`if` statement

In programming we need to be able to carry out actions depending on the result of a decision. People do this every day:

If I get hungry, *I will eat some food*
If the weather is cold, *I will wear my coat*.

Conditional statements

Such sentences have two parts:

* a **condition part** ('If I get hungry', 'If the weather is cold')

and

* an **action part** ('I will eat my lunch', 'I will wear my coat')

`if` statement

In Python, the keyword **`if`** is used for such a statement.

As an example, we modify the program to convert metres to centimetres to test if the value of metres is positive (greater than 0) before converting it to centimetres and we write:

```
if m > 0:                # condition
    cms = m * 100        # action part
```

The action statement is **only carried** out if the **condition is true.**

The condition must be followed by a **:**

The action must be **indented**

The `if` statement tests if the value of `m` is greater than `0` `(m > 0)`

If this is the case, then the conversion is carried out.

A condition (e.g. `m > 0`) is a **Boolean expression** or a **conditional expression**.

This simply means that there are only **two** possible values (**true** or **false**) that the condition can yield.

If the value of `m` is greater than 0 then the condition **evaluates to true.**

If the value of `m` is less than or equals 0 then the **condition is false**.

Example L4.1

```
# convert5.py: converts metres to centimetres
# check quantity of metres is positive

m = float (input('\nEnter number of metres: '))

# Check if metres is positive

if m > 0:
    centimetres = m * 100
    print(f'\n {m} metres is {centimetres} cms\n\n')

if m <= 0:
    print(f'\nEnter a positive number for metres\n')
    print(f'\nYou entered: {m} \n\n’)
```

Note: We can have **one or more actions** after an `if`

Executing this program with -42 as input produces as output:

```
Enter number of metres: -42

Enter a positive number for metres

You entered -42
```

Executing this program with 5 as input produces as output:

```
Enter number of metres: 5

5 metres is 500 cms
```

if – else statement

In the previous example, the conditions

```
m > 0
m <= 0
```

are **mutually exclusive** in that only one can be true at any point in time. The value of `m` cannot be `> 0` and at the same time be `<= 0`

To cater for such situation, we can use the ***if-else*** statement

```
if (condition):
    action statements1      # do if condition is true
else:
    action statements2      # do if condition is false
```

if – else statement

```
m = float (input('\nEnter number of metres: '))

# Check if metres is positive

if m > 0:
    centimetres = m * 100
    print(f'\n {m} metres is {centimetres} cms\n\n')
else:
    print(f'\nEnter a positive number for metres\n')
    print(f'\nYou entered: {m} \n\n')
```

```
# pay.py: Calculate and display hourly pay

hours = float(input('\nEnter hours worked: '))

if hours > 100:
     print(f'\nHours worked cannot be > 100: {hours}')
else:
     rate = float(input('\nEnter rate per hour: '))
     if rate > 50:
          print(f'\nRate too large {rate}')
     else:
          pay = rate * hours
          print(f'\nPay = {pay} for {hours} hours
worked at {rate} per hour')
```

Running pay.py displays:

```
Enter number of hours worked: 20

Enter rate per hour: 20

Pay =  400.0 for 20.0 hours worked at 20 per hour

and running it again:

Enter number of hours worked: 20

Enter rate per hour: 80

Rate too large: 80
```

Combining conditions: `and`

We often need to combine two or more conditions in a statement. For example, when we decide to wear a coat based on:

If (*it is raining)* **and** (*it is cold)* I will wear a warm raincoat

Here we test two conditions: is it raining **AND** is it cold.

We only carry out the action (wear a warm raincoat) **if both conditions are true.**

When we combine conditions with **`and`**, the action will only be carried out if both (all) conditions **are true**.

We often need to check if the input to a program **lies in a range.**

For example, the age of a child in Ireland lies between 1 and 18 years which can be expressed as `(age > 0)` **and** `(age < 18)`

```
age = float(input('\n Enter age: '))

if ( age > 0 ) and ( age < 18 ):
    print(f'\n Child age: {age}')
```

This code outputs:

```
 Enter age: 12

 Child age: 12.0
```

Combining conditions: `or`

Sometimes we wish to carry out an action if any **one** of the conditions in a statement is true.

For example, when we decide to wear a coat based on:

If (*it is raining)* **or** (*it is cold)* I will wear a warm raincoat

Here we test two conditions: *is it raining* **OR** *is it cold*.

We only carry out the action (wear a warm raincoat) **if one (any) of the conditions is true.**

When we combine conditions with **`or`**, the action will be carried out if any of the conditions **are true**.

The age of a child lies between 1 and 18 years and the age of an adult is from 18 to 122

```
# L4.8: age.py: Check if age is for a child or an adult

age = float(input('\n Enter age: '))

if (age <= 0 ) or ( age > 122 ):
    print(f'\n {age} not in age range for adult or child\n')

if ( age > 0 ) and ( age < 18 ):
    print(f'\n Child age: {age}')

if (age >= 18 ) and ( age <= 122 ):
    print(f'\n Adult age: {age}')
```

L4.8 outputs

```
 Enter age: 34

 Adult age: 34.0
```

and

```
 Enter age: 12

 Child age: 12.0
```

and

```
 Enter age: 150

 150.0 not in age range for adult or child
```

General format of `if, if-else`

```
if condition:
        action1A
        action1B
```

The statements `action1A` and `action1B` will only be executed **if the condition is true**

```
if condition:
        action1A
        action1B
else:
        action2A
        action2B
```

The statements `action2A` and `action2B` will only be executed **if the condition is false**

Note that the `action` statements can be **any Python statement,** including `if` statements.

```
if condition1 and condition2:
            action1A
            action1B
```

The statements `action1A` and `action1B` will be executed **if both** `condition1` **and** `condition2` **are true**

```
if condition1 or condition2:
            action1A
            action1B
```

The statements `action1A` and `action1B` will be executed **if any one** (or both) of `condition1` **or** `condition2` **is true**

There are only six types of condition that can arise when comparing two numbers

They can be tested for

1. **equality** - are they the same ? `metres == 0`
2. **inequality** – are they different ? `metres != 0`
3. is one **greater than** the other ? `metres > 0`
4. is one **less than** the other ? `metres < 0`
5. is one **greater than or equal to** the other ? `metres >= 0`
6. is one **less than or equal to** the other ? `metres <= 0`

Time to practice !

- Copy all the examples from the slides above and get them to run in your Python environment.
- Then complete the exercises from the Handbook and get them to run.
- Finally carry out the assignments from the Handbook and get them to run.