Programming Fundamentals

Lecture 3 – Python Decision Making

Iresh Bandara

ROBERT GORDON UNIVERSITY ABERDEEN
INFORMATICS INSTITUTE OF TECHNOLOGY

Learning Outcomes

- This lecture addresses LO1, LO2 and LO4 for the module
- On completion of this lecture, students are expected to explain and apply
    - Flowcharts and program flows
    - IF – ELSE
    - IF – ELIF
    - Conditional expressions
    - Boolean operators
- Analyse program flows based on if-elif-else conditions

Programming

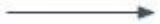
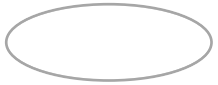
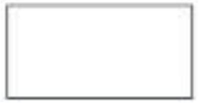
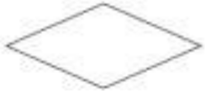
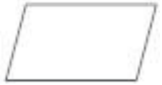
- How to create good code
  - Trial and error?
  - Specific strategy?
  - Time management and other environmental factors.
- How can we solve problems?
  - We need to clearly address the problem
    - Words, diagrams, models, maths ...
  - Design before the coding starts
  - Several versions due to revisions
  - Implementation and finally testing

error
//

Flowcharts

- Flowchart Diagrams advantages:
  - Decomposition: breaking down a problem into smaller **sub problems**.
  - Algorithm design: the ability to build a **step-by-step** process to solve a particular problem.
- Identifying the flow of the program.
  - Conditions: Various decisions and paths that lead
  - Repetitions
- Inputs and outputs of the program flows
- Part of the program design stage, before start writing the code

Flowchart elements

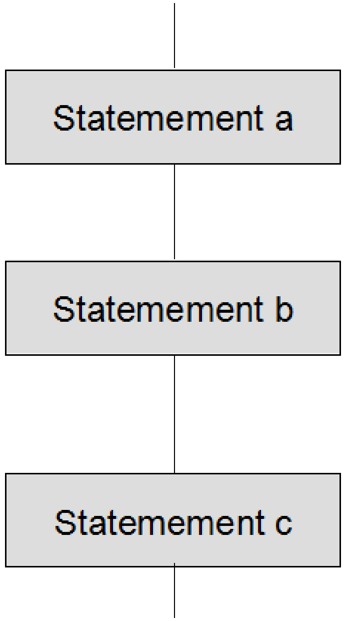
Symbol	Purpose	Description
	Flow line	Indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	start and end of flowchart.
	Process	Arithmetic operations and data-manipulations.
	Decision	When takes a decision/condition
	Input/Output	input and output operation

Program flow

- Sequential statements
- Conditional Statements
  - If, elif, else
- Repetitions
  - Loops
- Complex programs can have unimaginable number of conditions, repetitions
- What is the best starting point ?
  - Design stage : Pseudocode , flowcharts, etc.

Sequential statements

- Set of instructions that follows a logical order.



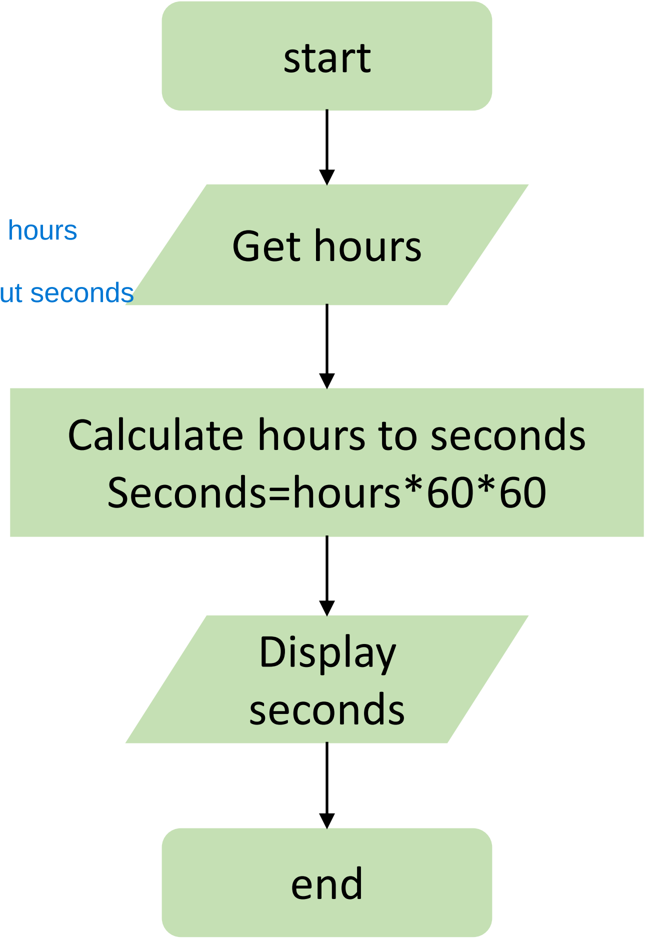
```
#start

a=50
b=60
total=a+b
print(total)

#end
```

Flowchart example

- Write an algorithm to convert hours into seconds
  - Start the program
  - Input number of hours
  - Calculate seconds.
  - Output seconds
  - End of the program

```
# Start the program
hours = int(input("Enter number of hours: ")) # Input number of hours
seconds = hours * 3600 # Calculate seconds
print("The equivalent number of seconds is:", seconds) # Output seconds
# End of the program
```

- What if the programmer wants to give 2 option?
  - Option 1 : convert hours to seconds
  - Option 2: convert seconds to hours



start

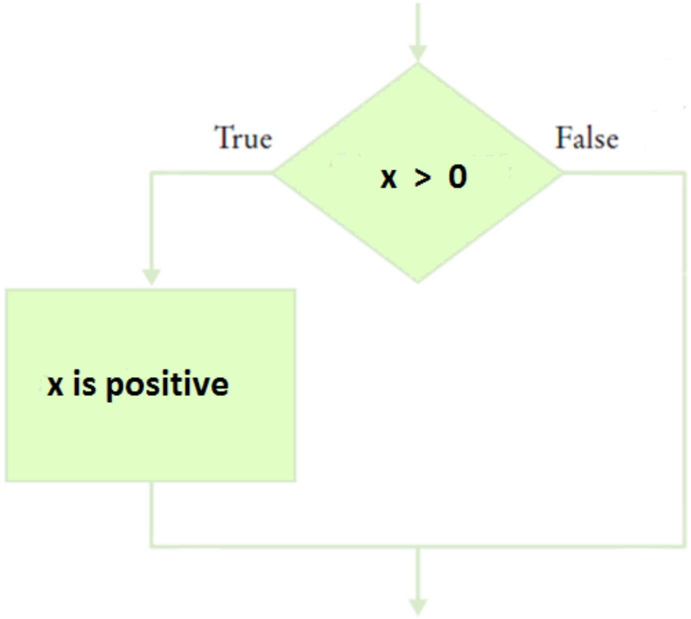
Get hours

Calculate hours to seconds
Seconds=hours*60*60

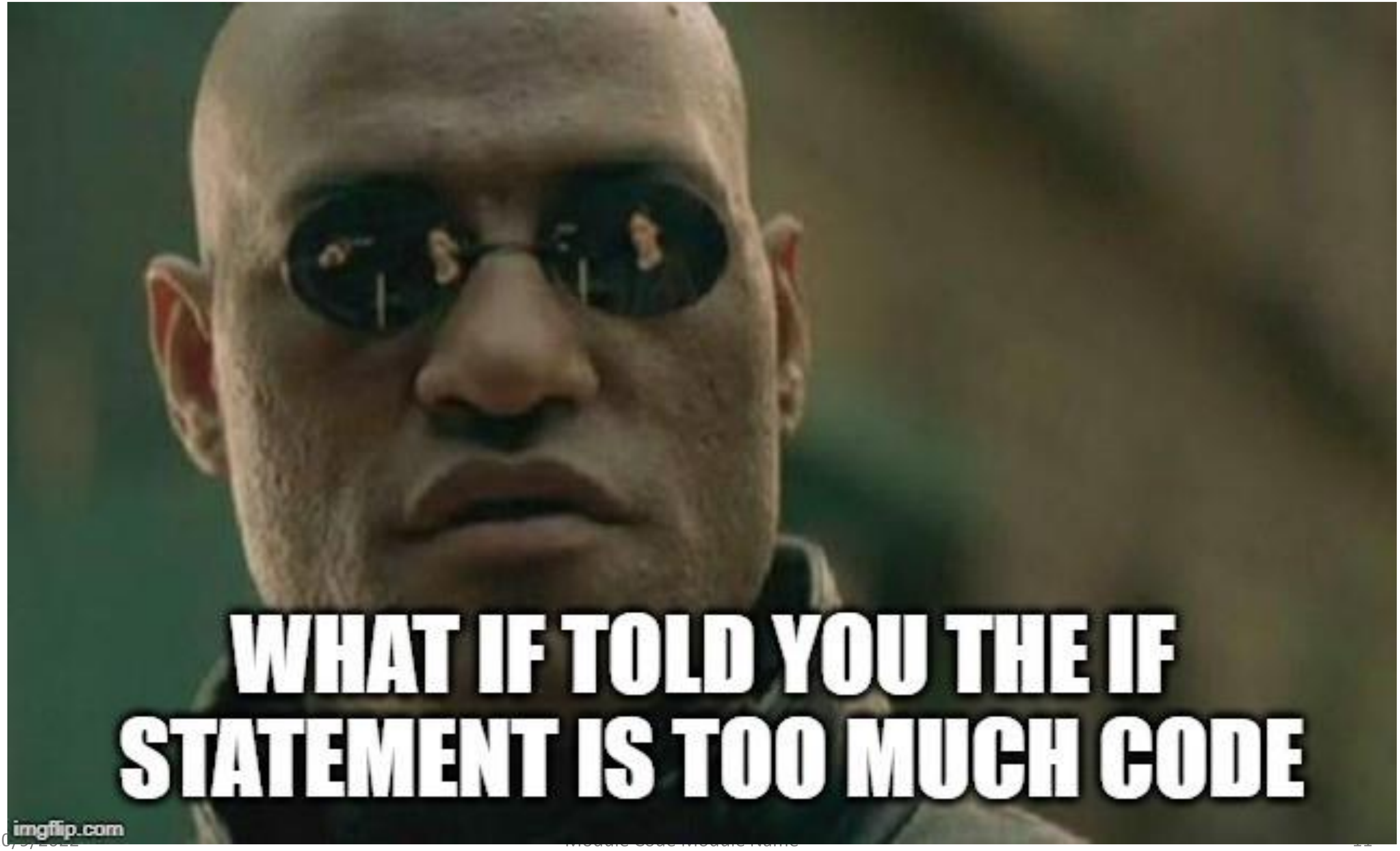
Display seconds

end

Flowchart vs the code : if conditions



```
#start

if x > 0:
     print('x is positive’)
else:
     #statement here
#end
```


WHAT IF TOLD YOU THE IF STATEMENT IS TOO MUCH CODE
imgflip.com

If – Else

```
#start
if x > 0:
    print('x is positive’)
else:
    #statement here
#end
```

The **if** and **else** must be in **lower case.** You must add a **colon** at the end of the statement.

Execution only x greater than 0

You must **indent** your print statement so that it is part of the if statement

Conditional Expressions

- Following conditional expressions can be used in order to form a if, elif

Operator	Meaning
==	Equal to
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or Equal to
<=	Less than or Equal to

Conditional Expressions - Example

```
letter = “b”
if letter == “a”:
    print(“Letter is a”)
else:
    print(“Letter is not a”)
```

‘==’ to compare the letter and “a”

It will execute else as the letter is b

Indentation

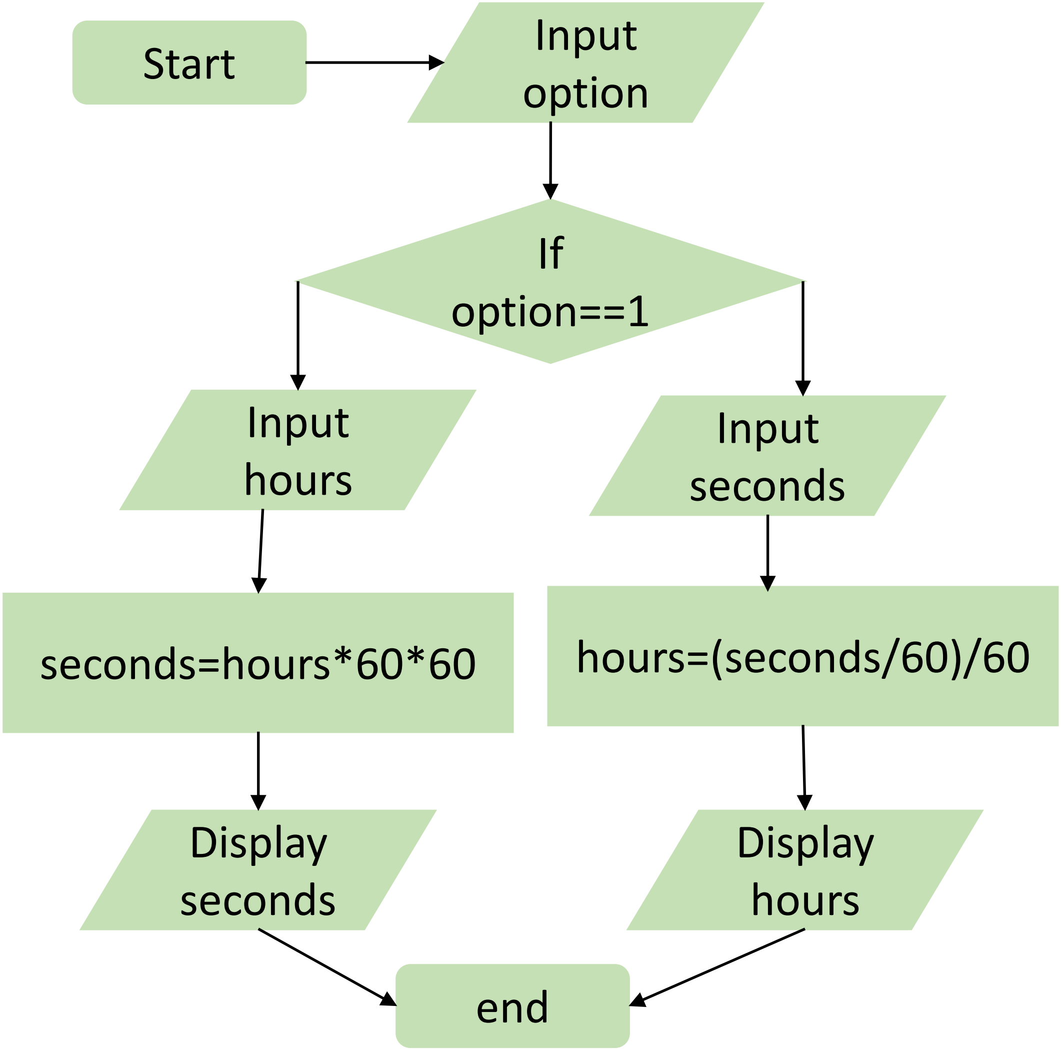
- Indentation is extremely important in Python.

```
letter = “b”
if letter == “a”:
     print(“Letter is a”)
     print (“inside if branch”)
print(“Outside if”)
```

block

- The print statement needs to be indented to be applied to the **if statement block.** Otherwise it will not execute as expected or “Indentation error”

Hours to seconds /seconds to hours



```
#start

op=input("Enter an option")

if op==1:
        hours=input("Enter hours")
        seconds=hours*60*60
        print(seconds)
else:
        sec=input("Enter seconds")
        hours=(sec/60)/60
        print(hours)


#end
```

Exercise 1

- Check the following statements are TRUE or FALSE
  - 2 != 3-1 false
  - 5 <= 4 false
  - 7 == -8+9 false
  - ‘apple’ == ‘Apple’ false
  - ‘a’ != ‘A’ true
  - 10 +11 >= 9+1 true
  - 25 => 6-9 False

Read about ASCII and UNICODE

- Insert above statements inside a if and see the results

Exercise 2

- What is the final value in *b*?
- Draw a flowchart for the scenario

```
a = 3
if a==3 :
    b = a * 2
if a < 4:
    b = a + 2
if a > 2:
    b = a * 2
```

a=3 then b=6

Exercise 3

- Is this program will execute ?

```
if a = b :
    print(“a and b are equal”)
```

- What is the opposite of the following statement
  - a>5

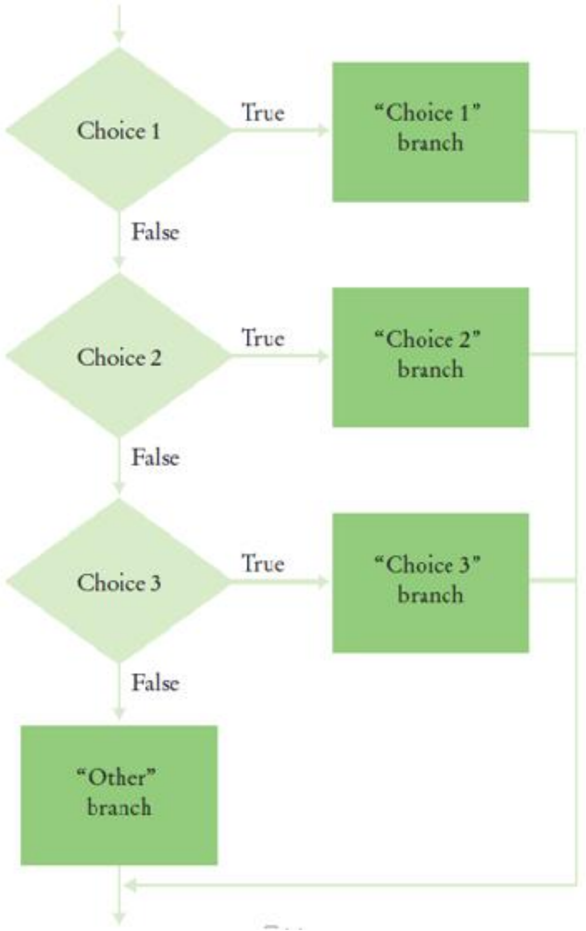
a<6

```
num = int(input("enter the marks")
if
    print("FAIL")
else
```

- Write a program to display “FAIL” if the mark entered is less than 50%, otherwise it should display “PASS”. Before write the program, draw the flowchart

IF – Elif - ELSE

- Elif to additional conditions.
- Can have multiple eilfs
- Program executes either if , one elif or else
- According to the diagram choice 1 is a if, choice 2 and 3 are 2 elifs and finally the else



Home
End
Insert
Delete
Backspace
Enter
Shift
gDn

Example

```
letter=“c”
if letter == “a”:
     print (“Letter is a”)
elif letter == “b”:
     print(“Letter is b”)
elif letter == “c”
     print(“Letter is c”)
else:
     print(“Letter is not a
 or b”)
```

As the value of variable letter is “c” it executes the second elif

Spot the difference

```
if mark >= 70:
          print(’Exceptional result’)
      if mark >= 40:
         print(’Satisfactory result’)
      else:
         print(’You have failed’)
```

```
if mark >= 70:
         print(’Exceptional result’)
      elif mark >= 40:
         print(’Satisfactory result’)
      else:
         print(’You have failed’)
```

Boolean operators : AND

- Evaluate TWO expression using AND

```
x = 20
y = 30
```

Expression	True/False
print(x == 20 **and** y == 30)	True
print(x == 10 **and** y == 30)	False
print(x == 10 **and** y == 20)	False
print(x == 20 **and** y == 100	False

Boolean operators : OR

- Evaluate TWO expression using OR

```
x = 20
y = 30
```

Expression	True/False
`print(x == 20 or y == 30)`	True
`print(x == 10 or y == 30)`	True
`print(x == 10 or y == 20)`	False
`print(x == 20 or y == 100`	True

Boolean operators : NOT

- NOT evaluates the opposite Boolean value
  - TRUE evaluates to FALSE
  - FALSE evaluates to TRUE

```
x = 20
```

Expression	**True/False**
`print(not x == 20)`	False
`print(not x == 10)`	True

Input validation using Boolean operators

- AND used to see the number follows the valid range (1 to 1000)

```
x=int(input(“Enter a number between 1 and 1000:”))
if x>=1 and x<=1000:
      print(“Valid number”)
else:
      print(“Your number is not valid”)
```

- Need several test cases to test all flows of the program

Challenge

- How a nested if (if inside a if) will work?

```
if some_condition :
   if another_condition:
        #statements
   elif alternative condition:
        if another_condition1
           #code here
   else:
        #some statements
else:
   #code here
```

- Draw a flowchart and see the flow of the program

Summary

- How to create good code
- Flowchart elements
- Indentation
- If condition
- Nested if-elif-else

- Thank you