr lab block on Tuesday, September 25th. You will be given the entire 2-hrs to complete the exam.

- The exam is closed book and closed notes, but you are allowed to bring a single 8.5 x 11 inch piece of paper to class with any *handwritten* notes on it you wish. Whatever you write must be in your own handwriting (no printouts or photocopies), and you must submit the page with your exam when you turn it in. These sheets you make will be returned to you after the exam has been graded, so you can build on them for subsequent exams.

- If an ASCII table is needed for the exam, it will be provided. You do not need to memorize or write down any of the ASCII codes on your notes sheet.

- *Important: You may not share any information about the exam with anyone until the exam is graded and returned to you.*

Topics That May Be Covered On The Exam

1. Python basics:
    a. Variables, assignment statements
    b. Type conversions / casts
    c. Basic operators (`+`, `-`, `/`, `//`, `*`, `%`, etc.)
    d. Basic comparisons (`<`, `>`, `==`, `!=`, `<=`, `>=`)
    e. Arithmetic order of precedence
    f. Basic input/output
2. Conditional statements:
    a. `if-elif-else` statements
    b. `and` / `or` precedence rules
3. Loops:
    a. `for` loops
    b. `while` loops
4. Files:
    a. `read`, `write`, `open`, `close`
5. Input validation:
    a. `try – except`
    b. Dealing with out-of-range values

Types of Questions

1. Tracing programs
2. Writing short programs
3. Correcting programs or making edits to programs to change functionality
4. There may be a few short answers.

The best preparation is to practice writing short programs!

Review Problems

Work through all problems on paper and pencil, then type them into a `.py` file and run them to check your answers.

1. Write down the output of the following program:

```
x = 15
i = 1
while i <= x:
  print(i)
  i += 4
```

2. Write a program that reads a number `n` (e.g. `7`) from the user, and prints a multiplication table for that number. The number entered must be between `1` and `12`.

- *Implement exception handling and data validation to ensure you receive a valid integer from the user and that integer is in the allowable range.*
- *For practice, use formatted output to display your table in columns, with each entry right justified.*

3. Write a program that implements a four-function calculator. The program should read the operator first (`+`, `-`, `*`, `/`), and then it should read two numbers that the operator should be applied to. The program should repeat until the user types "`Q`" for the operator.

You do not need to implement exception handling or data validation. You may assume that valid operators and valid integers will be entered.

Example:

```
> +
> 89
> 40
The answer is 129
> -
> 65
> 20
The answer is 45
> Q
```

4. Write a program that reads a number `n` from the user, and prints the first `n` Fibonacci numbers.

- *You may need to Google Fibonacci numbers…*
- *You do not need to implement exception handling or data validation. You may assume that valid integers `>= 1` will be entered.*

```
Enter n: 9
Numbers are 0 1 1 2 3 5 8 13 21
```

5. Write a program that reads a number `n` (where `n` is greater than `1`) from the user and determines if `n` is a prime number.

- *Implement exception handling and data validation to ensure you receive a valid integer from the user and that the integer is `> 1`.*
- *When we're talking about primes, the number `2` is an edge case. How do you handle it?*

```
Enter n: 2
Your number is prime

Enter n: 1001
Your number is not prime

Enter n: 1000033
Your number is prime
```

6. The program below is a student's attempt to print all odd numbers between `1` and `100`. The program doesn't work as specified. Explain why it doesn't work and then adjust the program to fix it.

```
i = 1
while i != 100:
  print(i)
  i += 2
```

7. Write a program that reads numbers from a file (one number per line, no decimal points) and finds the minimum and maximum numbers in the file, and prints them to the screen.

8. Write a program that accepts a string from the user and then prints both the original string and the original string reversed:

Example:

```
Enter a string: This is a test of my excellent program.
This is a test of my excellent program.
.margorp tnellecxe ym fo tset a si sihT
```