SU-

Python customized

Exercises

Table of Contents

Module 0 Prepare your Lab environment

0.1 Establish a Python development environment

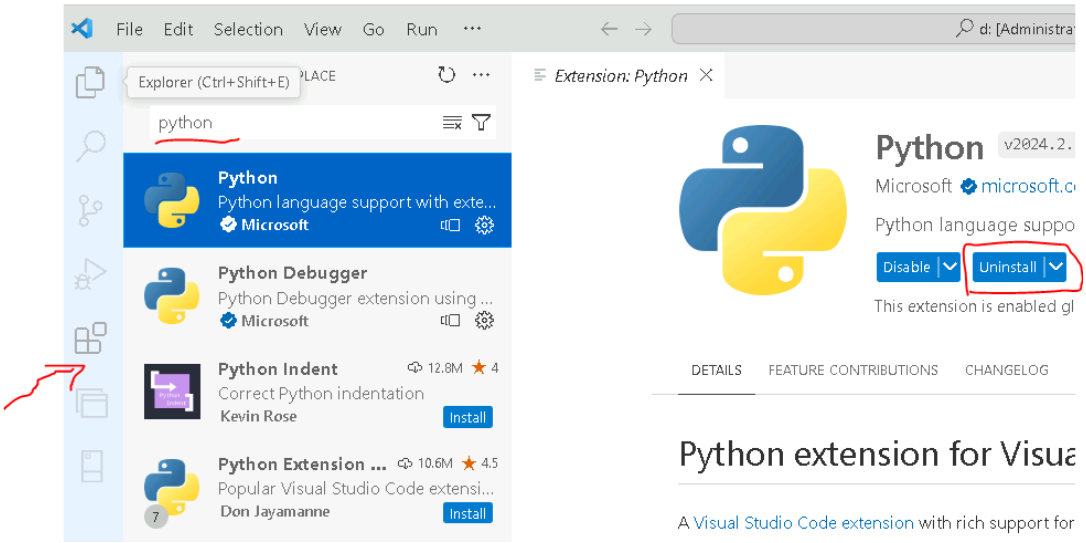
Follow the steps below to get ready for Python programming in this course:

1) Download and Install the latest version of Python from python.org on your lab machine. Use the Install now button to select a default installation.

2) Create a folder on your D drive called PythonCourse.

3) We will use VS Code as our IDE for the course. Open the Visual Studio Code application (Click on the Windows button and start typing vis – then the application will hopefully show up).

4) Click on the Extensions icon in the left vertical panel of VS Code. Search for Python and install the extension if it is not installed already.



5) Go back to the Explorer in VS Code (first item in left vertical panel) and create the following subfolders in your PythonCourse folder. (You can drag and drop if you by mistake create a folder inside another folder):
   - MondayProject

- Project2026

6) Create a new file in your MondayProject folder called **main.py** with a single line **print('Hello Python from VS Code editor!')**
7) Test that your newly installed Python interpreter works from VS Code by running your main.py script. Run your script using F5. Select the Python 3.13 interpreter.
8) Run the script again – this time with the Anaconda 3.11.7 interpreter. You can select the interpreter in the bottom right corner.
9) Write a function in your main.py file called is_leap_year. The function should take a year as input and return a bool value whether the year is a leap year (skudår) or not. Every 4th year is a leap year and 2024 was a leap year. This means if year is divisible by 4 – it's a leap year. Tip: Use the % (modulo) operator to get the remainder when you divide by 4.
10) Extend your function above so it includes the special rule: A century is only a leap year if it is divisible by 400 e.g. 1900 was not a leap year, but 2000 was. It's a famous bug in Excel that it regards 1900 as a leap year.
11) Test your function to print out years between 1970 and current year and show whether is was a leap year or not e.g.

1970 True
1971 False
1972 False
…
2025 False

Module 7 - Modules and packages.

7.1 Simple module, import and __name__

OEIS is an abbreviation of Online Encyclopedia of Integer Sequences, and the Fibonacci sequence is a famous integer sequence found in the nature e.g. in

the sun flower seed pattern.

Write a simple Python module: **myoeis.py** containing the function **Fibonacci** and save the file in your MondayProject folder.

```
# myoeis.py
def fibonacci(limit=10):
    a, b = 0, 1
    for i in range(limit):
        print(f" {a}", end='')
        a, b = b, a+b
```

Call the **fibonacci(7)** function inside myoeis.py and run myoeis.py to check that it displays: "0 1 1 2 3 5 8".

Implement a protection in myoeis.py which ensures that fibonacci(7) is only called when the module myoeis.py is run directly and not when imported.

Hint: **if __name__ == '__main__':**

Write another python script called main_program.py which **import myoeis** and call **fibonacci(9).** Run the script and check that it displays "0 1 1 2 3 5 8 13 21"

7.2 Simple package: my_utils

a. Create a directory named: **my_utils** to be used as package directory
Some of the files are found in folder with the exercises.

b. Create **within** the package two modules: **my_mod1.py** and **mod_mod2.py**

```
# my_utils/my_mod1.py
__all__ = ['my_func1']  # What to export when using import *

def my_func1():
    print('my_func1')
```

```
# my_utils/my_mod2.py
__all__ = [             # What to export when using import *
    'my_func2',
    #'my_func2b',
    ]

def my_func2():
    print('my_func2')

def my_func2b():
    print('my_func2b')
```

c. Create **within** the package the **__init__.py** to initialize the package.

```
# my_utils/__init__.py

from .my_mod1 import *
from .my_mod2 import *
```

d. Write a simple script called main_program.py to test the my_utils package

```
#
import my_utils as mu

mu.my_func1()
mu.my_func2()
#mu.my_func2b()
```

e. Make changes so my_func2b() becomes available and test again.
f. Copy paste the package folder and files to a new my_utils2 folder and change the code, so the syntax becomes mu.my_mod1.my_func1() to call func1 from my_mod1 module. Is the func2b function available?

Module 10 – Virtual environments

Module 10.1 Create a Python virtual environment for a project

In this exercise we will prepare a new Python virtual environment for a project where the pandas package is required.

VS Code supports virtual environments via conda or venv. We will use venv since it's free and included in a standard Python installation.

1) Open your pandasProject folder in VSCode
2) Install the Python Environment Manager extension (recently deprecated) or the Python Environments (new from Microsoft) extension following the guidance of your instructor.
3) Click on the Python icon in VS Code in the left vertical pane.
4) Create a new venv environment via the + icon to the right of the Venv folder under GLOBAL ENVIRONMENTS
5) Pick your global Python interpreter e.g Python 3.13.x from the box that pops up. Wait for the creation of your environment.
6) Your new environment will now be visible as a .venv (3.13.x) folder under Venv. You will also find a .venv folder in your pandasProject folder in the Explorer. Browse the .venv folder to see that pip.exe is found in the Scripts subfolder.
7) Click on the Shell command icon for your newly created .venv
8) Type **pip install pandas** This will install pandas in your environment.
9) Study the Packages which have now been installed in your workspace environment.
10) Run the command **pip freeze** in the Shell and notice the version nr for each package.
11) Type **pip install matplotlib.** This will install matplotlib and any dependant packages.
12) Run **pip freeze** again and study the list of packages.
13) Now run **pip freeze > requirements.txt** in the Shell. This will generate a text file in your current working directory with the output from the pip freeze command. (Greater than just means pipe to text.file)

14) Look at your requirements file in the explorer.

Module 10.2 Create additional Python environments

VS Code supports multiple virtual environments and it's easy to switch between them. In this exercise we will create two environments and learn to switch back and forth between them.

1) Open the D:\PythonCourse folder in VS Code
2) Create a .venv environment with the Venv + icon in the Python Environment GUI. This will always create an environment called .venv and that is the standard way to do it.
3) We want an extra environment, so we have to come up with another name for it. Let's call it venv313custom. Make sure you stand in the D:\PythonCourse folder. Then type **py -m venv ven313custom** in the Terminal in VS Code.
4) Check that you now have two different venv environments in the Venv folder.
5) Create a script called current_version.py in a folder called Lab10.
6) Write info on the screen about the current python exe file via **print('I am running via ', sys.executable)**
7) Run your script in VS Code and notice you can switch interpreter in the right left corner. It has probably chosen the .venv. Try to run both of them and compare the output.

.1 Simple module, import and __name__

```
# myoeis.py
# http://oeis.org
# Fibonacci numbers
# Fibonacci numbers: F(n) = F(n-1) + F(n-2) with F(0)=0 and F(1)=1

def fibonacci(limit=10):
    a, b = 0, 1
    for _ in range(limit):
        print(f" {a}", end='')
        a, b = b, a+b


if __name__ == '__main__':
    fibonacci(7)
```

```
import myoeis

myoeis.fibonacci(9)
```

7.2 Simple package: my_utils

```
# my_utils/__init__.py

from .my_mod1 import *
from .my_mod2 import *
```

```
# my_utils/my_mod1.py
__all__ = ['my_func1']

def my_func1():
    print('my_func1')
```

```
# my_utils/my_mod2.py
__all__ = [
    'my_func2',
    #'my_func2b',
    ]

def my_func2():
    print('my_func2')

def my_func2b():
    print('my_func2b')
```

```
#
import my_utils as mu

mu.my_func1()
mu.my_func2()
#mu.my_func2b()
```