

Week 3

Introduction to Programming

19.9.2024

Last week (week 2)

- More conditional statements
 - elif
 - else
 - logical operators
- Simple loop: while True

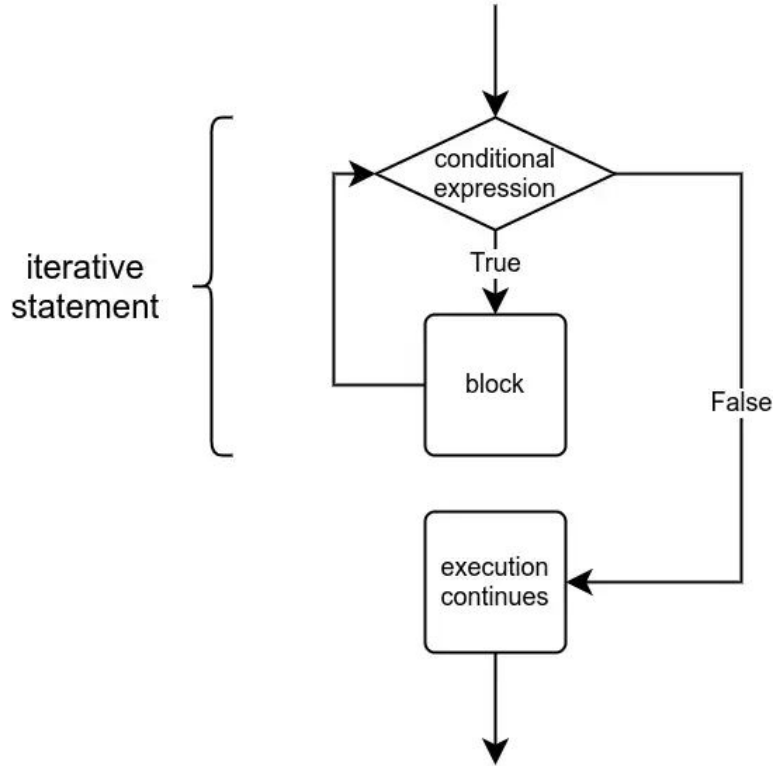
Questions

- Using external editor
- Private msgs to instructors

Repetition: while

Loop continues while the condition is True

If the condition is False to begin with, the loop is not executed at all



Initialisation, Condition & Update

The three components of a loop

Forgetting update is a common mistake
→ may lead to "inifinite" loop

```
# Ask the user for a number
number = int(input("Please type in a number: "))

# Repeat while the number is less than 10
while number < 10:

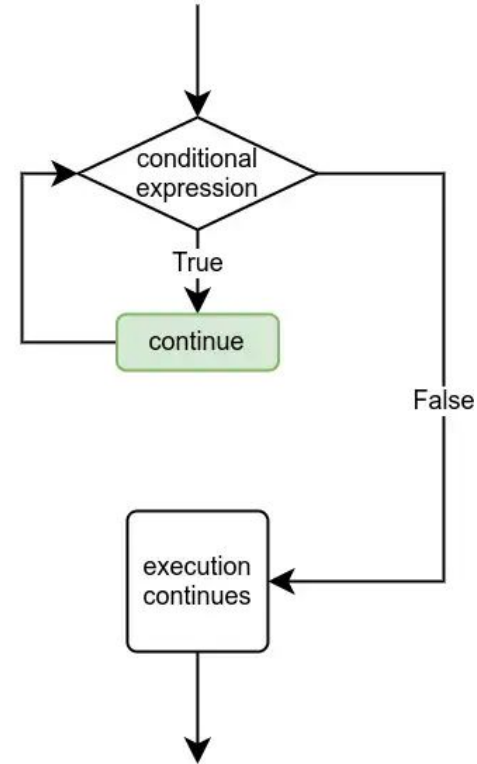
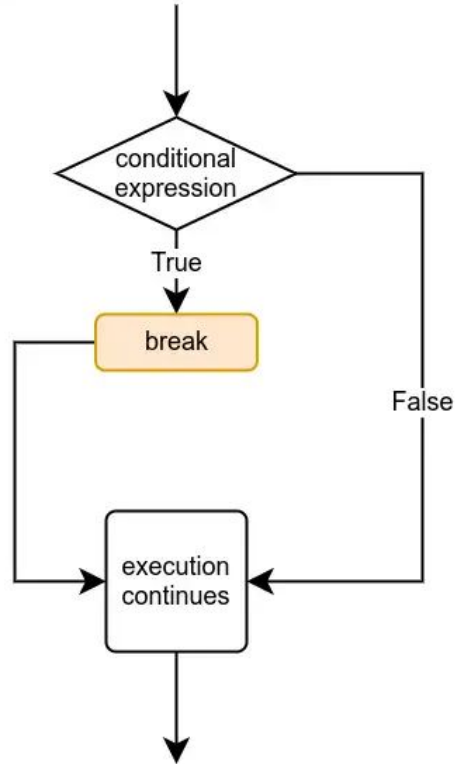
    # Print out and increment
    print(number)
    number += 1

print("Execution finished.")
```

The diagram illustrates the three components of a loop in the provided Python code:

- initialisation**: Points to the line `number = int(input("Please type in a number: "))`.
- condition**: Points to the line `while number < 10:`.
- updating variables**: Points to the line `number += 1`.

Break and continue



Nested loops

Pay attention to

- variables and their updates
- indentation

Indexing strings

A string has characters
between indices

```
[0, len(string) - 1]
```

```
String my_string = "Exemplary";
```

index

0	1	2	3	4	5	6	7	8
E	x	e	m	p	l	a	r	y

Character from a string

With brackets:

```
String my_string = "Exemplary";
```

```
my_string[0] # "E"
```

```
my_string[2] # "e"
```

```
my_string[4] # "p"
```

index

0	1	2	3	4	5	6	7	8
E	x	e	m	p	l	a	r	y

Indexing from end, negative indices

With brackets:

```
my_string[-1] # "i"
```

```
my_string[-5] # "e"
```

```
my_string[-9] # "E"
```

```
my_string = "Exemplary"
```

index	-9	-8	-7	-6	-5	-4	-3	-2	-1
	E	x	e	m	p	l	a	r	y

Substrings and slicing

Also with brackets

Syntax:

```
my_string[begin : end]
```

```
string = "Exemplary"  
sub = string[2 : 6]
```

index

0	1	2	3	4	5	6	7	8
E	x	e	m	p	l	a	r	y

Finding substrings

Operator **in** returns True, if substring can be found in string, False otherwise

Method **find** returns the first index of substring in a string or -1 if the substring cannot be found

Own functions

Functions are defined as

```
def function_name():  
    function code
```

Functions and parameters

Functions are defined as

```
def function_name(parameter list):  
    function code
```

Testing functions

Test code inside "main program":

```
def greet():  
    print("Hi!")
```

Write your main function within a block like this:

```
if __name__ == "__main__":  
    greet()
```

Next week

More functions

Lists

More tools for string manipulation