Week 3 - Loops, Conditions and Functions

Author: Johanne Sejrskild
Date: 12.08.2025

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!

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

```
Requirement already satisfied: lxml in /Users/sofiascharf-matthiesen/minicon
da3/lib/python3.12/site-packages (6.0.1)

[notice] A new release of pip is available: 24.3.1 -> 25.2
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

Green excercises

Conditional Statements

The *if* statement

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

```

In [ ]:
```
# Example of if statement
# Before running: What will happen and why?
# "x is greater than 5" will be printed as the condition (x > 5) is True.

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

```
x is greater than 5
```

In [ ]:
```
# Exceise: Modify the code to check if x is less than 5 and print a differen
x = 3
if x < 5:
    print("x is less than 5")
```

```
x is less than 5
```

The *else* statement

In [ ]:
```
# Basic syntax

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

In [ ]:
```
# Example of if-else statement
# Before running: What will happen and why?
# "x is not greater than 5" will be printed as the condition (x > 5) is Fals

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
```

In [ ]:
```
# Excercise - Change the code above so that it prints "x is greater than 5."
# How many solutions can you find? and is some of them better than others?

# solution 1: modify cutoff value
```

```
x = 3
if x > 2:
    print("x is greater than 5")
else:
    print("x is not greater than 5")

# this is not that cool, as the print statement is a lie now

# solution 2: modify x value
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")

# this is the best solution, as the print statement is still true
```

```
x is greater than 5
x is greater than 5
```

The *elif* statement (for multiple conditions)

In [ ]:
```
# Basic syntax

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

In [ ]:
```
# Example of if-elif-else statement
# Before running: What will happen and why?
# "x is equal to 5" will be printed as cond1 (x > 5) is False
# but cond2 (x == 5) is True.

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
```

In [ ]:
```
# Exercise - What happens when the condition for if and elif are the same? W
# if the condition is true, then only the first will be printed.
# as soon as first condition is met, the rest is skipped
```

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.

In [ ]:
```
# 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
```

In [ ]:
```
# Exercise - Modify the code to print the length of each fruit instead of th
# Example of for loop
for fruit in fruits:
    print(len(fruit))
```

```
5
6
6
```

In [ ]:
```
# 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
```

In [ ]:
```
# Excercise - Modify the code to count up from 1 to 5 instead of counting do
x = 1
while x <= 5:
    print(x)
    x += 1
```

```
1
2
3
4
5
```

Functions

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

In [ ]:
```
# Try it out with different numbers
add_two_numbers(3, 5)
```

```
3 + 5 = 8
```

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

In [ ]:
```
# Try it out with different numbers
subtract_two_numbers(3, 5)
```

```
3 - 5 = -2
```

Yellow excercises

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: ")**

In [ ]:
```python
# code here
def password_checker():
    password = input("What is your password:")
    # too short
    if len(password) < 8:
        print("Password is too short, must be at least 8 characters.")
    # too long
    elif len(password) > 32:
        print("Password is too long, must be no more than 32 characters.")
    else:
        print("Password is strong.")
```

```
Password is too short, must be at least 8 characters.
```

Now lets add some extra requirements:

- The password needs to have at least 3 digits
- And there needs to be an uppercase letter

In [ ]:
```python
# Modified code here
def password_checker():
    password = input("What is your password:")
    if len(password) < 8:
        print("Password is too short, must be at least 8 characters.")
    elif len(password) > 32:
        print("Password is too long, must be no more than 32 characters.")
    #  digits
    elif sum(c.isdigit() for c in password) < 3:
        print("Password must contain at least 3 digits.")
    # uppercase letters
    elif sum(c.isupper() for c in password) < 1:
        print("Password must contain at least 1 uppercase letter.")
    else:
        print("Password is strong.")
```

```
Password is strong.
```

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.

In [ ]:
```python
# Last modifications on the code - final password_checker
def password_checker():
    password = input("What is your password:")
    if len(password) < 8:
        print("Password is too short, must be at least 8 characters.")
    elif len(password) > 32:
        print("Password is too long, must be no more than 32 characters.")
    elif sum(c.isdigit() for c in password) < 3:
        print("Password must contain at least 3 digits.")
    elif sum(c.isupper() for c in password) < 1:
        print("Password must contain at least 1 uppercase letter.")
    # special characters
    elif sum(not c.isalnum() for c in password) < 1:
        print("Password must contain at least 1 special character.")
    else:
        print("Password is strong.")
```

```
Password is strong.
```

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

In [ ]:
```
# celsius to fahrenheit converter
def celcius2fahrenheit():
    c = input("Enter temperature in °C: ")
    f = (c * 9/5) + 32
    print(f"{c}°C is equal to {f}°F")
```

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

In [ ]:
```
# Fahrenheit to Celsius converter
def fahrenheit2celcius():
    f = input("Enter temperature in °F: ")
    c = (f - 32) * 5/9
    if f == 451:
        print(f"Celcius {c} is not as catchy a booktitle as Fahrenheit {f}."
    else:
        print(f"{f}°F is equal to {c}°C")
```

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.

In [ ]:
```
# Temperature converter here
def temperature_converter():
    # let them choose convertion direction
    choice = input("Enter F for fahrenheit to celcius conversion or C for ce

    if choice == "C":
        c = float(input("Enter temperature in °C: "))
        f = (c * 9/5) + 32
        print(f"{c}°C is equal to {f}°F")

    if choice == "F":
        f = float(input("Enter temperature in °F: "))
        c = (f - 32) * 5/9
        if f == 451:
            print(f"Celcius {c} is not as catchy a booktitle as Fahrenheit {
        else:
            print(f"{f}°F is equal to {c}°C")

temperature_converter()
```

```
Celcius 232.77777777777777 is not as catchy a booktitle as Fahrenheit 451.0.
```

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

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.

In [ ]:
```
# Make a function that calculate the average of test scores.
def calculate_avg(scores):
    total = sum(scores)
    avg = total / len(scores)
    return avg

# store average
test_scores = [85, 93, 78, 96, 88]
average_score = calculate_avg(test_scores)
```

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

In [ ]:
```
# Make a function that converst the test scores to letter grades

def percentage_to_letter_grade(perc):
    if perc>=100 or perc<0:
        return "Invalid percentage"
    if perc >= 90:
        return "A"
    elif perc >= 80:
        return "B"
    elif perc >= 70:
        return "C"
    elif perc >= 60:
        return "D"
    else:
        return "F"

# Test with different percentages
percentage_to_letter_grade(38)
```

Out[ ]:
```
'F'
```

Last task, We need to calculate the weighted grade

In [ ]:
```
def calculate_weighted_grade(test_avg, hmwk_score,
                             test_weight = 70, hmwk_weight = 30):

    # calc weighted avg of each
    test_ovall = test_avg * (test_weight / 100)
    hmwk_ovall = hmwk_score * (hmwk_weight / 100)

    final_grade = test_ovall + hmwk_ovall
    return final_grade
```

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.

In [ ]:
```
# now lets make a dictionary first to avoid too many inputs
def calculate_student_grade(test_scores, hmwk_score,
                             test_weight = 70, hmwk_weight = 30):
    # calculate test average
    test_avg = calculate_avg(test_scores)

    # final weighted
    final_grade = calculate_weighted_grade(test_avg, hmwk_score)
    # convert to letter grade
    letter_grade = percentage_to_letter_grade(final_grade)

    # returnnn
    return test_avg, final_grade, letter_grade

# try it out
test_scores = [85, 92, 78, 96, 88]
hmwk_score = 90.0

calculate_student_grade(test_scores, hmwk_score)
```

Out[ ]:
```
(87.8, 88.46, 'B')
```

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

In [ ]:
```
# Code here :)
class Grade_Calculator:

    def __init__ (self, test_scores, hmwk_score,
                  test_weight = 70, hmwk_weight = 30):

        # let's get them again
        self.test_scores = test_scores
        self.hmwk_score = hmwk_score
        self.test_weight = test_weight
        self.hmwk_weight = hmwk_weight

    def calculate_avg(self):
        total = sum(self.test_scores)
        avg = total / len(self.test_scores)
        return avg

    def percentage_to_letter_grade(self, perc):
        if perc>=100 or perc<0:
            return "Invalid percentage"
        if perc >= 90:
            return "A"
        elif perc >= 80:
            return "B"
        elif perc >= 70:
            return "C"
        elif perc >= 60:
            return "D"
        else:
            return "F"

    def calculate_weighted_grade(self, test_avg):
        # calc weighted avg of each
        test_ovall = test_avg * (self.test_weight / 100)
        hmwk_ovall = self.hmwk_score * (self.hmwk_weight / 100)

        final_grade = test_ovall + hmwk_ovall
        return final_grade

    def calculate_student_grade(self):
        # calculate test average
        test_avg = self.calculate_avg()

        # final weighted
        final_grade = self.calculate_weighted_grade(test_avg)

        # convert to letter grade
        letter_grade = self.percentage_to_letter_grade(final_grade)

        # returnnn
        return test_avg, final_grade, letter_grade

# try it out
test_scores = [85, 92, 78, 96, 88]
hmwk_score = 90.0
student = Grade_Calculator(test_scores, hmwk_score)
student.calculate_student_grade()
```

```
Out[ ]: (87.8, 88.46, 'B')
```

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.