MIT AITI
Python Software Development

Lab 03: Mooooooooore Looooooooooops

Part I: More Practice with While Loops

1. Create a Python file named Lab03_1.py that displays the first fifty *prime* numbers in five lines (each line contains 10 numbers). An integer greater than 1 is prime if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime but 4, 6, 8, and 9 are not prime. The output of your program should look like this:

```
The first 50 prime numbers are
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
```

   You need to write a loop and test whether each new number is prime. Declare a variable count to store the number of primes encountered so far. If the number is prime, increment count by 1. When count is greater than 50, exit the loop.

   Hint: To test whether a number n is prime, check if the number is divisible by 2, 3, 4, up to the square root of n. If a divisor is found, the number is not prime. For example, for the number 17, you need to test whether each of 2, 3 and 4 are divisors of 17. Since none are divisors, 17 is prime. If a number is not prime, once you find the first divisor, you should not keep checking for additional divisors

2. Create a Python file named Lab03_2.py. Use nested loops to print out each of the following patterns. Create separate sections inside of the file for each pattern, and put a comment before each saying Lab03_2a, Lab03_2b, Lab03_2c, Lab03_2d, and Lab03_2e.

```
a. 1
   1 2
   1 2 3
   1 2 3 4
   1 2 3 4 5
   1 2 3 4 5 6

b. 1 2 3 4 5 6
   1 2 3 4 5
   1 2 3 4
   1 2 3
   1 2
   1
```

c.
```
          1
        2 1
      3 2 1
    4 3 2 1
  5 4 3 2 1
6 5 4 3 2 1
```

d.
```
1 2 3 4 5 6
  1 2 3 4 5
    1 2 3 4
      1 2 3
        1 2
          1
```

e.
```
     1
    212
   32123
  4321234
 543212345
```

Part II: Secret Messages

The goal of this exercise is to write a cyclic cipher to encrypt messages. This type of cipher was used by Julius Ceasar to communicate with his generals. It is very simple to generate but it can actually be easily broken and does not provide the security one would hope for.

The key idea behind the Ceasar cipher is to replace each letter by a letter some fixed number of positions down the alphabet. For example, if we want to create a cipher shifting by 3, you will get the following mapping:

```
Plain:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:  DEFGHIJKLMNOPQRSTUVWXYZABC
```

To be able to generate the cipher above, we need to understand a little bit about how text is represented inside the computer. Each character has a numerical value and one of the standard encodings is ASCII (American Standard Code for Information Interchange). It is a mapping between the numerical value and the character graphic. For example, the ascii value of 'A' is 65 and the ascii

value of 'a' is 97. To convert between the ascii code and the character value in Python, you can use the following code:

```
letter = 'a'
# converts a letter to ascii code
ascii_code = ord(letter)

# convers ascii code to a letter
letter_res = chr(ascii_code)

print ascii_code, letter_res
```

Start small. Do not try to implement the entire program at once. Break the program into parts as follows:

1. Create a file called cipher.py. Start your program by asking the user for a phrase to encode and the shift value. Then create a new string that contains the original phrase value using a for loop as follows:

```
encoded_phrase = ''

for c in phrase:
   encoded_phrase = encoded_phrase + c
```

2. Now modify the program above to replace all the alphabetic characters with 'x'. For example:

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is:  Xxxxxx! Xxxxxx!
```

We are going to apply the cipher only to the alphabetic characters and we will ignore the others.

3. Now modify your code, so that it produces the encoded string using the cyclic cipher with the shift value entered by the user. Let's see how one might do a cyclic shift. Let's say we have the sequence: 012345

If we use a shift value of 4 and just shift all the numbers, the result will be: 456789

We want the values of the numbers to remain between 0 and 5. To do this we will use the modulus operator. The expression x%y will return a number in the range 0 to y-1 inclusive, e.g. 4%6 = 4, 6%6 = 0, 7%6 =1. Thus the result of the operation will be:

```
450123
```

Hint: Note that the ascii value of 'A' is 65 and 'a' is 97, not 0. So you will have to think how to use the modulus operator to achieve the desired result. Apply the cipher separately to the upper and lower case letters.

Here is what you program should output:

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is:  Qechec! Qechec!
```