

Midterm Feedback


CMPT 120 D300 Spring 2024

Reminder - 1

- When we are given *Sample Runs* as part of **Step 1 – Problem Statement** we must implement our solution such that our program produces **exactly** what we see in these *Sample Runs*:
 - Same words,
same spacing,
same layout
 - Our program
must produce the
same result
displayed in the
same way

```
Lunch Menu:
Sandwich -> $5
Salad -> $4
Juice -> $2
Apple -> $1
For lunch, would you like ...
a Sandwich (y/n)? y
a Salad (y/n)? n
a Juice (y/n)? n
a Apple (y/n)? Y
Total for your lunch is $6.
```

Reminder - 2

- **No repeated code**
- When the code that is being repeated is very similar from one repetition to the next
-> see example 
- In order to avoid repeated code, we can use Python *building blocks* such as
 - sequences (strings, lists or tuples) and indexing sequences
 - loops

```
for aChar in normalMsg:
    if aChar == "a":
        strOfEncryptedChars += "n"

    elif aChar == "b":
        strOfEncryptedChars += "o"

    elif aChar == "c":
        strOfEncryptedChars += "p"

    elif aChar == "d":
        strOfEncryptedChars += "q"

    elif aChar == "e":
        strOfEncryptedChars += "r"

    elif aChar == "f":
        strOfEncryptedChars += "s"

    elif aChar == "g":
        strOfEncryptedChars += "t"

    elif aChar == "h":
        strOfEncryptedChars += "u"

    elif aChar == "i":
        strOfEncryptedChars += "v"

    elif aChar == "j":
        strOfEncryptedChars += "w"

    elif aChar == "k":
        strOfEncryptedChars += "x"

    elif aChar == "l":
        strOfEncryptedChars += "y"

    elif aChar == "m":
        strOfEncryptedChars += "z"

    elif aChar == "n":
        strOfEncryptedChars += "a"
```

Reminder - 3

- In order to avoid repeated code, we can use Python *building blocks* such as
 - sequences (strings, lists or tuples) and indexing sequences
 - loops

```
def rotate(aMsg):  
    """encrypts/decrypts a message using the substitution algorithm  
    called "ROT13" and returns its result."""  
  
    # Start with an empty string (accumulator)  
    resultMsg = ""  
  
    keyL = "abcdefghijklmnopqrstuvwxyz"  
    keyU = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
  
    # Consider each character in the message  
    for aChar in aMsg:  
  
        #originalChar = aChar  
        if keyU.find(aChar) >= 0:  
            resultMsg += keyU[(keyU.find(aChar) + 13) % 26]  
        elif keyL.find(aChar) >= 0:  
            resultMsg += keyL[(keyL.find(aChar) + 13) % 26]  
        else:  
            resultMsg += aChar  
  
    # Return the resulting mmessage  
    return resultMsg
```

Reminder - 4

- No hard coded values!
 - Again, use Python variables (such as lists)

```
# Set up the variables:  
# lunch items, their associated cost, and a running total variable  
menu = ["Sandwich", "Salad", "Juice", "Apple"]  
cost = [5, 4, 2, 1]  
total = 0
```

```
for item in range(len(menu)) :  
    response = input(f"a {menu[item]} (y/n)? ").lower()  
    # If the user says yes ...  
    if response == 'y': # in ["Y", "y"]:  
        # Add the cost of the item to the running total  
        total += cost[item]
```

- No **while true:** (**break**)
- Always think *function*!
 - Any code fragment that has a specific purposes (functionality) can become a function

Reminder - 5

- **You must follow all of these reminders even if the question does not tell you to do so!**
 - Keeping all these in mind when developing software is what it means to be a **software developer**

Note that you are not told which Python statements to use in order to implement this program. You need to use your own judgment and your software development experience acquired so far in this course to make the proper decisions.¶