

STMC coding team Training

Lesson 3: Conditional statement

Tsai Yun Chen

February 29, 2024



Goal today

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

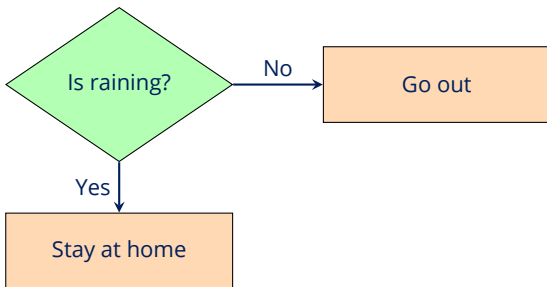


Making decisions

- 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 → not go out; It's not raining → 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:

- Receive 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 *comparision operators*
- Examples:
 - `1 == 0` → `false`
 - `3 == 3` → `true`
 - `'a' == 'A'` → `false`
 - `'b' == 'b'` → `true`



Equal to ==

- We can also compare variables. For example

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

- Another example

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



Equal to ==

- Similarly we can do that for string

```
1 yourName = input('What is your name: ')
2 print('yourName is:', yourName)
3 print(yourName == 'James') # Try inputting '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:
 - $1.2 < 3.7 \rightarrow \text{true}$
 - $0 < 1 \rightarrow \text{false}$
 - $3 > 1 \rightarrow \text{true}$
 - $9.9 > 3.7 \rightarrow \text{true}$
 - $3 < 3 \rightarrow \text{false}$



Larger than > or smaller than <

- Some examples:

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

- Some more examples

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



More operators: \geq , \leq , \neq

- Similarly, we also have **smaller than or equal to** \leq and **larger than or equal to** \geq
- Examples
 - $1 \geq 1 \rightarrow \text{true}$
 - $3 \geq 1 \rightarrow \text{true}$
 - $4 \leq 1 \rightarrow \text{false}$
- We also have **not equal to** \neq
- Examples:
 - $1 \neq 1 \rightarrow \text{false}$
 - $3 \neq 2 \rightarrow \text{true}$
 - $'a' \neq 'a' \rightarrow \text{false}$
 - $'a' \neq 'A' \rightarrow \text{true}$



More operators: >=, <=, !=

- Some examples:

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

- Another example:

```
1 number1 = int(input('Enter a number: '))
2 number2 = int(input('Enter another number: '))
3 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:

```
1 # This code checks if a number is negative
2 number = int(input('Enter a number: '))
3
4 if number < 0:
5     print('This is negative!') # Make sure to indent!
6
7 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 redundant*
- To introduce a more efficient way to do it, we introduce the if-else statement. Here is the syntax:

```
1 if ""condition"":  
2     # If True run here  
3 else:  
4     # Otherwise run here
```



if-else statement

Some examples:

```
1 # Compare height
2 timHeight = float(input('Tim\'s height: '))
3 mayHeight = float(input('May\'s height: '))
4 if timHeight > mayHeight:
5     print('Tim is taller')
6 else:
7     print('May is taller or they have same height')
```

Again draw a flowchart of the code



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 a pencil, b books and c cake. 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:
1	101	1	5
2	0	27	4
3	0	0	3
4	EXCEEDED 3.000000	NO OVERBUDGET	NO OVERBUDGET
5			



Nested if statement

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

```
1 if ""condition"":  
2     # Condition 1 True  
3 else:  
4     if ""condition 2"":  
5         # Condition 1 False but Condition 2 True  
6     else:  
7         # Condition 1 and Condition 2 both False
```

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



if-elif-else statements

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

```
1 if ""Condition 1"":  
2     # If 1 is true  
3 elif ""Condition 2"": # Check if 1 is false  
4     # If 2 is true  
5 else:  
6     # 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:

```
1 if ""condition 1"":  
2     # If 1 is true  
3 elif ""condition 2"":  
4     # If 1 false and 2 true  
5 # 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

```
1 if ""condition 1"":
2     # 1 is true
3 elif ""condition 2"":
4     # 2 is true, 1 is false
5 elif ""condition 3"":
6     # 3 is true, 1 and 2 are false
7 elif ""condition 4"":
8     # 4 is true, 1 and 2 and 3 are false
9
10 # You can continue to stack as many cases as you like
```



if-elif-else statement

Some examples:

```
1 # Compare height
2 timHeight = float(input('Tim\'s height: '))
3 mayHeight = float(input('May\'s height: '))
4 if timHeight > mayHeight:
5     print('Tim is taller')
6 elif mayHeight > timHeight:
7     print('May is taller')
8 else:
9     print('They have the same height')
```

Again draw a flowchart of the code



if-elif-else

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
$40 < s \leq 55$	D
$s < 40$	F



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 \wedge

- For example:

```
1 username = input('Username: ')
2 password = input('Password: ')
3 if username == 'animal' and password == 'elephant':
4     print('Login success')
5 else:
6     print('Login failed')
7 # P.S. Don't use this kind of password in real life
```



Logical AND \wedge

- 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 \vee

- Some examples:

```
1 instru = input('What musical instruments you like? ')
2
3 if instru == 'guitar' or instru == 'bass':
4     print('So you like string instruments!')
5
6 elif instru == 'brass' or instru == 'trumpet':
7     print('So you like wind instruments!')
8
9 else:
10    print('Sorry, I don\'t know what is ',instru)
```



Logical OR ∨

- Another example that combines and with or

```
1 # Tickets are sold only to kids from age 8 to 15 or elderly from age
   60-80
2 age = int(input('Enter you age: '))
3
4 if (8 <= age and age <= 15) or (60 <= age and age <= 80):
5     print('You are eligible')
6 else:
7     print('Nope, you cannot buy this')
```



Negation operation \neg

- The **negation operation** \neg 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 \leq 3)`
 - `not(var == 3) \leftrightarrow (var \neq 3)`

a	$\neg a$
T	F
F	T

Table 3: Truth table of \neg



Negation operation \neg

- Some examples

```
1 # Today (5/11) is the open day of HKUST (for real)
2 # There are events that bring visitors to visit different labs
3 # However these event should only open to student that are not
   already the student of HKUST
4
5 school=input('Which school do you belongs to? ')
6
7 if not (school=='HKUST'):
8     print('welcome to the event')
9 else:
10    print('you are already a student of HKUST')
```



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.

1	<code>\$/main</code>	<code>./main</code>	<code>./main</code>
2	<code>1700</code>	<code>1600</code>	<code>2012</code>
3	<code>Not leap year</code>	<code>Is leap year</code>	<code>Is leap year</code>
4			



Summary Exercises

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

```
1 $./main
2 13 83 19
3 2nd number is largest. The value is 83
4
5 $./main
6 77 21 3
7 1st number is largest. The value is 77
8
9 $./main
10 0 21 88
11 3rd number is largest. The value is 88
```



Finally, Homework

- Homework 2 is posted on the course website, namely the HW2.ipynb
- same as last time, 3 problems, sorted in ascending order of difficulty
- submit the homework to **the same place**, inside the folder of HW2 submission
- remember to include all your group member's name in the document
- deadline: before next lesson, i.e. 16/3
- the solution will be disclosed one week after the deadline, i.e. 23/3
- comment on HW1 will be given

