

○ Hands-on Tests!  
○ Quizzes

# CONTROL FLOW STATEMENTS

## Conditional Statements

- ✓ If
- ✓ Elif
- ✓ Else

## Loops

- ✓ Range()
- ✓ For
- ✓ While
- ✓ Break
- ✓ Continue

## List Comprehension

5.

# Previously on the course...

---

- ✓ We learnt how to make different data structures and variables in python:
  - int , float , str , bool
  - list, dictionary, set , tuple
- ✓ We learnt how to calculate and operate different operations between variables:
  - $X + Y$  ,  $X * Y$  ,  $X // Y$  ...
  - $X \text{ in } Y$  ,  $X \text{ is } Z$
  - ...

## And now...



- But how can we change control flow?
  1. How to run only a specific parts of the code? → Conditional Statements
  2. How to run some parts of the code more than once? → Loops

# Contents



1

## Conditional Statements

*if ... elif ... else*

2

## Loops

*while ... for*

3

## List Comprehension

*A way to make lists quickly*

# Conditional Statements

---

- Sometimes we just want to execute certain parts of the code,
  - For example
    - if my salary < expenses → Send a warning for me.
    - If my salary >= equal to my expenses → Print “You’re fine!”
- Three important key words:
  - *if*
  - *elif*
  - *else*

# if Syntax

The key word to start conditional statements

```
if some_condition:  
    #some_code  
    #some_code
```

Always this notation comes after each conditional statement

Indentation is really important for all statements inside the conditional scope

*A simple example...*

```
if salary < expense:  
    debt = salary - expense  
    print("Warning!")
```

# if-else Syntax

If and else notations  
are lined up with each  
other

```
if some_condition:  
    #some_code  
    #some_code  
else:  
    #some_code  
    #some_code
```

# if-elif-else Syntax

If and else and elif notations are all lined up with each other again

You can have as many elif statements as you want!

```
if some_condition:
    #some_code
    #some_code
elif some_other_condition:
    #some_code
    #some_code
else:
    #some_code
    #some_code
```

*A simple example...*

```
if salary < expense:
    debt = expense - salary
    print("Warning!")
elif salary > expense:
    saving = salary - expense
    print("Great")
else:
    debt = 0
    saving = 0
    print("Not Bad")
```



**Now let's get back to the code...**

# Contents

1

## Conditional Statements

*if ... elif ... else*



2

## Loops

*while ... for*

3

## List Comprehension

*A way to make lists quickly*

# While loops

- With while loops in python, we can execute a set of lines of code as long as the condition is true:

This notation is used to start a while-loop.

**while**

**some\_condition:**

**#some\_code**

**#some\_code**

Always this notation comes after each conditional statement

Indentation is really important, it separates all the statements inside and outside the loop

*A simple example...*

```
i = 1
while i < 4:
    print(i)
    i += 1
```

```
>> 1
>> 2
>> 3
```

# While loops

---

- With while loops → You can execute a set of commands as many times as you want.
- break – continue - pass
- Be careful about using while loops!
  - You need to make sure that at some point, the condition in while loop will be True! Otherwise your program keeps running with no ending.
  - These errors are called run-time errors.

**Now let's see some examples  
using 'while' in code...**

# For loops - Iteration

---

- Some objects in Python are iterable ,
  - it's possible to iterate in every element of the object
    - **Mylist = ["hey" , 1 , 2 , 3]**
    - **#Some Iteration Code →**  

```
>> hey  
>> 1  
>> 2  
>> 3
```
- For iteration in python we use for loops

# For loops - Iteration

This notation is used for for-loops

```
for i in iterable_object:  
    #some_code  
    #some_code
```

Don't forget to use `""` here too!

Indentation is really important, it separates all the statements inside and outside the loop

*A simple example...*

```
iterable_object = (1 , 5 , 3, 2)  
for i in iterable_object:  
    print(i ** 2)
```

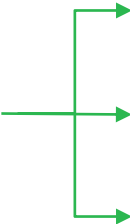
```
>> 1  
>> 25  
>> 9  
>> 4
```

**Now let's see the iteration  
concept in code...**



# For loops – range() function

- The range() function returns a sequence of numbers.

range(Start , Stop , Step) 

- Start: *Starting index (default = 0)*
- Stop: *Ending index – 1*
- Step: *Size of every jump (default = 1)*

- We can use range() function to make another type of loops.

# For loops – range() function

- Here's an example:

```
for i in range(0, 5, 1):
```

```
    print(i)
```

```
>> 0
```

```
>> 1
```

```
>> 2
```

```
>> 3
```

```
>> 4
```

"in" operator is used here again.

Don't forget to use ":" here too!

- Let's return to the code for more examples...

# Contents

1

## Conditional Statements

*if ... elif ... else*

2

## Loops

*while ... for*



3

## List Comprehension

*A way to make lists quickly*

# List Comprehension

---

- List comprehension is one of the special ways of creating lists in python.
- Other ways like using *.append()* method is also available.
- But, list comprehension is easier and includes less lines of code.
- Let's see some examples...

*Thanks!*

*Got any questions or suggestions?*

*Here's some contact info:*

*@KMasoumi*

