STMC HKOI Training

Lesson 4: Conditional statement

Chan Yan Mong

October 10, 2021

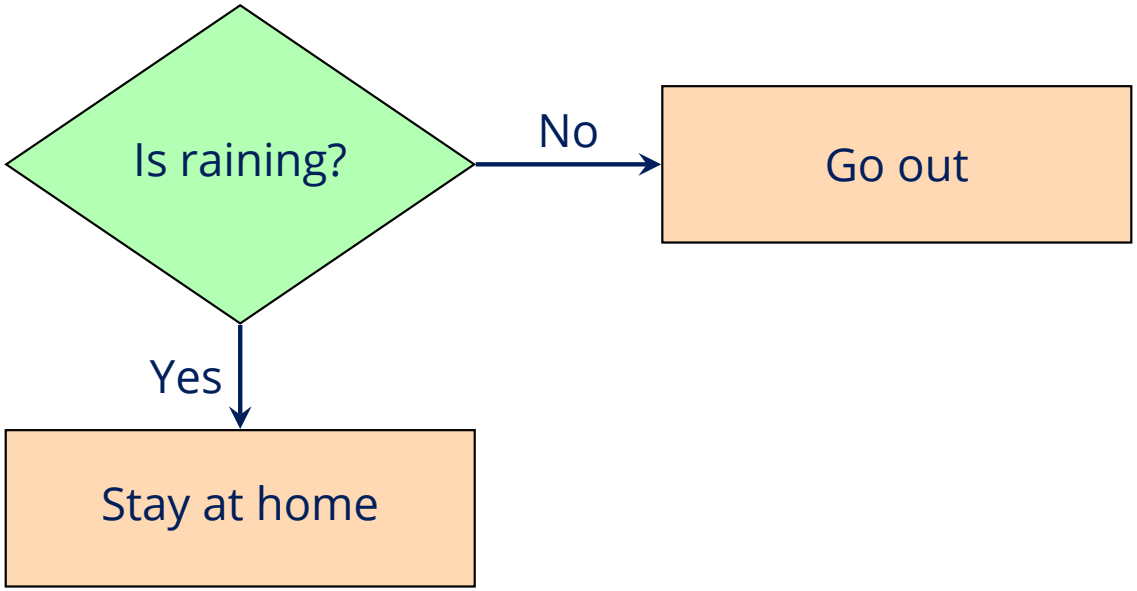
Goal today

- Conditional statement `if`
- Comparison operators `==`, `!=` `>`, `<`, `<=`, `>=`
- More conditional statements `elif`, `else`
- Truth table, Boolean operators $\wedge$, $\vee$, $\neg$

Making decisions

- Sometimes we want a program to not only compute values, but also to *make decisions*
- How do we make decisions? Let's consider the simplest case:
  - We ask a true or false question (e.g. Is it raining today?)
  - We look at the answer of the question and act differently if the answer is true or false (e.g. It is raining today $\rightarrow$ not go out; It's not raining $\rightarrow$ Go out)
- We shall depict such relationship as follows:

Making decision: Flow chart



Exercise

Before writing actual programs, let's exercise our thinking with flow charts. Draw a flow chart for a program that:

- Recieve two numbers a, b as input and print out the larger number. Return either a or b if $a = b$.
- Reads body temperature T from the user and prompt `"Fever"` if $T > 38.0$
- Takes in an age A and prompt `"valid"` if $3 \leq A \leq 100$. Otherwise print `"invalid"`

Boolean expressions

- As illustrated above, we need be able to ask questions in order to make decisions
- In python, ask questions with an **boolean expression** that return *true* or *false* depending on the evaluated answer
- For example of boolean expressions are:
  - Is x > y ?
  - Is x equal to y?
  - etc.
- Let's look at some examples to see how exactly can we do that in Python

Equal to ==

- The operator == is the operator for **equal to**
- **Do not confuse it with a single =**
- `x==y` checks if x is equal to y
- An example of the so called *comparsion operators*
- Examples:
  - `1 == 0` → `false`
  - `3 == 3` → `true`
  - `'a' == 'A'` → `false`
  - `'b' == 'b'` → `true`

Equal to ==

- We can also compare variables. For example

```
age = int(input('Enter you age: '))
print(age == 14) # Should return True if age equals to 14
```

- Another example

```
fatherAge = int(input('Father age: '))
motherAge = int(input('Mother age: '))
print(fatherAge == motherAge)
# True only if fatherAge equals motherAge
```

Equal to ==

- Similarly we can do that for string

```
1 yourName = input('Give me your name: ')
2 print('yourName inputted: ',yourName)
3 print(yourName == 'James') # Try inputing 'james', 'jAmEs' etc.
```

- Another example

```
1 password = input('Enter password: ')
2 print(password == 'L^Enb2%')
3 # You can imagine checking password this way might not be safe, to
    enhance safty people use hashing
```

Larger than > or smaller than <

- Similarly, we have `x > y` and `x < y`
- Checks if x is strictly larger than y or strictly smaller than y
- Example:
  - `1.2 < 3.7` $\rightarrow$ `true`
  - `0 < 1` $\rightarrow$ `false`
  - `3 > 1` $\rightarrow$ `true`
  - `9.9 > 3.7` $\rightarrow$ `true`
  - `3 < 3` $\rightarrow$ `false`

Larger than > or smaller than <

- Some examples:

```
examScore = float(input('Exam score: '))
print(examScore > 50.0) # True if examScore greater or equal to 50.0
```

- Some more examples

```
integer = int(input('Enter integer: '))
print(integer < 0) # True if it's negative number
```

More operators: >=, <=, !=

- Similarly, we also have **smaller than or equal to** <= and **larger than or equal to** >=
- Examples
  - `1 >= 1` → `true`
  - `3 >= 1` → `true`
  - `4 <= 1` → `false`
- We also have **not equal to** `!=`
- Examples:
  - `1 != 1` → `false`
  - `3 != 2` → `true`
  - `'a' != 'a'` → `false`
  - `'a' != 'A'` → `true`

More operators: >=, <=, !=

- Some examples:

```
timHeight = float(input('Height of Tim: '))
mayHeight = float(input('Height of May: '))
print(timHeight <= mayHeight) # True if Tim's height is less than or
      equal to May's
```

- Another example:

```
number1 = int(input('Enter a number: '))
number2 = int(input('Enter another number: '))
print(number1 != number2) # True if the two are not the same
```

`if` statement

- Now we know how to ask questions. Let's see how we can make decision
- In python decisions are made using **conditional statement**
- One simplest type is the `if` statement. Here is the syntax:

```
1  if """boolean expression""":
2    # Run if true
```

- Note that you must **indent the after if. Otherwise error will be raised**

`if` statement

Example:

```
# This code checks if a number is negative
number = int(input('Enter a number: '))

if number < 0:
  print('This is negative!') # Make sure to indent!

print('This will always be run') # This will always be run
```

Draw the flowchart of the code above

`if` statement

Exercise (Password checking):

Make up a password. Write a program that reads in a string `password` and check it against your made up password. Print `Login successful` if the password is correct and print nothing otherwise.

Exercise (Password checking 2):

Modify the code. Print `Login success` if successful and `Login failed` otherwise

`if-else` statement

- As shown in previous examples, we can definitely use two if statements to check otherwise
- However, `True` and `not True` are **mutually exclusive**. That is, they cannot happen at the same time. So the *second checking is redudant*
- To introduce a more effcient way to do it, we introduce the `if-else` statement. Here is the syntax:

```
if """condition""":
  # If True run here
else:
  # Otherwise run here
```

`if-else` statement

Some examples:

```
# Compare height
timHeight = float(input('Tim\'s height: '))
mayHeight = float(input('May\'s height: '))
if timHeight > mayHeight:
  print('Tim is taller')
else:
  print('May is taller or they have same height')
```

Again draw a flowchart of the code

`if-else` statement

Exercise:

John is a middle schooler. Everyday he can either be happy or unhappy. If he is happy, he will study; If he is not, he will play computer games. Let `happiness` be John's happiness. If `happiness` is 1, he is happy; otherwise, he is not. Write a program that predicts what John will do given his happiness.

Input: An integer `happiness` which is either 0 or 1
Output: Print "He will study" if he is happy and "He will play computer games" if not.

`if-else` statements

Exercise: Rainstorm signal

According to HKO, the Black rainstorm signal is issued if the hourly rainfall exceeds 70mm. Write a program that takes in the hourly rainfall `rainfall` in mm and determine whether the black rainstorm signal is issued. Return `BLACK` if so and `OTHERS` if otherwise.

Source: HKO

Example Input/output

```
Example 1:      Example 2:      Example 3:      Example 4:
$./main         $./main         $./main         $./main
0               70.0            72.4            23.1
OTHERS          OTHERS          BLACK           OTHERS
```

`if-else` statement

Exercise: Overbudget

Merry has $300 dollar in her pocket. Since Christmas is approaching, she decided to buy some gifts for her friends. The types of gifts she wanted to buy are: Pencil ($3.0 each), Cake ($11.0 each) and Book ($ 80.0 each). Suppose she bought n_p pencil, n_c cake and n_b books. Write a program to determine whether she exceeded her budget. If no, print `NO OVERBUDGET`; otherwise, print `EXCEEDED <amount>`.

```
Example 1:              Example 2:          Example 3:
$ ./main                ./main              ./main
101 0 0                 1 27 0              5 4 3
EXCEEDED 3.000000       NO OVERBUDGET       NO OVERBUDGET
```

Nested `if` statement

- Some times we want to test for more cases at once before outputing otherwise
- In those cases we might want to follow `else` with another `if`.
- We called that a **nested conditional statement** For example:

```
if """condition""":
  # Condition 1 True
else:
  if """condition 2""":
    # Condition 1 False but Condition 2 True
  else:
    # Condition 1 and Condition 2 both False
```

Again, remember to **indent twice for the nested statement**

More decisions

Exercise: Rainstorm signal+

According to HKO, the amber, red and black rainstorm signal is issued if the hourly rainfall exceed 30mm, 50mm and 70mm respectively (inclusive). Write a program that takes in a float `rainfall` and return `AMBER`, `RED` or `BLACK` accordingly if there's a signal, and `NO SIGNAL` if there is no signal. Furthermore, return `ERROR` if `rainfall` < 0.

```
Example 1:          Example 2:          Example 3:          Example 4:
$./main             $./main             $./main             $./main
0                   70.0                53.4                30.2
NO SIGNAL           BLACK               RED                 AMBER

Example 5:
$./main
-3
ERROR
```

`if-elif-else` statements

- As shown in previous exercise, nested statements can sometimes be messy
- Luckily, python provide another way to implement `else` followed by `if`
- We introduce the `elif` (*else-if*), which has the following syntax

```
if """Condition 1""":
  # If 1 is true
elif """Condition 2""": # Check if 1 is false
  # If 2 is true
else:
  # If both 1 and 2 are false
```

`if-elif-else` statement

- Of course, there are multiple variants of such statement.
- For example, if you don't want to do anything if both 1 and 2 are false:

```
if """condition 1""":
  # If 1 is true
elif """condition 2""":
  # If 1 false and 2 true
# No else here, so do nothing when both are false
```

`if-elif-else` statement

- Similarly, you can stack them together if you want to check a lot of cases

```
if """condition 1""":
  # 1 is true
elif """condition 2""":
  # 2 is true, 1 is false
elif """condition 3""":
  # 3 is true, 1 and 2 are false
elif """condition 4""":
  # 4 is true, 1 and 2 and 3 are false

# You can continue to stack as many cases as you like
```

`if-elif-else`

Exercise: Rainstorm++

Redo *Rainstorm+* using `if-elif-else` structure

Exercise: Grading program

We shall write a grading program. Take a `score` as input and print out a grade according to the following table:

Score (s)	Grade
$85 < s \leq 100$	A
$70 < s \leq 85$	B
$55 < s \leq 70$	C
$s \leq 55$	D

Boolean expression

- Recall **boolean expression** is an expression that **either returns true or false**
- Examples:
  - ”John has beard”
  - ”Spiders more than 2 legs”
  - ”x is equal to y”
  - ”There is more sand on the Earth than stars on the universe”
- Now we want to *combine* or *modify* these expressions

Combining expressions

- Let’s consider how boolean expressions can be combined
- Consider the statements:
  1. Today is raining
  2. Eva has an umbrella
- One way to combine them is to use the connective ”and”
- So we have ”Today is raining and Eva has an umbrella”
- Now, when is the new statement true? (i.e. suppose we were to put it inside a if statement, when should the if statement fire)

Combining expression

We can investigate the problem by using a **truth table**

Today is raining	Eva has an umbrella	Today is raining and Eva has an umbrella
T	T	T
T	F	F
F	T	F
F	F	F

We can see that the final statement ”Today is raining and Eva has an umbrella” is true only if *both* ”Today is raining” and ”Eva has an umbrella” are individually true

Combining expression

- In fact, we can see the above table is not limited to ”Today is raining” and ”Eva has an umbrella”
- For any boolean expression *a*, *b*, we can always combine *a*,*b* by asking if *a* and *b* is true
- Hence, the truth table above define the operation ”and”
- Let’s define different operations together

Logical AND $\wedge$

- The **logical AND** $\wedge$ is a binary operation
- It combines two statements a, b and return true only if *both* statements are true
- In python this is done using the `and` keyword
- Example:
  - `1 <= var and var < 3`
  - `(number % 3 == 0) and (number % 2 != 0)`
  - `(chr != 'A') and (chr != 'B')`

a	b	$a \wedge b$
T	T	T
F	F	F
F	T	F
T	F	F

Table 1: Truth table of $\wedge$

Logical AND ∧

- For example:

```
username = input('Username: ')
password = input('Password: ')
if username == 'chany' and password == '1234':
  print('Login success')
else:
  print('Login failed')
# P.S. Don't use this kind of password in real life
```

Logical AND ∧

- Another example:

```
1 age = int(input('Enter age: '))
2 if 0 < age and age < 120:
3   print('This age make sense')
4 else:
5   print('This age does not make sense')
```

Logical OR $\vee$

- The **logical OR** $\vee$ is a binary operation
- It combines two statements a, b and return true only if *either* of the statements are true
- In python, this is done using `or` keyword
- Example:
  - `(var == 1) or (var == 2)`
  - `count > 1 or count < -1`
  - `(var == 1) or (var != 3)`

a	b	$a \vee b$
T	T	T
F	F	F
F	T	T
T	F	T

Table 2: Truth table of $\vee$

Logical OR ∨

- Some examples:

```
instru = input('What musical instruments you like? ')

if instru == 'guitar' or instru == 'bass':
  print('So you like string instruments!')

elif instru == 'brass' or instru == 'trumpet':
  print('So you like wind instruments!')

else:
  print('Sorry, I don\'t know what is ',instru)
```

Logical OR $\vee$

- Another example that combines `and` with `or`

```
# Tickets are sold only to kids from age 8 to 15 or elderly from age
    60-80
age = int(input('Enter you age: '))

if (8 <= age and age <= 15) or (60 <= age and age <= 80):
  print('You are eligible')
else:
  print('Nope, you cannot buy this')
```

Negation operation ¬

- The **negation operation** ¬ is a unitary operation
- It is equivalent to adding ”not” to the statement
- In python, this is done by adding `not` in front of conditionals
- Example:
  - `not(var > 3)` $\leftrightarrow$ `(var <= 3)`
  - `not(var == 3)` $\leftrightarrow$ `(var != 3)`

a	$\neg a$
T	F
F	T

Table 3: Truth table of ¬

Negation operation ¬

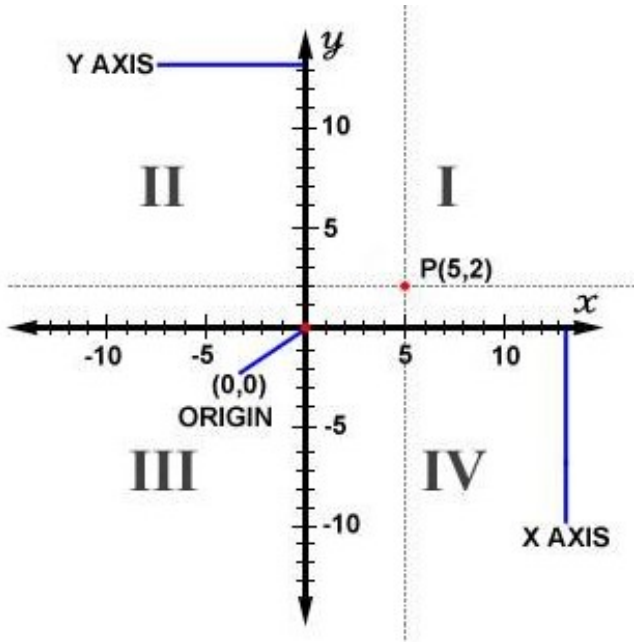
- Some examples

```
# Tick only available for people from 18 to 31
age = int(input('Enter your age: '))

if not (18 <= age and age <= 31):
  print('You cannot get the ticket')
else:
  print('Take the ticket')
```

Summary Exercises

1. Write a program that takes a coordinate point in a XY coordinate system and determine in which quadrant the coordinate point lies. If either *x* or *y* is zero, output `UNDEFINED`. Refer to the figure if you don't know what is a quadrant.



Source: CSGNetwork

Summary Exercise

Expected output:

```
./main          ./main          ./main          ./main
1.3 4.5         -1.4 2.0        -3.2 -4.4       3.3 -3.1
Quadrant 1      Quadrant 2      Quadrant 3      Quadrant 4

./main          ./main          ./main
0 4.5           -3.3 0          0 0
UNDEFINED        UNDEFINED      UNDEFINED
```

Summary Exercise

2. Write a program to find the eligibility of admission for a professional course based on the following criteria:

   Eligibility Criteria : Marks in Maths >=65 and Marks in Phy >=55 and Marks in Chem>=50 and Total in all three subject >=190 or Total in Maths and Physics >=140

Summary Exercise

Expected output:

- Math = 17, Phy = 32, Chem = 1

```
./main
17 32 1
Not Eligible
```

- Math = 70, Phy = 65, Chem = 55

```
./main
70 65 55
Eligible
```

- Math = 70, Phy = 69, Chem = 99

```
./main
70 69 99
Eligible
```

Summary Exercise

3. Every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the years 1700, 1800, and 1900 are not leap years, but the years 1600 and 2000 are. Write a program that determines whether a year is a leap year.

```
$./main                ./main                ./main
1700                   1600                  2012
Not leap year          Is leap year          Is leap year
```

Summary Exercises

4. Write a program that reads in 3 numbers and ouput the largest number. You are guaranteed that no two numbers are equal. Expected ouput

```
$./main
13 83 19
2nd number is largest. The value is 83

$./main
77 21 3
1st number is largest. The value is 77

$./main
0 21 88
3r number is largest. The value is 88
```

Additional exercise

Exercises

1. Write a program that takes in a number `num` and print `YES` if it is divisible by 2 or 3 and `NO` if otherwise
2. Body temperature T is considered `NORMAL` if $36 \leq T \leq 38$ and `ABNORMAL` otherwise. Without using `&&`, write a program that takes in a float `temp` and checks whether the body temperature is `NORMAL` or `ABNORMAL`

Additional exercise

Exercise

3. a, b are numbers that can only be 0 or 1. The operation $a \oplus b$ is defined using the table below. Write a program that takes in two integer a, b as input and compute $a \oplus b$

a	b	$a \oplus b$
0	1	1
1	0	1
0	0	0
1	1	0

4.

Additional exercise

Exercises

1. Write a program that takes in an integer `num` and print `"DIV BY 6"` if it's divisble by 6 and `"NOT DIV BY 6"` otherwise
2. Write a program that takes in a float `temp` and check if $0 \leq$ temp < 100.0. Print `"YES"` if it's within the range and `"NO"` otherwise