

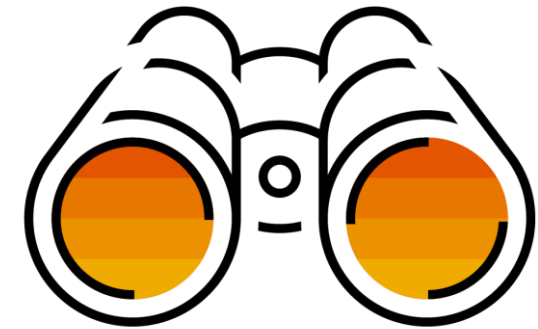


Week 0: Getting Ready

# Unit 1: Welcome – How This Course Works

### Introduction and welcome

- Welcome to the course “Python for Beginners”
- This course addresses real beginners in programming.
  - If you’ve never implemented a program but are curious about what *programming* is all about, then this course is aimed at you.
  - If you’ve learned to write programs in another computer language but are interested in getting an introduction to Python, then this course is great for you as well.
- The course covers the basics in Python like variables, data types, and control structures.
  - These concepts are introduced week by week.
    - It is explained how they are used in Python
    - Examples are given to support your understanding.



## To learn programming requires practice

- You cannot learn programming by just following a video or reading a book.
- If you want to learn programming, you have to program, program, program ...
  - This is like learning to play the piano. You cannot learn to play the piano by reading a book about it. You have to practice.
- For this reason, the course offers a lot of larger and smaller programming exercises.
- We expect you to practice, to go through the programming exercises, and to implement Python programs.
- It's best to do this on a weekly basis.
  - To learn programming requires regular repetition.
- Consequently, there are mandatory programming exercises to pass the course and gain a certificate.



Welcome – How this course works

**Each week focuses on a specific topic**

Week 1: **Python Fundamentals:** Statements, variables, data types, and the “if” statement

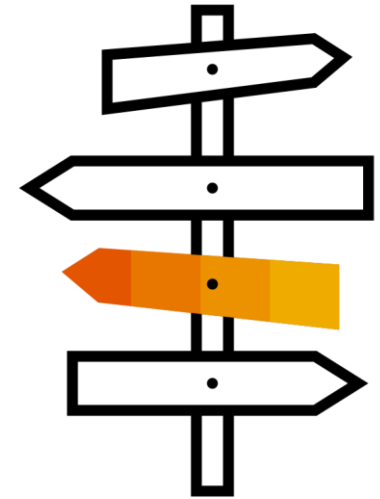
Week 2: **Lists and Loops:** Working with sequences and the “for” loop

Week 3: **Complex Data Types:** More data types and the “while” loop

Week 4: **Reading and Writing Data:** Handling files

Week 5: **Functions:** Reusing code and structuring your programs

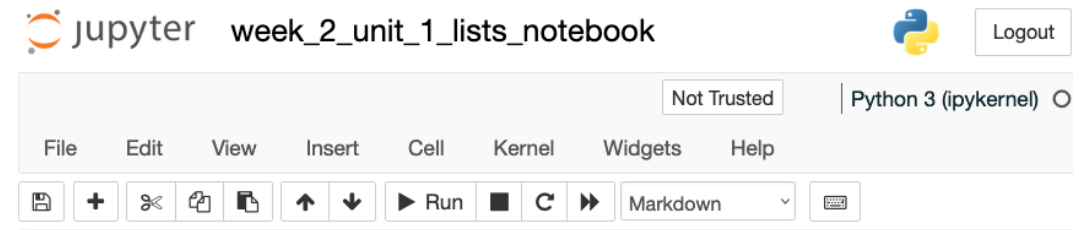
Week 6: **Libraries:** Finding and using functions written by others



Welcome – How this course works

## The main learning materials are Jupyter Notebooks

- Each week is split into several units, and each unit starts with a few slides, to explain the content of the unit
- However, the full content with explanations and examples is contained in Jupyter Notebooks
- What are Jupyter Notebooks?
  - Jupyter Notebooks combine documentation and the possibility to program in one place
  - Thus, Notebooks are perfect for teaching and learning, because all you need is in the same place
  - We will explain how to install the Jupyter Notebooks software in unit 3
  - We will explain how to work with a Jupyter Notebook in unit 4



### Lists in Python

A `list` is one of the [sequence types](#) in Python. We will learn about other sequence types in subsequent units.

In Python square brackets (i.e. `[` and `]`) are used to create a list. The individual items of the list are separated by commas. Just like primitive data types, lists can be assigned to variables. The following example shows a simple list containing different integer values. This list is assigned to the variable `my_number_list`. Finally, the variable `my_number_list` is printed using the `print()` function.

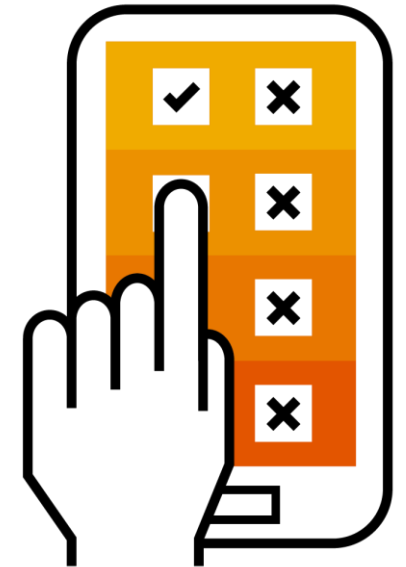
```
In [1]: 1 my_number_list = [0, 1, 1, 2, 3, 5, 8, 13, 21]
        2 print(my_number_list)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21]
```

## Welcome – How this course works

### Quizzes, mandatory exercises, and additional exercises

- All units contain a quiz or a short programming exercise.
  - These quizzes and exercises can be done as often as you like.
  - You don't need to complete them to get the course certificate.
  - We'll provide example solutions for these exercises.
- At the end of each week there is a **weekly assignment**.
  - These assignments consist of a questionnaire and a programming exercise.
  - You **must complete** these assignments **to get the certificate**.
  - We'll provide commented example solutions by beginning of the following week.
- We also offer *additional* exercises.
  - These do not count towards the certificate.
  - They aim to provide additional examples to extend the material for the week.
  - Again, commented example solutions will be provided.
- The **final exam** consists of a **questionnaire** and **programming exercises**.



Welcome – How this course works

## What can you expect from us? What do we expect from you?

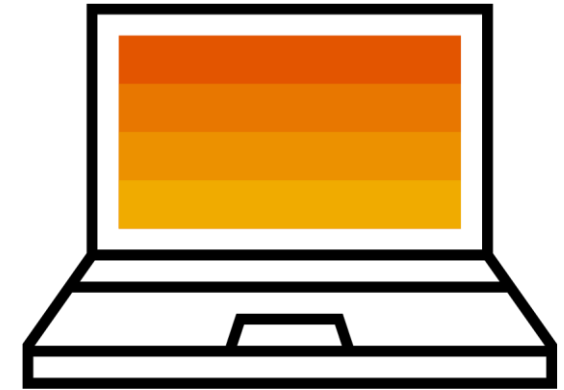
- If you participate in the course you can expect the following from us:
  - New learning material (notebooks, videos, ...) every week
  - Additional notebooks with additional exercises and example solutions
  - Most example solutions will be explained in a video
  - You can look over our shoulder while we're solving the exercises (Actually, observing experienced programmers is a good way of learning programming)
  - We'll try to answer questions in the forum as quickly as possible
- What we expect from you
  - You should try to implement the exercises yourself
  - You have to hand in the weekly assignment
  - You should actively participate in the forum
  - You should not hesitate to use other tutorials or learning materials. Lots of material related to Python can be found on the Internet



## Welcome – How this course works

### You want to use your own IDE?

- If you can't use Jupyter Notebooks or would simply like to use another IDE (integrated development environment), of course you can do so.
  - However, we cannot provide support for handling IDEs
- For the programming assignments, you have to upload your programs to *CodeOcean*, which will be explained in the next unit.





# Thank You!

**Contact Information:**  
[open@sap.com](mailto:open@sap.com)



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

hosted by  
**openSAP**  
[open.sap.com](https://open.sap.com)



# Week 0: Getting Ready

## Unit 2: CodeOcean

## CodeOcean is the system to evaluate your coding exercises

- CodeOcean is the system we use for grading the coding exercises.
  - For each of the exercises, we have some tests to check if your solution is correct.
  - CodeOcean provides feedback if there are problems with your solution.
- It is quite easy to use CodeOcean (cf. the following page).
  - For each exercise in the course, we provide a link to the exercise in CodeOcean.
- Actually, CodeOcean is a small environment to develop code.
  - You can enter your code, execute the code, observe the output, and get feedback in case of errors.
- However, we recommend to solve the exercises in the notebooks and copy and paste the solution for grading into CodeOcean.









## CodeOcean is easy to use

- The highlighted area (yellow) is used for coding.
- Here you can directly type in your code.
- Again, we recommend to first solve the program in the notebook and then copy your solution into CodeOcean.
- If you click the “Run” button, the code will be executed (cf. next page).

Implement two methods even and odd which return whether a given number is even or odd, respectively.

(Hide)

 Collapse  
Action Sidebar

Files    
 exercise.p  
 

 Run

 Score

 Request Comments



```
1 def even(x):
2     if x % 2 == 0:
3         print(x, "seems to be even")
4         return True
5     else:
6         print(x, "seems to be odd")
7         return False
8
9 def odd(x):
10     if x % 2 == 0:
11         return False
12     else:
13         return True
14
15 print(even(23))
16
```

## You can execute your code in CodeOcean

- The output of your program is shown in the highlighted field on the right.
- So far, the solution has NOT been handed in for evaluation!
- To do so, use the “Score” button (cf. next page).

EXERCISES / EVEN/ODD / IMPLEMENT

✓ Even/Odd

100%

Implement two methods even and odd which return whether a given number is even or odd, respectively.

(Hide)

Keyboard shortcut: ALT + r

Collapse Action Sidebar

Files ↓ ↺  
exercise.p

Stop

Score

Request Comments

```
1 def even(x):  
2     if x % 2 == 0:  
3         print(x, "seems to be even")  
4         return True  
5     else:  
6         print(x, "seems to be odd")  
7         return False  
8  
9 def odd(x):  
10     if x % 2 == 0:  
11         return False  
12     else:  
13         return True  
14  
15 print(even(23))  
16
```

Collapse Output Sidebar

23 seems to be odd  
False

# Your code is automatically evaluated

- Once you have handed in your program for scoring, your solution is evaluated, i.e. it is run against our tests.
- As you can see in the highlighted part of the figure, two tests were passed, and the score of one point is given.
- You can score your solution multiple times in order to fix previous errors.

Keyboard shortcut: ALT + s

(Hide)

Run Score Request Comments

Collapse Output Sidebar

```
1 def even(x):
2     if x % 2 == 0:
3         print(x, "seems to be even")
4         return True
5     else:
6         print(x, "seems to be odd")
7         return False
8
9 def odd(x):
10     if x % 2 == 0:
11         return False
12     else:
13         return True
14
15 print(even(23))
16
```

## Results

1 test files have been executed.

### Test File 1 (exercise\_tests.py)

<b>Passed Tests</b>	2 out of 2
<b>Score</b>	1 out of 1
<b>Feedback</b>	Well done. All tests have been passed.
<b>Error Messages</b>	

Score: 100%

## Finally, submit the solution to openSAP

- If you are fine with the score you have received, submit your solution to openSAP by pressing the “Submit Code for Assessment” button.
  - In the right sub-window, scroll down until the button appears.
  - Click the button.
- Important: You MUST NOT go back with the Back button of your browser.
  - If you want to redo the exercise, close the browser tab/window and start the exercise again from the course.

Complete the game “Guess The Number”.

The computer chooses a number between 1 and 100 (both included). Afterward, the player should try to guess the number skillfully.

If the guessed number is too low, “too small” should be printed. If the guessed number is too high, “too large” should be output. If the number is correct, the player should be congratulated and then the game should be ended.

Implement the missing code snippets marked by a ‘# ToDo’ in the source code. Also, pay attention to the programming style and satisfy the linter.

Keyboard shortcut: ALT + s

Collapse Action Sidebar

Files

default.pylintrc

guess\_the\_number.py

test\_check\_number.py

test\_main.py

test\_read\_number.py

test\_style.py

Tips

Tip 1: Input

Run

Score

Request Comments

```
9      # until the number is guessed correctly
10     . Use the `check_number`
11     # method, whose logic should also be
12     implemented, to verify the input.
13
14     guessed_number = read_number()
15
16     print('Would you like to play again?
17     Why not start another round right away?')
18
19     def check_number(guessed_number, correct_number):
20         # ToDo:
21         # Implement code here that determines
22         whether the `guessed_number` matches the
23         # `correct_number`. Also, make sure
24         that a message is issued whether the
25         # guessed number was `too small`, `too
26         large` or `correct`.
27
28     return False
```

Last saved: 14:23:10 | [Reset this file](#)

Collapse Output Sidebar

Score: 60%

Submit Code For Assessment

StdErr: test\_correct\_does\_not\_print\_incorrect\_strings (te

test\_correct\_guess\_prints\_correct\_string (test\_check\_num

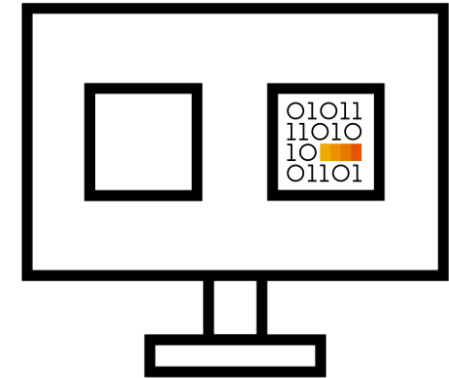
test\_too\_big\_does\_not\_print\_incorrect\_strings (test\_check

test\_too\_big\_prints\_correct\_string (test\_check\_number.Ass

test\_too\_small\_does\_not\_print\_incorrect\_strings (test\_che

test\_too\_small\_prints\_correct\_string (test\_check\_number.A

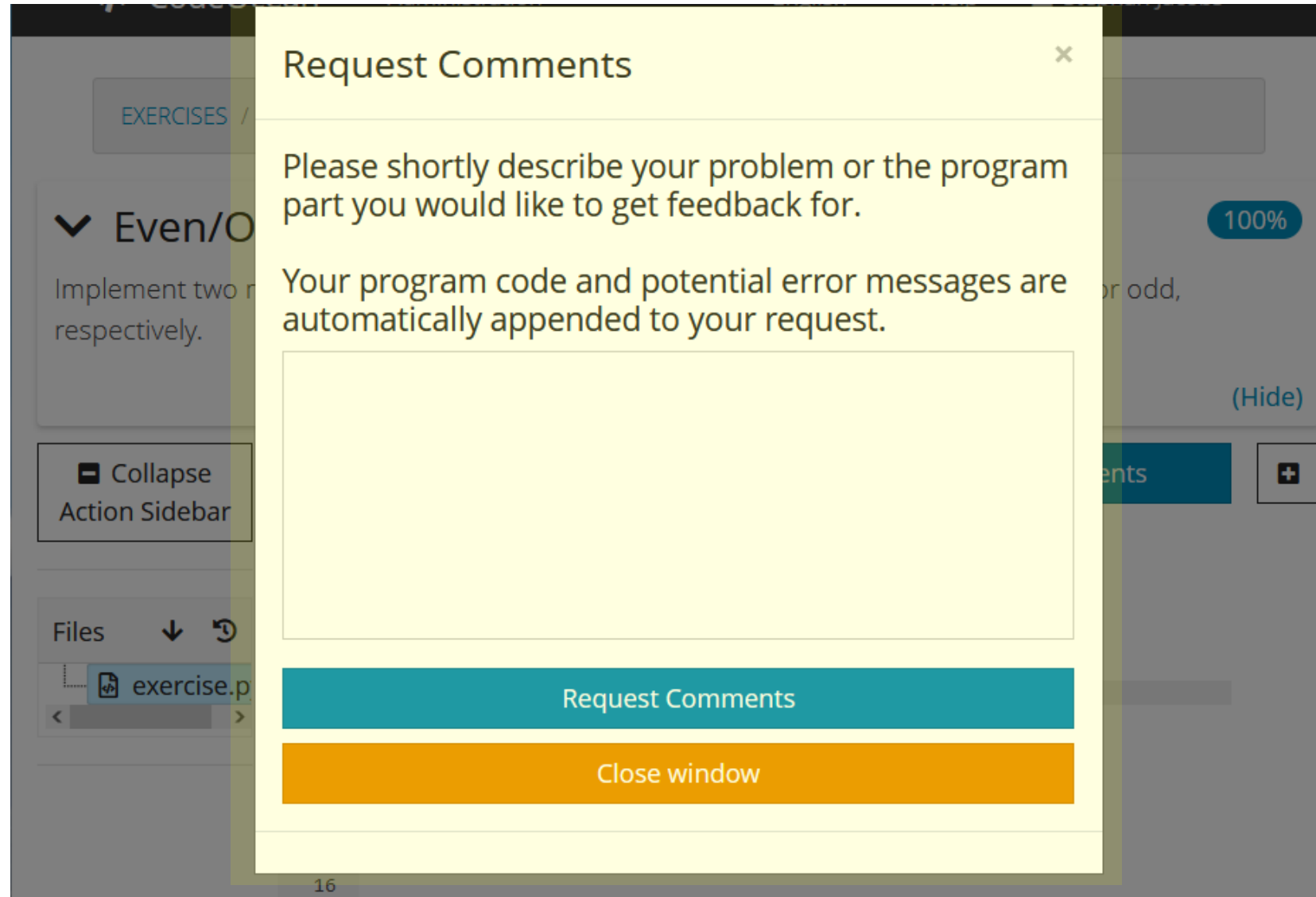
- What are the differences between these actions?
  - **Run:** Executes the code. The output of the code is shown in the right sub-window. If the execution produces errors, these are shown as well.
  - **Score:** Executes some test cases. Potential output from the test cases is presented in CodeOcean. In addition, you can see how many points the scored solution gets. The points are not yet forwarded to the course in openSAP.
  - **Submit:** The program is executed, the points are evaluated, and the result is forwarded to openSAP. Now the points are awarded to you in the course.
- You can redo all three actions as often as you like. But ...
  - Before you re-submit your code, you have to first close the TAB and open it again by clicking on the exercise from the course in openSAP.
  - Submission can only be done **before** the deadline.





## CodeOcean supports cooperation between participants

- In case of problems, you can request feedback.
- This request is visible to other participants, who can give feedback.
  - To have a look at your requests (and follow up on the feedback by others), click on your name (upper right corner) and go to the requests.
- Alternatively, you can post your questions in the forum.
  - Don't forget to include the number of the exercise.



## CodeOcean is really easy to use

- You will be directed to CodeOcean to hand in your solution for the programming exercises.
- We recommend to first solve the exercises in Notebooks.
  - Once you are fine with the solution, go to CodeOcean.
- Copy your code into the input field, then run the code.
- If everything looks fine, hand in the code for scoring.
- If the score evaluation is okay, submit the code to get the points in the course.
- If there are tests which are not passed, try to fix the code in the notebook and start again.
- You can run, score, and submit the code as often as you like.
- Use the possibility to request comments for support, if required.



# Thank You!

**Contact Information:**  
[open@sap.com](mailto:open@sap.com)



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

hosted by  
**openSAP**  
[open.sap.com](https://open.sap.com)



Week 0: Getting Ready

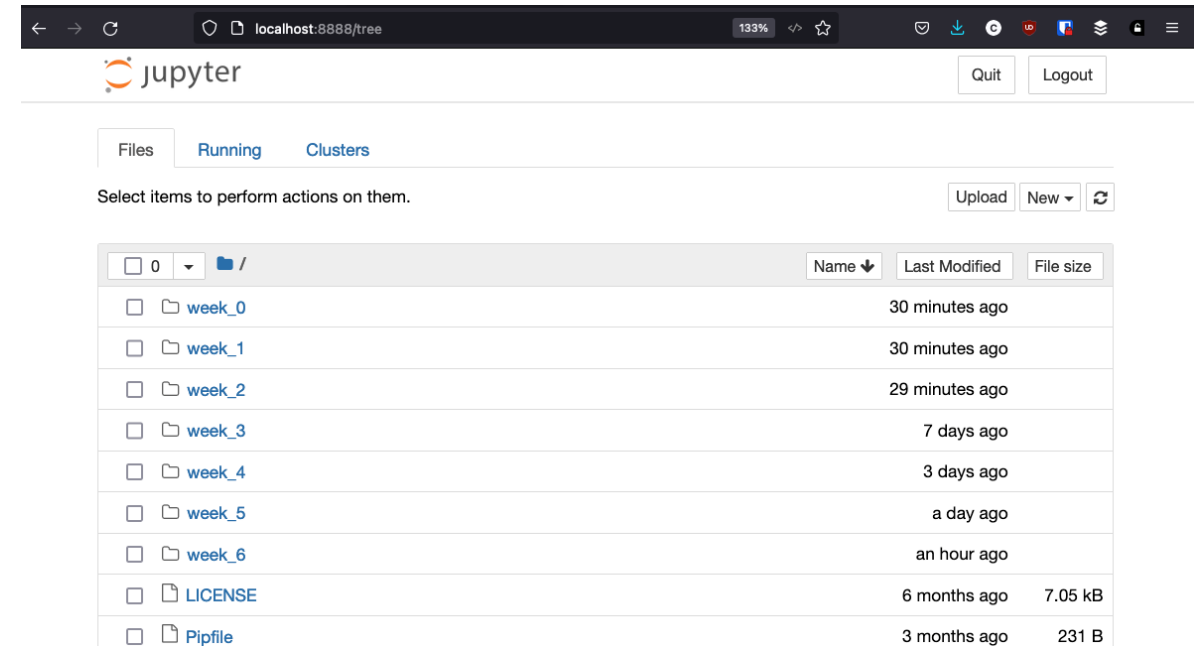
# Unit 4: How to Use Jupyter Notebooks

# How to use Jupyter Notebooks

## Now it's time to start a Jupyter Notebook

1. Open a shell (cmd-shell, PowerShell, or Terminal)
2. Type “jupyter notebook” and hit enter ↵
  - The Jupyter server starts
  - It opens either a new browser or a new tab in an already running browser (cf. figure)
  - The shown content depends on your computer.

Depending on your installation of Python and Jupyter Notebook there may be other options to start the Jupyter server.



### Navigate to your course directory and open the notebook

- At this point, you should think about where to store all the course material.
  - We recommend creating a folder dedicated to this course.
  - All the course material should be stored in this folder (notebooks, pdfs, exercises, notes, ...).
  - You could name the folder e.g. “open-sap-python”
- Download the Jupyter Notebook called “openSAP\_python1\_week\_0\_unit\_4\_helloworld.ipynb” and store it in the folder you just created.
- Now go back to the Jupyter tab in your browser, navigate to this folder and open it by clicking on it.
- Your browser should look like the screenshot on the right.

#### Introduction to Jupyter Notebooks

We will use Jupyter Notebooks for the introduction to Python programming. The Jupyter Project (<https://jupyter.org/>) develops open source software that can be used to create special documents called Jupyter Notebooks. These documents can contain:

- programme code
- formatted text
- equations
- Visualisations

The notebooks are therefore very suitable for learning a programming language, such as Python. In practice, Jupyter Notebooks are used for applications in data cleaning and transformation, numerical simulation, statistical modelling, data visualisation and machine learning. You can see this practical relevance in the fact that virtually all cloud providers offer options for hosting Jupyter Notebooks (Microsoft: (<https://notebooks.azure.com/>), Google: (<https://cloud.google.com/datalab/>)).

Within the MOOC "Python for Beginners" we will provide you with Jupyter Notebooks. These notebooks contain the theoretical background of the lecture as well as the exercises.

#### Jupyter Notebooks are based on Cells

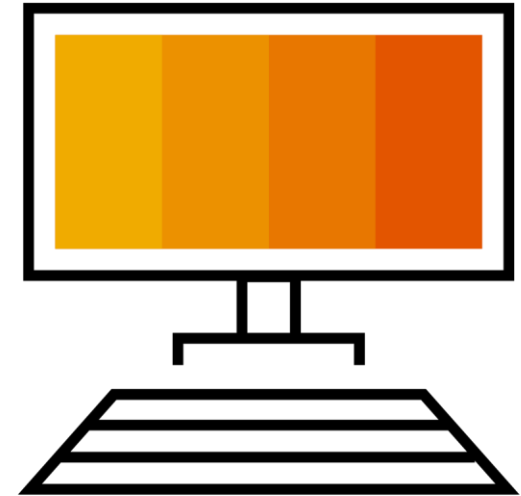
Jupyter Notebooks are based on cells. There are two types of cells, which are interesting for this MOOC:

- Cells containing text and explanations (Markdown Cells)
- Cells to do the programming (Code Cells)

Actually, the text you are just reading is part of a Markdown Cell!

# Congratulations, your workplace is prepared to start the course

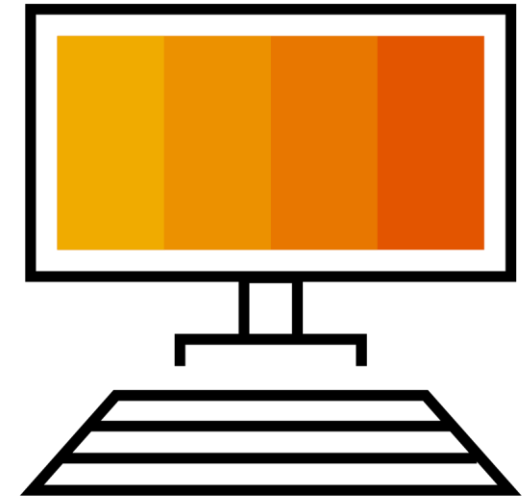
- Now you're ready to start the course.
- There's one important note:
  - Due to the myriads of different combinations of hardware, operating systems, installed software on the computer, configurations, settings ... we are not able to provide support for the installation of Python and Jupyter Notebook
  - In case of problems, please try to find a solution yourself.
- Starting with week 1, we assume that Python and Jupyter are installed and working properly.



## How to use Jupyter Notebooks

### **The course is based on Jupyter Notebooks**

- Nevertheless, each unit starts with some slides to introduce the following content.
- After this introduction, the course continues with the notebook. If you are following the course, you have to switch from pdf to notebook. This switch is always introduced by the following slide (cf. next page)





# How to use Jupyter Notebooks

## Showtime

Now it's time to get hands on and start programming!

If you like, you can open the [Jupyter Notebook](#) instructions in parallel to the demo.

If you haven't done so yet:

- [Download the Notebook](#)
- [Start the Jupyter Server](#)
- [Open the Notebook](#)



```
Jupyter Showtime_2
```

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel) Logout

Run Code

## Showtime 🎉

It's time to get hands on and start programming!

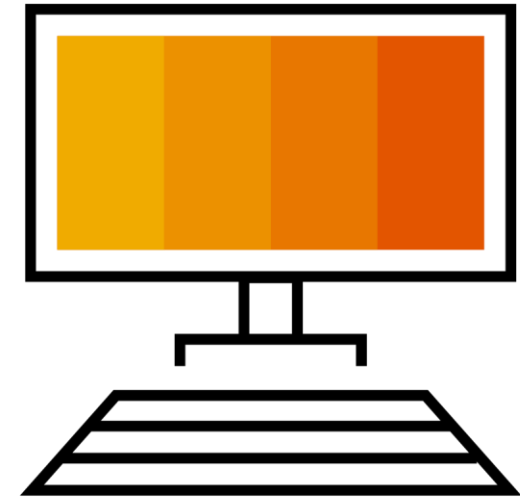
```
In [1]: 1 for i in range(3, 0, -1):
        2     print("...", i)
        3     print("🎉 Showtime 🎉")

... 3
... 2
... 1
🎉 Showtime 🎉
```

# This course is based on Jupyter Notebooks

Let us have a closer look at notebooks:

- Our notebooks consist of two types of cells:
  - Markup cells which contain explanations
  - Code cells which contain programs
- The notebooks are interactive, i.e. you can (and have to) start programming yourself.
- The cells have two modes:
  - In *edit mode* you can implement or change programs
  - In *command mode* you can execute the program in the cell.
- Let's have a look. Let's go into a notebook ...



# How to use Jupyter Notebooks

## Showtime

Now it's time to get hands on and start programming!

If you like, you can open the [Jupyter Notebook](#) instructions in parallel to the demo.

If you haven't done so yet:

- [Download the Notebook](#)
- [Start the Jupyter Server](#)
- [Open the Notebook](#)



**Showtime** 🎉

It's time to get hands on and start programming!

```
In [1]: 1 for i in range(3, 0, -1):
        2     print("...", i)
        3     print("🎉 Showtime 🎉")

... 3
... 2
... 1
🎉 Showtime 🎉
```

# Thank You!

**Contact Information:**  
[open@sap.com](mailto:open@sap.com)



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

hosted by  
**openSAP**  
[open.sap.com](https://open.sap.com)