class exercises CE

Selection Statements

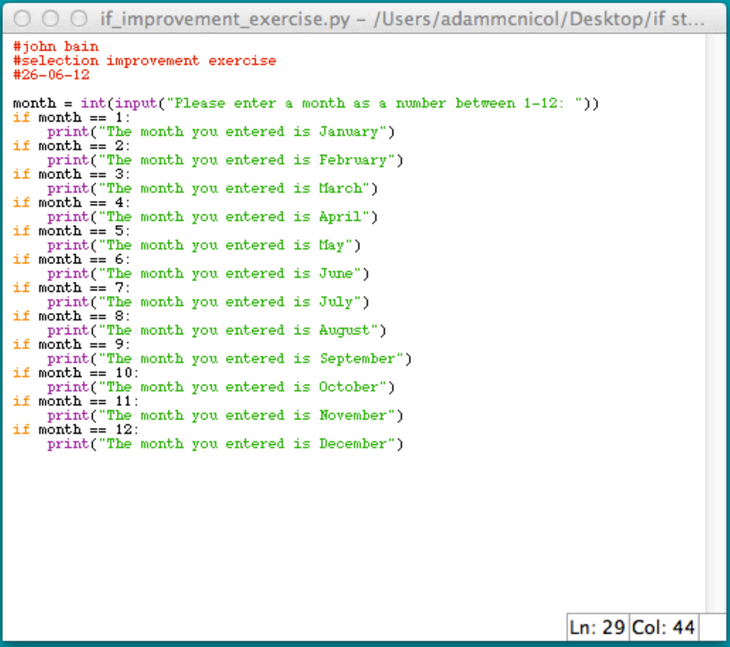
These tasks are designed to help you practice the knowledge and skills you have developed. There are exercises to help you revise and develop your understanding and also to challenge you further.

Improving a piece of code

Sometimes you may need to improve code produced by other people. Their code may contain errors or have a poor style. It is important that you can interpret and revise the work of others.

Task One

Open the **python** file called **if_improvement_exercise.py** and **refactor** the code so that it uses a more appropriate style and provides appropriate feedback when an invalid integer has been entered.


if_improvement_exercise.py – /Users/adammcnicol/Desktop/if st...

```
#john bain
#selection improvement exercise
#26-06-12

month = int(input("Please enter a month as a number between 1-12: "))
if month == 1:
    print("The month you entered is January")
if month == 2:
    print("The month you entered is February")
if month == 3:
    print("The month you entered is March")
if month == 4:
    print("The month you entered is April")
if month == 5:
    print("The month you entered is May")
if month == 6:
    print("The month you entered is June")
if month == 7:
    print("The month you entered is July")
if month == 8:
    print("The month you entered is August")
if month == 9:
    print("The month you entered is September")
if month == 10:
    print("The month you entered is October")
if month == 11:
    print("The month you entered is November")
if month == 12:
    print("The month you entered is December")
```

In the next section you may choose the exercises you attempt. There are three types of exercises to consider:

1. **Revision Exercises** - choose these exercises if you are not confident in your understanding of selection statements

2. **Development Exercises** - choose these exercises if you are confident in your understanding but want some more practice

3. **Stretch and Challenge Exercises** - choose these exercises if you feel you have mastered selection and want to tackle some tougher problems

Once you feel more confident attempt some of the more difficult exercises. The more practice you get now the more comfortable you will be using selection in more complex programs later in the course.

Remember, practice makes perfect!

Syntax

```
if guess == number:
    print('you guessed correctly')
```

prints the statement if the variables are equal.

```
if guess == number:
    print('you guessed correctly')
else:
    print('you guessed wrong')
```

prints the first statement if the variables are equal and the second if they are not.

```
if guess == number:
    print('you guessed correctly')
elif guess < number:
    print('No it is higher')
else:
    print('No it is lower')
```

prints the first statement if the variables are equal, otherwise it checks to see whether one is less than the other. If one is less than the other the second statement is printed and the third if they are not.

Vocabulary

`if statement`
: structure used when we want our program to execute a statement only if a certain condition is met.

`if..then..else`
: structure used when we want our program to do one statement if a certain condition is met, and another statement if the condition is not met.

`nested if (if..elif..if)`
: structure used when we want our program to check several possible conditions.

`compound statements`
: when more that one statement is between the if and elif/else part of an if statement.

`indentation`
: the process of structuring your code to make error location easier.

Class exercises

1. Create **pseudocode** or **flowchart** or **structure chart** first - plan your solution on paper before attempting it
2. Create a set of **test data** - select values you will enter to test the program and know beforehand what you expect the answers to be
3. Write the **program** - write the program in Python using the pseudocode to assist you and the test data to ensure the program functions correctly

Revision exercises

Attempt these tasks if you need to build your confidence and understanding of selection statements.

1. Write a program that asks for two numbers from the user and then displays a suitable message if the two numbers are the same.
2. Write a program that will read in a person's age and display a message whether they are old enough to drive or not.
3. Write a program that checks whether a number input is within the range 21 to 29 inclusive, and displays an appropriate message.
4. Write a program that asks the user to enter 2 numbers and displays the larger of the two numbers.
5. Extend exercise 3 so a number out of range will cause a message saying whether it is above or below the range.
6. Adapt exercise 4 to determine which is the largest of three given integers.

Development exercises

Attempt these tasks if you are confident in your understanding but feel you need more practice

7. Write a program that lets the user enter a number between 1 and 12 and displays the month name for that month number.

   The input *3* would therefore display March.
8. Write a program that reads in the temperature of water in a container (in Centigrade) and displays a message stating whether the water is frozen, boiling or neither.
9. Write a program that asks the user for the number of hours worked this week and their hourly rate of pay. The program is to calculate the gross pay.

   If the number of hours worked is greater than 40, the extra hours are paid at 1.5 times the rate. The program should display an error message if the number of hours worked is not in the range 0 to 60.
10. Write a program that reads in an exam mark and displays the relevant grade. The grade boundaries are:
    - 0 to 40 marks - grade U
    - 41 to 50 marks - grade E
    - 51 to 60 marks - grade D
    - 61 to 70 marks - grade C
    - 71 to 80 marks - grade B
    - 81 to 100 marks - grade A

Stretch and challenge exercises

Attempt these tasks if you feel you have mastered selection and want to tackle tougher problems.

11. Extend your program for exercise 7 to include leap years. A year is a leap year if the year divides exactly by 4, but a century is not a leap year unless it is divisible by 400. For example the year 1996 was a leap year, the year 1900 was not, the year 200 was a leap year.
12. Write a program that accepts a date as three separate integers such as 12 5 03. The program should display the date in the form 12th May 2003.
13. Adapt your program to interpret a date such as 12 5 95 as 12th May 1995. Your program should interpret the year to be in the range 1931 to 2030.