week3

September 23, 2025

1 Week 3 - Loops, Conditions and Functions

Instructor: Johanne Sejrskild
Date: 12.08.2025

Student: Réka Forgó (au737257@uni.au.dk)

Date of latest changes: 23.09.2025

Good afternoon
Today we are going to work with loops, condions and functions. The green excercises will be highly linked to what you livecoded with Anna. If you find them challenging use yesterdays work as a help or ask. If you want to challenge yourself, try and do them all without using any help. In the yellow excercises we will make a password checker and a function that converts temperatures. And for the red excercise we will make a function that calculates grades

Structure of the notebook:
Green excercises

Conditional Statements

Loops

Functions

Yellow excercises

Password checker

Temperature converter

Red excercises

Grade calculator

Optional extra: Grade calculator as a class

Start with the first excercise, and then continue in order. Feel free to work together, and see how far you can get.
The important thing is to learn, not to solve all the challenges!

---

[2]:
```
# Before we start we need to import the necessary packages
%pip install pandas
%pip install lxml
import pandas as pd
```

```
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.12/site-
packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in
/opt/anaconda3/lib/python3.12/site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/anaconda3/lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/opt/anaconda3/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
/opt/anaconda3/lib/python3.12/site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: lxml in /opt/anaconda3/lib/python3.12/site-
packages (5.2.1)
Note: you may need to restart the kernel to use updated packages.
```

Green excercises

1.1 Conditional Statements

The *if* statement

[3]:
```
# Basic syntax
# if condition:
    # code to execute if condition is True
```

[4]:
```
# Example of if statement
# Before running: What will happen and why?

x = 10
if x > 5:
    print("x is greater than 5")
```

```
x is greater than 5
```

[5]:
```
# Exceise: Modify the code to check if x is less than 5 and print a different␣
↪message.

if x < 5:
    print("x is less than 5")
```

The *else* statement

[6]:
```
# Basic syntax

# if condition:
    # code if condition is True
# else:
    # code if condition is False
```

[7]:
```
# Example of if-else statement
# Before running: What will happen and why?

x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

```
x is not greater than 5
```

[8]:
```
# Excercise - Change the code above so that it prints "x is greater than 5." by␣
 ↪only changing one number/symbol.
# How many solutions can you find? and is some of them better than others?

if x < 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

```
x is greater than 5
```

The *elif* statement (for multiple conditions)

[9]:
```
# Basic syntax

# if condition1:
    # code if condition1 is True
# elif condition2:
    # code if condition2 is True
# else:
    # code if neither condition is True
```

[10]:
```
# Example of if-elif-else statement
# Before running: What will happen and why?

x = 5
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
```

```
else:
    print("x is less than 5")
```

```
x is equal to 5
```

[11]:
```
# Exercise - What happens when the condition for if and elif are the same? Why?


if x > 5:
    print("x is greater than 5")
elif x > 5:
    print("x is equal to 5")
else:
    print("x is less than 5")

# The first condition is checked, and if it's true, the corresponding block is␣
 ↪executed.
# Hence if if and elif are the same, the elif block will never be executed.
```

```
x is less than 5
```

1.2 Loops

Loops in Python are used to execute a block of code repeatedly. We will look at two different types of loops; the *for loop*, and the *while loop*.
The for loop iterates over sequences, so it keeps executing until it has reached the end of the loop.
The while loop keeps repeating while a condition is true, and first ends when the condition becomes false.

[12]:
```
# For loop - Basic syntax
# for element in sequence:
    # code to execute for each element

# Example of for loop
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

```
apple
banana
cherry
```

[13]:
```
# Exercise - Modify the code to print the length of each fruit instead of the␣
 ↪fruit itself.

for fruit in fruits:
    print(len(fruit))
```

```
5
6
```

```
6
```

[14]:
```python
# While loop - Basic syntax
# while condition:
    # code to execute while condition is True

# Example of while loop
x = 5
while x > 0:
    print(x)
    x -= 1
```

```
5
4
3
2
1
```

[15]:
```python
# Excercise - Modify the code to count up from 1 to 5 instead of counting down␣
 ↪from 5 to 1.

x = 1
while x <= 5:
    print(x)
    x += 1
```

```
1
2
3
4
5
```

1.3 Functions

[16]:
```python
# Heres a calculator
def add_two_numbers(a, b):
    result = a + b
    print(f"{a} + {b} = {result}")
```

[17]:
```python
# Try it out with different numbers
add_two_numbers(3, 5)
add_two_numbers(10, 20)
add_two_numbers(-1, 1)
```

```
3 + 5 = 8
10 + 20 = 30
-1 + 1 = 0
```

[18]:
```
# Make a subtraction function
def subtract_two_numbers(a, b):
    result = a - b
    print(f"{a} - {b} = {result}")

subtract_two_numbers(10, 5)
subtract_two_numbers(20, 30)
```

```
10 - 5 = 5
20 - 30 = -10
```

Yellow excercises

1.4 Password checker

You're making a simple password checker. How would you check if the password is correct?

Write code that asks for a password and checks if it follows the requirements: - It needs to be between 8 and 32 characters long

Make sure you print some meaningful message the user can understand. Try it with both the correct password and wrong passwords.

Hint: use the input function to allow for user inputs e.g. **input("Enter the password:")**

[19]:
```
# code here

def password_checker(password):
    if len(password) < 8:
        return "Password is too short"
    elif len(password) > 32:
        return "Password is too long"
    else:
        return "Password is valid"

input_password = input("Enter your password: ")
print(password_checker(input_password))
```

```
Password is too short
```

Now lets add some extra requirements: - The password needs to have at least 3 digits - And there needs to be an uppercase letter

[20]:
```
# Modified code here

def password_checker(password):
    if len(password) < 8:
        return "Password is too short"
    elif len(password) > 32:
        return "Password is too long"
    elif sum(char.isdigit() for char in password) < 3:
        return "Password must contain at least three digits"
    elif not any(char.isupper() for char in password):
        return "Password must contain at least one uppercase letter"
    else:
        return "Password is valid"

input_password = input("Enter your password: ")
print(password_checker(input_password))
```

```
Password must contain at least one uppercase letter
```

And the last bits, now we also need to ensure that the password contains: - 2 special characters

Make sure the user understands why a wrong password is rejected. And ensure that only a password that holds all requirements are passed.

[21]:
```
# Last modifications on the code - final password_checker

special = "!@#$%^&*()-_=+[]{}|;:'\",.<>?/`~" # Define special characters


def password_checker(password):
    if len(password) < 8:
        return "Password is too short"
    elif len(password) > 32:
        return "Password is too long"
    elif sum(char.isdigit() for char in password) < 3:
        return "Password must contain at least three digits"
    elif not any(char.isupper() for char in password):
        return "Password must contain at least one uppercase letter"
    elif sum(char in special for char in password) < 2:
        return "Password must contain at least two special characters"
    else:
        return "Password is valid"

input_password = input("Enter your password: ")
print(password_checker(input_password))
```

```
Password must contain at least one uppercase letter
```

1.5 Temperature converter

Lets now make a function that can convert celcius to fahrenheit. Make an IPO with your group and discuss what hte input, the process and the outpus should be.

The function should print an informational message and return something usable.

Hint: F = (celsius * 9/5) +32

[22]:
```
# celsius to fahrenheit converter

def celsius_to_fahrenheit(celsius):
    if celsius < -273.15:
        return "Temperature below -273.15°C is not possible."
    else:
        fahrenheit = (celsius * 9/5) + 32
        return f"{celsius}°C is equal to {fahrenheit}°F"

# Try it out with different numbers

print(celsius_to_fahrenheit(0))
print(celsius_to_fahrenheit(100))
print(celsius_to_fahrenheit(-300))
```

```
0°C is equal to 32.0°F
100°C is equal to 212.0°F
Temperature below -273.15°C is not possible.
```

Now make a function that does the opposite.
Hint: C = ((fahrenheit-32)/(9/5))

[23]:
```
# Fahrenheit to Celsius converter

def fahrenheit_to_celsius(fahrenheit):
    if fahrenheit < -459.67:
        return "Temperature below -459.67°F is not possible."
    else:
        celsius = (fahrenheit - 32) * 5/9
        return f"{fahrenheit}°F is equal to {celsius}°C"

# Try it out with different numbers
print(fahrenheit_to_celsius(32))
print(fahrenheit_to_celsius(212))
print(fahrenheit_to_celsius(-500))
```

```
32°F is equal to 0.0°C
212°F is equal to 100.0°C
Temperature below -459.67°F is not possible.
```

Well done! Now wouldn't it be neat if we could just wrap that into a function that we could call temperature_converter with two parameters "temperature" and "unit" so that we could convert both ways.

[26]:
```
# Temperature converter here

def temperature_converter(value, scale):
    if scale == 'c': #check if the scale of the value is celsius
        if value < -273.15:
            return "Temperature below -273.15°C is not possible."
        else:
            fahrenheit = (value * 9/5) + 32
            return f"{value}°C is equal to {fahrenheit}°F"
    elif scale == 'f': #check if the scale of the value is fahrenheit
        if value < -459.67:
            return "Temperature below -459.67°F is not possible."
        else:
            celsius = (value - 32) * 5/9
            return f"{value}°F is equal to {celsius}°C"
    else:
        return "Invalid scale. Please use 'c' for Celsius or 'f' for Fahrenheit.
 ↪"

#Let's make input fields which asks the temperature value in one scale and␣
 ↪converts it to the other.
value = float(input("Enter the temperature value: "))
scale = input("Enter the scale (c for Celsius, f for Fahrenheit): ")
print(temperature_converter(value, scale))
```

```
34.0°F is equal to 1.1111111111111112°C
```

OBS: To ensure you can go back in 3 months time and read you code and understand the logics behind it it needs to be well commented.
So, take look at the yellow excercises and see if you need to add some meaningful comments about the logics and coding choices.

:))

Red excercises

1.6 Grade calculator

The scenario: You're helping a teacher who wants to calculate student grades quickly. The teacher has been doing these calculations by hand, which takes a long time and sometimes leads to mistakes.

The challenge: - Calculate the average of test scores - Convert percentage grades to letter grades (it's just easier than the danish grading metric ngl) - Calculate final grades with different weights for different assignments

Example data:
Tests count for 70%, homework counts for 30%

Test scores: 85, 93, 78, 96, 88 Homework average: 90

Now we start with one thing first. Each functions should do *one* thing effectively.

[27]:
```
# Make a function that calculate the average of test scores.
#we need to provide a list of scores to the function when we call it.
# we need to set it up so it is able to handle homework and test scores with␣
 ↪different weights
```

```
def calculate_average(test_scores, homework_scores, test_weight=0.7,␣
↪homework_weight=0.3):
    if not test_scores and not homework_scores:
        return "No scores provided."
    test_average = sum(test_scores) / len(test_scores) if test_scores else 0
    homework_average = sum(homework_scores) / len(homework_scores) if␣
↪homework_scores else 0
    overall_average = (test_average * test_weight) + (homework_average *␣
↪homework_weight)
    return print(f"Overall average score: {overall_average}"), overall_average

# Try it out with different scores
#print(calculate_average([85, 93, 78, 96, 88], [90]))

overall_average = calculate_average([85, 93, 78, 96, 88], [90])[1]
```

```
Overall average score: 88.6
```

Well done, now we move on to the next sub-task

Hint: Grade to lettter grade converter
90–100 → A
80–89 → B
70–79 → C
60–69 → D
0–59 → F

[28]:
```
# Make a function that converts the test scores to letter grades

def percentage_to_letter_grade(percentage):
    if percentage >= 90:
        return "A"
    elif percentage >= 80:
        return "B"
    elif percentage >= 70:
        return "C"
    elif percentage >= 60:
        return "D"
    else:
        return "F"



# Test with different percentages
print(percentage_to_letter_grade(overall_average))
```

```
B
```

Last task, We need to calculate the weighted grade

[29]:
```
#def calculate_weighted_grade():

# already done above by the calculate_average function
```

Now we want to give the professor one function that uses all these three and wraps them in a neat little function with everything put together.

[30]:
```
def calculate_student_grade(test_scores, homework_scores, test_weight=0.7,␣
 ↪homework_weight=0.3):
    if not test_scores and not homework_scores:
        return "No scores provided."
    overall_average = calculate_average(test_scores, homework_scores,␣
 ↪test_weight, homework_weight)[1] # just the numeric average (not the print␣
 ↪statement)
    return percentage_to_letter_grade(overall_average)

# Create input fields that ask for test scores and homework scores
input_test_scores = input("Enter test scores separated by commas: ")
input_homework_scores = input("Enter homework scores separated by commas: ")

# Fix the parsing to handle decimal numbers properly
test_scores = []
if input_test_scores.strip():
    for score in input_test_scores.split(","):
        try:
            test_scores.append(float(score.strip()))
        except ValueError:
            print(f"Warning: '{score.strip()}' is not a valid number, skipping..
 ↪.")

homework_scores = []
if input_homework_scores.strip():
    for score in input_homework_scores.split(","):
        try:
            homework_scores.append(float(score.strip()))
        except ValueError:
            print(f"Warning: '{score.strip()}' is not a valid number, skipping..
 ↪.")

print(f"Final Grade: {calculate_student_grade(test_scores, homework_scores)}")
```

```
Overall average score: 82.1
Final Grade: B
```

1.7 Optional red: Take your grade calculator and make it a class!

[31]:
```
# "Grader"

class Grader:
    def __init__(self, test_weight=0.7, homework_weight=0.3):
        """Initialize the grader with default weights"""
        self.test_weight = test_weight
        self.homework_weight = homework_weight

    def calculate_average(self, test_scores, homework_scores):
        """Calculate weighted average of test and homework scores"""
        if not test_scores and not homework_scores:
            return None, None, None

        test_average = sum(test_scores) / len(test_scores) if test_scores else 0
        homework_average = sum(homework_scores) / len(homework_scores) if␣
 ↪homework_scores else 0
        overall_average = (test_average * self.test_weight) + (homework_average␣
 ↪* self.homework_weight)
        return test_average, homework_average, overall_average

    def percentage_to_letter_grade(self, percentage):
        """Convert percentage to letter grade"""
        if percentage >= 90:
            return "A"
        elif percentage >= 80:
            return "B"
        elif percentage >= 70:
            return "C"
        elif percentage >= 60:
            return "D"
        else:
            return "F"

    def generate_grade_report(self, test_scores, homework_scores, student_name):
        """Generate a full grade report for a student"""
        if not test_scores and not homework_scores:
            return f"No scores provided for {student_name}"

        test_avg, hw_avg, overall_avg = self.calculate_average(test_scores,␣
 ↪homework_scores)
        letter_grade = self.percentage_to_letter_grade(overall_avg)

        return f"""
=== Grade Report for {student_name} ===
Weighting: Tests ({self.test_weight*100:.0f}%) | Homework ({self.
  ↪homework_weight*100:.0f}%)
Test Average: {test_avg:.1f}%
Homework Average: {hw_avg:.1f}%
Overall Average: {overall_avg:.1f}%
Final Grade: {letter_grade}
"""

grader = Grader()  # Uses default weights (70% tests, 30% homework)

test_scores = [85, 93, 78, 96, 88]
homework_scores = [90]

print(grader.generate_grade_report(test_scores, homework_scores, "Aelin␣
  ↪Galathynius"))
```

```
=== Grade Report for Aelin Galathynius ===
Weighting: Tests (70%) | Homework (30%)
Test Average: 88.0%
Homework Average: 90.0%
Overall Average: 88.6%
Final Grade: B
```

1.7.1 Wuhuu - now you are all done. Impressive!

If you have time alter the function above so it outputs a neat dtudent grade report with test scores, test average, homework average, final grade and letter grade.