Practical 05 - Coding Checkpoint 1

Submit your prac work via LearnJCU each week by the due date.

Do the numbered questions in a file called `programs.py`

Input, Processing, Output

1. Write a program that calculates a new value based on an original value and a (percentage-like) increase or decrease.
   Example:

- Original: 100, Change: 0.05, Result: 105
- Original: 50.5, Change: -0.1, Result: 45.45

Decision Structures

2. Ask the user for the time of day (0-23 hours) and if they are *coming* or *going*.
   Then print a statement about their situation like:

```
It is before noon and you are coming. Hi!
It is after noon and you are going. Bye!
```

Note:

The intention with this exercise is for you to see that what the user inputs for these 2 questions are unrelated.
You can store the results to be printed in string variables and then print a *SINGLE* output using those variables.
You do *NOT* want to handle this with 4 separate print statements; one for each possible outcome.
Why not?
Because in *ALL* cases, you print one thing. So, only print once... but customise what you print based on the 2 user inputs.
Also, there's no such thing as "noon" :) As soon as you hit 12, it's "after noon".

3. Coffee orders made simple.

- Ask the user for white or black coffee
- Ask for their chosen size: small, medium or large
- Then print the cost: For Black, Small = $3, Medium = $4, Large = $5. White coffee costs $1 more per size.
- If a user makes a mistake with either part of their order, just pick the more expensive option in each case :)
  E.g. a purple small coffee would cost $4 and a black tiny coffee would cost $5.
  (This can be done by thinking about the default case for each user input.)
  Note: You can use the string methods `.upper()` or `.lower()` to make your string comparisons case insensitive, e.g.

```
string = input("Prompt: ").lower()
if string == "string":
    print("It matches without capitals")
```

Repetition Structures

4. Add error-checking to the coffee order program so you repeat until you get valid inputs.

5. Write a program to ask the user for a low value and a high value, then print all of the integers between those values inclusive and show the total of those numbers.
   Example, if the inputs were `10` and `20`, you would print:

   10 11 12 13 14 15 16 17 18 19 20 totals: 165

All Together Now

File: `happy_products.py`
Happy Products is an exciting new company that provides happy products to people who use their awesome new software...
But it's up to you to write this awesome new software.

They want a menu with the options:

- Instructions
- Calculate
- Quit

When you calculate, you are asked how many products you want to buy and at what price (see how exciting this is?).
If you buy 0-5 items, they're full price, over 5 items and each one is 10% off!
Examples:

- 5 x $10 products = $50
- 6 x $50 products = $270

Under 0 items is invalid and so are negative/$0 products, so keep prompting the user until they enter valid inputs.
The menu should also handle invalid choices by printing "Invalid choice".

The instructions are:

Enter the number of products you want to buy and your chosen price. If you buy 0-5 items, they're full price, over 5 items and each one is 10% off!

Sample Output

Here is some sample output from a run of the program:

```
Menu:
(I)nstructions
(C)alculate
(Q)uit
Choice: w
```

```
Invalid choice
Menu:
(I)nstructions
(C)alculate
(Q)uit
Choice: I
Enter the number of products you want to buy and your chosen price.
If you buy 0-5 items, they're full price, over 5 items and each one is 10%
off!
Menu:
(I)nstructions
(C)alculate
(Q)uit
Choice: c
Number of products: -1
Invalid input
Number of products: 0
Price: 450.32
0 x $450.32 products = $0.00
Menu:
(I)nstructions
(C)alculate
(Q)uit
Choice: C
Number of products: 3
Price: 12.34
3 x $12.34 products = $37.02
Menu:
(I)nstructions
(C)alculate
(Q)uit
Choice: c
Number of products: 6
Price: 100
6 x $100.00 products = $540.00
Menu:
(I)nstructions
(C)alculate
(Q)uit
Choice: q
Farewell
```

Deliverables

This section summarises the expectations for marking in this practical.

- `programs.py` with all of the numbered checkpoint questions:

1. Percentage program (I, P, O)
2. Time of day (Decisions)
3. Coffee orders (Decisions)
4. Coffee order error-checking (Repetitions)
5. Low-high printing (Repetitions)

- `happy_products.py` (The All Together Now question)