IS111 – Lab 4
More on conditions, Strings and for-Loops

Instructions

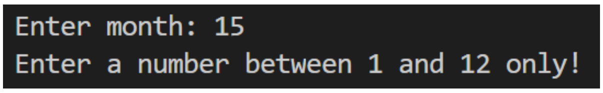
You should attempt the questions using Visual Studio Code. Download and extract the files from the **Lab4_starting_code.zip** and save the files. You should obtain several *.py files to be used for some of the lab questions.

Q1: Number of Days in a Month [*]

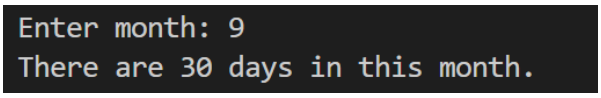
Create `q1.py` and write a piece of Python code that prompts the user for a month number (1 for January, 2 for February, etc.) The program then displays the number of days in that month. You can ignore leap years and assume that February always has 28 days.

You can assume that the user is always going to enter an integer value. If the value is outside of the range between 1 and 12, you should display an error message.

Two sample runs of the code look as follows:



```
Enter month: 15
Enter a number between 1 and 12 only!
```



```
Enter month: 9
There are 30 days in this month.
```

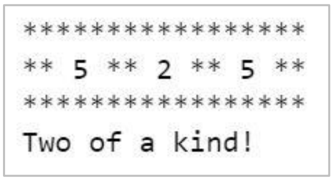
Q2: Jackpot [*]

You are given a piece of code in `Q2.py` that generates and displays three random numbers between 0 and 9 (both included). This is to simulate a Jackpot machine.

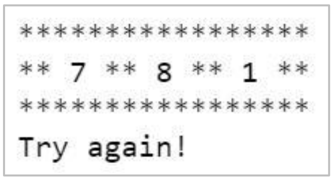
Complete the program to display one of the following messages:

- **"Try again!"** if the three numbers are all different.
- **"Two of a kind!"** if there are two numbers that are the same.
- **"Jackpot!"** if all three numbers are the same.

Three sample runs of the code are shown below:



```
****************
** 5 ** 2 ** 5 **
****************
Two of a kind!
```



```
****************
** 7 ** 8 ** 1 **
****************
Try again!
```

Q3: Mobile Plans [**]

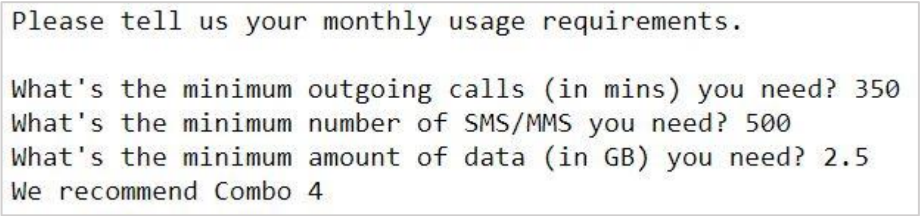
A telco company offers the following postpaid mobile plans:

plan	monthly charge	outgoing calls (mins)	local SMS/MMS	local data
Combo 1	$25.50	80	300	100MB
Combo 2	$39.50	200	500	2GB
Combo 3	$59.50	300	1000	3GB
Combo 4	$79.50	400	1500	4GB
Combo 5	$109.50	800	2000	10GB

Each row shows the monthly charge of each plan and the maximum amount of usage covered by the plan.

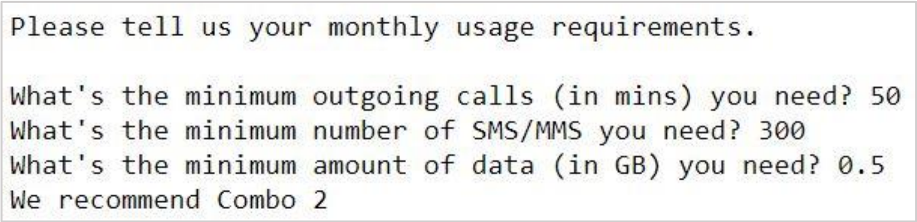
Create a new file `q3.py`, prompt the user for his/her requirements and help him/her find the most cost-effective postpaid plan. The recommended plan should satisfy all his/her needs.

Three sample runs of the program can be found below:



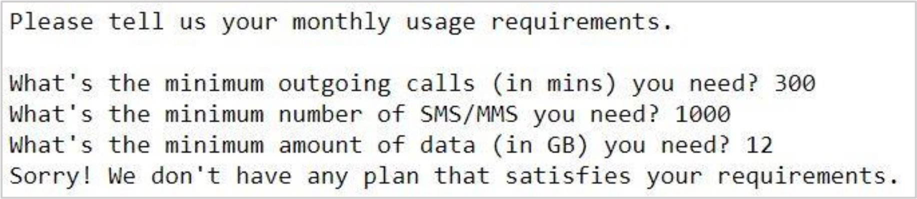
```
Please tell us your monthly usage requirements.

What's the minimum outgoing calls (in mins) you need? 350
What's the minimum number of SMS/MMS you need? 500
What's the minimum amount of data (in GB) you need? 2.5
We recommend Combo 4
```



```
Please tell us your monthly usage requirements.

What's the minimum outgoing calls (in mins) you need? 50
What's the minimum number of SMS/MMS you need? 300
What's the minimum amount of data (in GB) you need? 0.5
We recommend Combo 2
```



```
Please tell us your monthly usage requirements.

What's the minimum outgoing calls (in mins) you need? 300
What's the minimum number of SMS/MMS you need? 1000
What's the minimum amount of data (in GB) you need? 12
Sorry! We don't have any plan that satisfies your requirements.
```

Q4: Message Encryption¶

Part I [**]

Create a new file `q4.py` and write a program that encrypts a message as follows:

- All the occurrences of letter 'a' in the message are replaced by 'e'
- All the occurrences of letter 'e' in the message are replaced by 'i'
- All the occurrences of letter 'i' in the message are replaced by 'o'
- All the occurrences of letter 'o' in the messsge are replaced by 'u'
- All the occurrences of letter 'u' in the messsge are replaced by 'a'

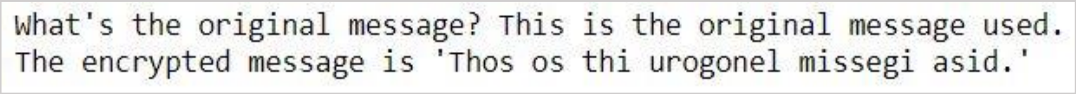
For example, suppose the original message is

```
    This is the original message used.
```

Then the encrypted message becomes

```
    Thos os thi urogonel missegi asid.
```

A sample run of the program is shown below:



```
What's the original message? This is the original message used.
The encrypted message is 'Thos os thi urogonel missegi asid.'
```

Part II [**]

Can you modify the program above such that the message is not only encrypted as described above but also reversed character by character?

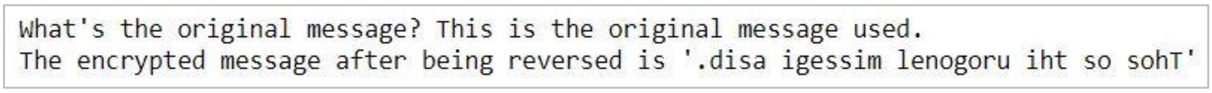
For example, suppose the original message is

```
    This is the original message used.
```

Then the final output is the following:

```
    .disa igessim lenogoru iht so sohT
```

A sample run of the modified program is as follows:



```
What's the original message? This is the original message used.
The encrypted message after being reversed is '.disa igessim lenogoru iht so sohT'
```

Q5: Message Padding [***]

a) Create `q5.py` and define a function called `pad_message`.

The function takes in two parameters:

- msg (type: str): This is a string representing a message.
- width (type: int): This is a number indicating the final length of the returned padded message.

The function **returns** a new string whose length is `width`, the second parameter. If the original message is shorter than `width`, then **spaces are padded** to the left of the message. If the original message is longer than `width`, then the original message is **truncated** on the right hand side to fit into the specified `width`.

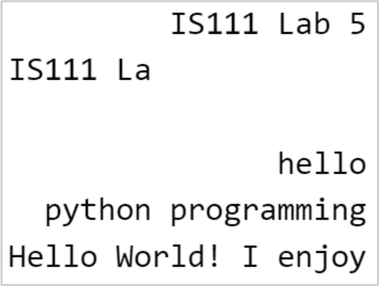
For example,

- If the message is `"IS111 Lab 5"`, which contains 11 characters, and if the specified width is 20, then this function returns `"         IS111 Lab 5"` (ie. padded with 9 preceding spaces)
- If the message is the same but the specified width is 8, then this function returns `"IS111 La"`, which consists of the first 8 characters of the original message.

b) In the same file, add the following codes after your written `pad_message` function definition:

```
print(pad_message("IS111 Lab 5", 20))
print(pad_message("IS111 Lab 5", 8))
print(pad_message("hello", 20))
print(pad_message("python programming", 20))
print(pad_message("Hello World! I enjoy programming in Python.",20))
```

The expected output is as follows:



```
         IS111 Lab 5
IS111 La

               hello
  python programming
Hello World! I enjoy
```

Q6: Shopping Cart

Part A [**]

Recall that in Lab 2, there was a question related to discounted items. In that exercise, you have implemented the `calculate_price_after_discount` function. This function has now been provided for you in `Q6.py`.

In `Q6.py`, make use of the function `calculate_price_after_discount` and write a program that helps a customer check out her items in her shopping cart at a self-checkout counter in a supermarket.

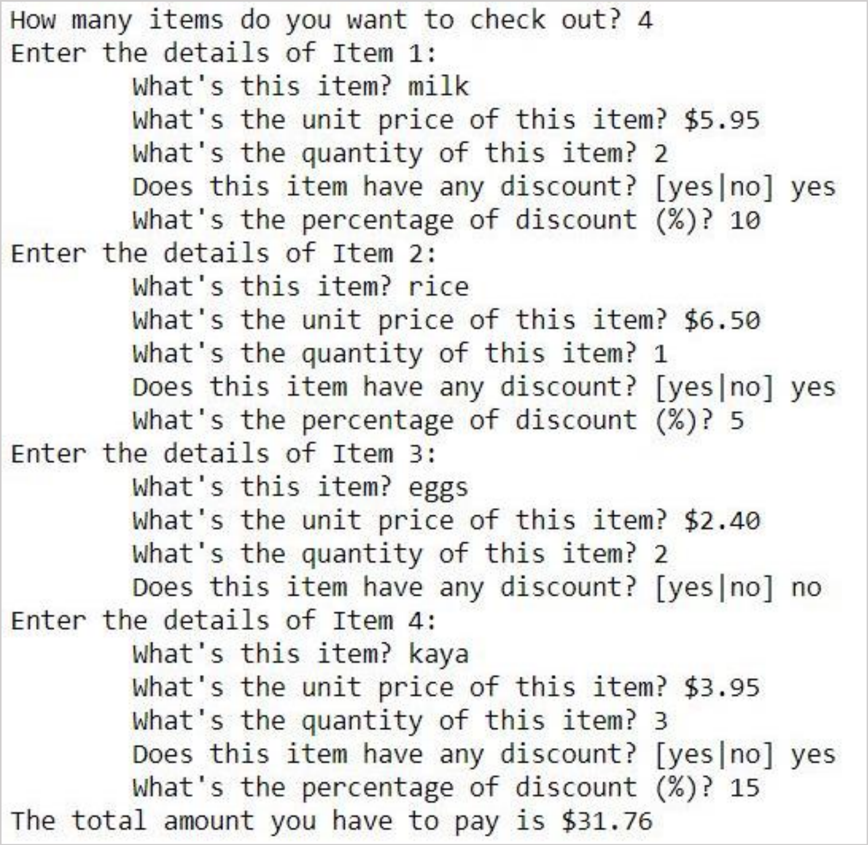
The program first asks the customer how many items the customer wants to check out. The program then prompts the user for the details of each item, including its unit price, its quantity being purchased, whether or not it has a discount, and how much the discount is if there's a discount.

For example, suppose the customer's shopping cart contains the following items:

]

description	unit_price	quantity	discount
milk	$5.95	2	10%
rice	$6.50	1	5%
eggs	$2.40	2	0%
kaya	$3.95	3	15%

The program shall run as follows:



```
How many items do you want to check out? 4
Enter the details of Item 1:
        What's this item? milk
        What's the unit price of this item? $5.95
        What's the quantity of this item? 2
        Does this item have any discount? [yes|no] yes
        What's the percentage of discount (%)? 10
Enter the details of Item 2:
        What's this item? rice
        What's the unit price of this item? $6.50
        What's the quantity of this item? 1
        Does this item have any discount? [yes|no] yes
        What's the percentage of discount (%)? 5
Enter the details of Item 3:
        What's this item? eggs
        What's the unit price of this item? $2.40
        What's the quantity of this item? 2
        Does this item have any discount? [yes|no] no
Enter the details of Item 4:
        What's this item? kaya
        What's the unit price of this item? $3.95
        What's the quantity of this item? 3
        Does this item have any discount? [yes|no] yes
        What's the percentage of discount (%)? 15
The total amount you have to pay is $31.76
```

Note:

- The shopping cart may change and your program should be able to handle a different shopping cart. i.e., you should not hard code any information such as the number of items, unit prices, quantities, discount rates, etc.
- You may wish to use `round(x, 2)` to round the number `x` to contain two decimal places after the decimal point when you display the final total amount.

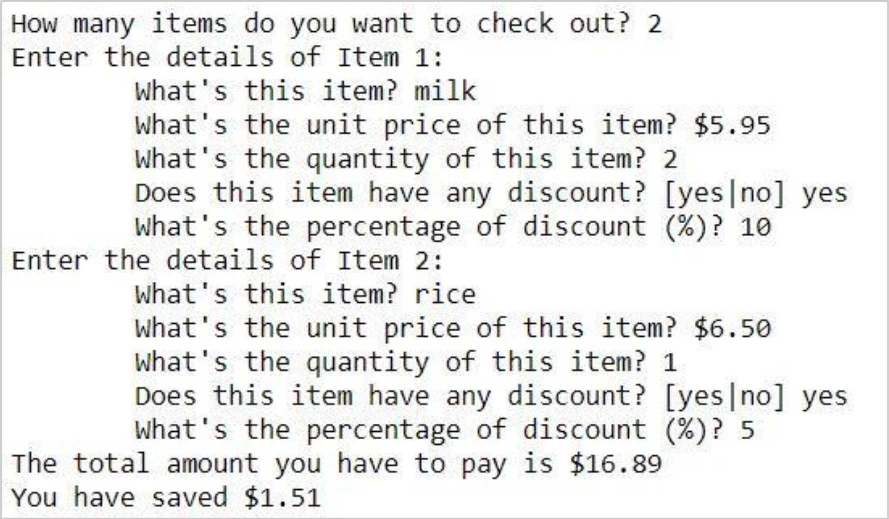
Part B [*]**

Copy your Part A code and use it for Part B. Amend it to display the total amount of savings the customer has got, i.e., the difference between the original total price and the total price after discount.

Note:

- You may wish to write a second version of the function we give you earlier (call it `calculate_price_after_discount2`) such that it returns both the price after discount and the savings.

A sample run of the modified program can be found below:



```
How many items do you want to check out? 2
Enter the details of Item 1:
        What's this item? milk
        What's the unit price of this item? $5.95
        What's the quantity of this item? 2
        Does this item have any discount? [yes|no] yes
        What's the percentage of discount (%)? 10
Enter the details of Item 2:
        What's this item? rice
        What's the unit price of this item? $6.50
        What's the quantity of this item? 1
        Does this item have any discount? [yes|no] yes
        What's the percentage of discount (%)? 5
The total amount you have to pay is $16.89
You have saved $1.51
```

Q7: Count Words [**]

In this question we'll count the number of "a" and "an" in a piece of text.

Note: For this question, you are NOT allowed to use the `count()` method of string.

In the file `q7.py` that has been provided for you, define the following functions:

a) Write a function called `count_a` that takes in a piece of text (as a string) and returns the number of times the word "a" occurs in the text. You do not need to handle uppercase "A". You can assume that each time the word "a" occurs, both its previous character and its next character are a space.

   For example, `count_a("I have a room with a window, a desk and a chair.")` should return `4`.

b) Write a function called `count_an` that takes in a piece of text (as a string) and returns the number of times the word "an" occurs in the text. You do not need to handle uppercase "An". You can assume that each time the word "an" occurs, its previous character and its next character are both a space.

   For example, `count_an("Every day I have an egg, an apple and a banana for breakfast.")` should return `2`.

Use the provided `count_words_test.py` file to test your code. You should not modify `count_words_test.py`.

Q8: Fibonacci Numbers [***]

Refer to the following link to understand Fibonacci numbers:
https://en.wikipedia.org/wiki/Fibonacci_number

Essentially, it is a sequence of numbers where each number is equal to the sum of its previous two numbers in the sequence.

Create `q8.py` and write a function called `display_fibonacci()`. This function takes in an integer `n` (greater or equal to 3). It **prints out** the first `n` Fibonacci numbers, starting from 1. The function doesn't return anything.

For example:

`display_fibonacci(3)` prints out the following output: `1 1 2`

`display_fibonacci(5)` prints out the following output: `1 1 2 3 5`

`display_fibonacci(10)` prints out the following output: `1 1 2 3 5 8 13 21 34 55`

Use the provided file `fibonacci_test.py` to test your implemented function. You should not modify `fibonacci_test.py.`

Note: In Week 3's extra in-class exercises we had the same question. However, for this Lab 4, you should use `for`-loops to solve the problem (and recursive functions are not needed).