Control Structures

if / else / elif

Structuring by indentation

- Python uses indentations to group blocks of code
- A tab character is set for each indentation depth.

```
x = 243
y = 13
if x/y > 10:
    print("x/y is greater than 10")


if x/y > 10:
    if x/y < 100:
        print("x/y is greater than 10 and smaller than 100")
```

if (case distinction)

- The if keyword initiates a case distinction.
- The output is only made if the result of the test is `True`

```
x = 243
y = 13
if x/y > 10:
    print("x/y is greater than 10")


if x/y > 10:
    if x/y < 100:
        print("x/y is greater than 10 and smaller than 100")
```

Test

Result

```
x/y is greater than 10
x/y is greater than 10 and smaller than 100
```

if - else (otherwise)

If the condition of the test is not met, the statements in the else area are executed.

```
x = 243
y = 13
if x/y > 10:
    print("x/y is greater than 10")
else:
    print("x/y is smaller or equal to 10")

if x/y > 10:
    if x/y < 15:
        print("x/y is greater than 10 and smaller than 15")
    else:
        print("x/y is greater than 15")
else:
    print("x/y is smaller than 10")
```

Result

```
x/y is greater than 10
x/y is greater than 15
```

elif (multiple if)

If the first condition is not met the second condition is tested

```
x = 243
y = 13

if  x/y > 20:
    print("x/y is greater than 20")
elif  x/y > 10:
    print("x/y is greater than 10")

if x/y < 10:
    print("x/y is smaller than 10")
elif x/y > 20:
    print("x/y is greater than 20")
else:
    print("x/y is greater than 10 and smaller than 20")
```

Result

```
x/y is greater than 10
x/y is greater than 10 and smaller than 20
```

Comparisons

greater **>**, smaller **<**, equal **==**, unequal **!=**
greater or equal **>=**, smaller or equal **<=**

```
v1 = 57 + 98/12
v2 = 1728/14
if v1 >  v2:
    print("v1 is greater than v2")
elif v1 < v2:
    print("v1 is smaller than v2")
else:
    print("v1 is equal to v2")
```

```
x1 = 33*43
x2 = 345 + 1109
x3 = 18 + 2289 - 888
if x1 == x2:
    print("x1 is equal to x2")
elif x2 == x3:
    print("x2 is equal to x3")
elif x1 == x3:
     print("x1 is equal to x3")
else:
    print("x1, x2 and x3 are all different")
```

Result

```
v1 is smaller than v2
x1 is equal to x3
```

And / Or

Comparisons can be combined with and and or :

```
v1 = 33*43
v2 = 345 + 1109
v3 = 18 + 2289 - 888
if v1 == v2 and v2 == v3:
    print("all values are equal")
elif v1 == v2 or v2 == v3 or v1 == v3:
    print("at least two values are equal")
else:
    print("all values are different")
```

Resultat

```
at least two values are equal
```

Loops

while (as long as)

- The while keyword initiates a loop (iteration).
- The output is repeated until the result of the test becomes `False`

```
v1 = 33 * 43
v2 = 445

print("v2=", v2)
while(v1 > v2):
    print("v1=", v1, "v2=", v2)
    v1 = v1 - 199
```

Test

Result

```
v2= 445
v1= 1419 v2= 445
v1= 1220 v2= 445
v1= 1021 v2= 445
v1= 822 v2= 445
v1= 623 v2= 445
```

while (as long as)

- The instructions are repeated until the result of the test becomes `False`

```
v = 200
while v > 2:
    print(round(v,4))
    v = v/3
```

Result

```
200
66.6667
22.2222
7.4074
2.4691
```

```
v=2
while v < 1000:
    print(v)
    v = pow(v,2)
```

Result

```
2
4
16
256
```

while (Number Guess)

```
myNumber = 1305595
again = True
counter = 0

while again:
    counter = counter + 1
    nextValue = input("Please insert a number\n")
    test = int(nextValue) * 56765
    if test < myNumber:
        print("This number is too small")
    elif test > myNumber:
        print("This number is too big")
    else:
        print("You found the correct number")
        again = False
print("Game over, number of trials: ", counter)
```

The search continues until the correct number is found

Result

```
Please insert a number
4
This number is too small
Please insert a number
100
This number is too big
Please insert a number
23
You found the correct number
Game over: number of trials:  3
```

for (for all values in...)

- The for keyword initiates an iteration.
- The output is performed for all items in a list or in a range

```
#range creates a number range (from - to - step)
for i in range(3):
    print(i)
print("---")
for i in range(4, 8):
    print(i)
print("---")
for i in range(2, 8, 2):
    print(i)
```

All numbers up (and without) 3

All numbers from 4 to 7, step 1

All numbers from 2 to 7, step 2

Result

```
i= 0
i= 1
i= 2
----
i= 4
i= 5
i= 6
i= 7
----
i= 2
i= 4
i= 6
```

for (for all values in…)

- Add all numbers from 0 to 9

```
sum = 0
for elem in range(10):
    sum = sum + elem
print(sum)
```

Result

```
45
```

- Multiply the numbers from 1 to 5

```
product = 1
for elem in range(1,6):
    product = product * elem
print(product)
```

Result

```
120
```

Example with while: Create Triangle

Using a while construct

```
characterSet = "@" * 10
counter = 1
while counter <= len(characterSet):
    print(characterSet[:counter])
    counter = counter + 1
```

Result

```
@
@@
@@@
@@@@
@@@@@
@@@@@@
@@@@@@@
@@@@@@@@
@@@@@@@@@
@@@@@@@@@@
```

Example with for: Create Triangle

Using a for construct

```
characterSet = "@" * 10
for counter in range(len(characterSet)):
    print(characterSet[:counter])
```

Result

```
@
@@
@@@
@@@@
@@@@@
@@@@@@
@@@@@@@
@@@@@@@@
@@@@@@@@@
@@@@@@@@@@
```

Example with while: Create Triangle

Using a while construct

```
size = 10
characterSet = "@" * size
empty = " " * size
counter = 1
while counter <= size:
    print(empty[:size-counter] + characterSet[:counter] +
          characterSet[:counter])
    counter = counter + 1
```

Result

Example with for: Create Triangle

Similarly with a for construct

```
size = 10
characterSet = "@" * size
empty = " " * size
for counter in range(size):
    print(empty[:size-counter] + characterSet[:counter] +
          characterSet[:counter])
```

Result