Week 3 Computer Seminar: Conditional Statements

This tutorial covers:

- flowcharts;
- if, if-else, elif;
- boolean operators: and, or, not;
- exception handling;
- generating random numbers.

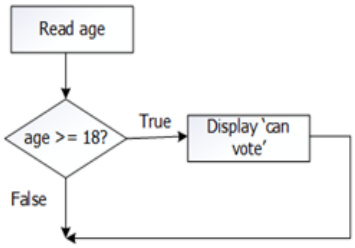
Part 1: if

Use **if** by itself to test a condition and do one set of actions if it is true, otherwise do nothing.

a) Type and run this example. Test with boundary values: 99, 100, 101

```
n = int(input('Give me a number over 100: '))
if n <= 100:
   print(n, 'is not over 100')
```

b) Write a program to read in someone' age. Display 'can vote' if they are old enough. We created the flowchart in the lecture. Now write the program.



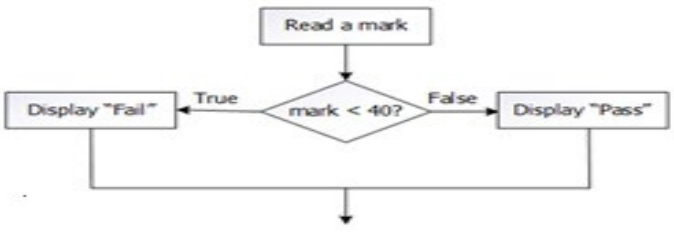
Part 2: if-else

Use an **if-else** to test condition, do one set of actions if it is true, otherwise do another set of actions.

a) Type and run this example. Test with values: -1, 0, 1

```
x = int(input('Give me a number: '))
if x < 0:
   print(x, 'is negative')
else:
   print(x, 'is positive')
```

b) Write a program to display "FAIL" if the mark entered is less than 40, otherwise display "PASS". We created the flowchart in the lecture.



- Now write the program. Test with boundary values: 39, 40, 41.

c) Write a program that checks whether a number is even or odd and then displays the appropriate message. (Hint – use the % operator).

Part 3: elif (chained conditional)

elif allows you to give the program additional conditions. Only the **first** true branch runs. If none of the conditions match then run the else-block (optional).

a) Temperature Program. Extend the program created for tutorial 2.
The program should do the following.
- Ask the user to enter '1' if they want to convert from Celsius to Fahrenheit or enter '2' if they want to convert from Fahrenheit to Celsius.
- Ask the user to enter a temperature in the correct unit.
- Formulas:

  f = (c * 1.8) + 32

  c = (f - 32) / 1.8
- Apply the conversion formula and display the converted temperature.
- Use an if to check if the user has entered '1', an elif to check if the user has entered '2' and an else to display 'invalid option entered'.

b) Create a simple calculator program in Python. The calculator should resolve simple arithmetic expressions by the user entering:
- An integer (e.g., *6*)
- One of the operators `+', `-', `*' or `/' (e.g., *)
- Another integer (e.g., *4*)
  - E.g., to give the expression *6*4*
- The program should calculate the value of the expression entered and display the result.
- Use the following data to test your program.

Integer 1	Operator	Integer 2	Expected Result
3	+	2	5
3	-	2	1
3	*	2	6
3	/	0	Error
3	/	2	1.5

c) Tip Calculator – A restaurant wants to suggest a tip according to the service diners receive. Write a program that calculates a tip according to the diner's satisfaction.
- Ask the diner to enter the cost of the meal.
- Ask the diners' satisfaction level using ratings: 1 = Totally satisfied, 2 = Satisfied, 3 = Dissatisfied.
  - If the diner is totally satisfied (1), calculate a 20 percent tip.
  - If the diner is satisfied (2), calculate a 15 percent tip.
  - If the diner is dissatisfied (3), calculate a 10 percent tip.
- Report the satisfaction level and tip. Your solution should use an *if*, *elif*, *else*.

Part 4: Boolean Operators

Boolean Operators **and**, **or**, **not -** compare values and evaluate to a single Boolean value.

- **and** - if the Boolean values on both sides of the and keyword are True, then the expression evaluates to True. If either or both of the Boolean values are False, then the expression evaluates to False.

  a) Would the following evaluate to True or False? Test your answers by entering the following expressions into the interactive shell at the prompt symbol (>>>):

  ```
  >>>True and True
  >>>True and False
  >>>False and True
  >>>False and False
  >>>spam = 'Hello'
  >>>10 < 20 and spam == 'Hello'
  ```

  - Note that 10 < 20 evaluates to True and spam == 'Hello' also evaluates to True, so the two Boolean expressions joined by the **and** operator evaluate as True.

  b) In the following, the overall condition is met if the number input is both greater than or equal to 1 and less than 11 (in the range 1 to 10).

  Fill in the missing line and then test with 0, 1, 10, 11.

  ```
  m = int(input('Give me number between 1 and 10: '))
  # complete missing line here
    print('Well done! You gave me: ', m)
  ```

- **or** - evaluates to True if *either* of the two Boolean values is True. The only time the or operator evaluates to False is if *both* of the Boolean values are False.

  c) Would the following evaluate to True or False? Test your answers by entering the following expressions into the interactive shell at the prompt symbol (>>>):

  ```
  >>>True or True
  >>>True or False
  >>>False or True
  >>>False or False
  >>>10 > 20 or 20 > 10
  ```

  - In the last example, 10 is not greater than 20 but 20 is greater than 10, so the first expression evaluates to False and the second expression evaluates to True. Because the second expression is True, this whole expression evaluates as True.

  d) Test the solution created in the lecture for Self-Check Question (part 2).

  - Add an additional condition at the start of the program to check if the exam mark is invalid (less than 0 or greater than 100):

  ```
  if mark >= 70:
  ```

```
    print('Exceptional result!')
elif mark >= 40:
    print('Satisfactory result!')
else:
    print('You have failed.')
```

- Trace which line would be printed for the following marks?

-1, 0, 1	69, 70, 71
39, 40, 41	99, 100, 101

- **not** – evaluates to the opposite Boolean value: True expressions evaluate to False, and False expressions evaluate to True.

e) Would the following evaluate to True or False? Test your answers by entering the following expressions into the interactive shell at the prompt symbol (>>>):

```
>>>not True
>>>not False
>>>not ('black' == 'white')
```

- Expression 'black' == 'white' is False. Therefore, not ('black' == 'white') is True.

f) Type and run the following to check your understanding of the not operator.

```
x = 10
if not x > 10:
    print("not returned True")
else:
    print("not returned False")
```

Part 5 - Create a program using an *if*, *elif*, *else*

- Ask the user 'Do you like Python? (yes/no): '
- if, elif, else structure:
  - If the user responds with 'yes' give an appropriate message (e.g. you are on the right course)
  - If the user responds with 'no' give an appropriate message (e.g., you might change your mind)
  - If the user gives a different response give an appropriate message (e.g., I did not understand).
- Run and test your program.
  - Then amend the program to accept additional ways of saying yes and no, e.g. Yes/No, y/n, etc
  - Experiment with the Python string method lower() and check it can improve your solution. Hint: if the users response is stored in a variable called response use: response = response.lower()

Part 6. Extended Exercises – Adding exception handling

Possible solution for **Question 1a**:

```
n = input('Give me a number over 100:  ')
```

```
n = int(n)
if n <= 100:
    print(n, 'is not over 100')
```

- Add exception handling to avoid a *ValueError* if an integer value is not entered.

Possible solution for **Part 1b**:

```
age = input('Enter your age:  ')
age = int(age)
if age >= 18:
    print("You can vote")
```

- Add exception handling to avoid a *ValueError* if an integer value is not entered.

Part 4 d) - You created a simple calculator program in Python**.**

- Add a test to ensure you do not divide by zero and produce a ZeroDivisionError. Try two solutions:
  - One using a nested if else solution.
  - Another using a try and except.

Part 7: Generating random numbers

Python has a random module that provides pre-written functions to generate random numbers. To use the functions in the random module use the import statement at the **top of your code**:

```
import random
```

Then randint function will return a random integer between (and including) the two integer arguments you pass it. We will store this random number in the variable called secret_number.

```
secret_number = random.randint(1,20)
```

To access the randint function we prefix it with the name of the module imported.

You can also try different ranges of numbers by changing the arguments. For example:

```
random.randint(1, 4)            # integers between 1 and 4 (including both 1 and 4).
```

a) Write a program that simulates a single flip of a coin. It should randomly print either 'Heads' or 'Tails'. Hint: generate a random number (0 or 1) and use an if-else statement to print 'Heads' when the result is 0, or 'Tails' otherwise.