

INSTALLATION ANACONDA UND VS CODE

IDE – Entwicklungsumgebung

Visual Studio Code: Erweiterungen



Über Erweiterung müssen zunächst diverse „Extension Packs“ installiert werden (falls noch nicht vorhanden):

Bitte installieren Sie dazu folgende Software Packages im Voraus auf das Notebook, diese sind beide Lizenzen frei:

- Visual Studio Code von Microsoft:
<https://code.visualstudio.com/download>
- Anaconda, beinhaltet Python Runtime und eine Python Notebook Oberfläche
<https://www.anaconda.com/download>

Oder

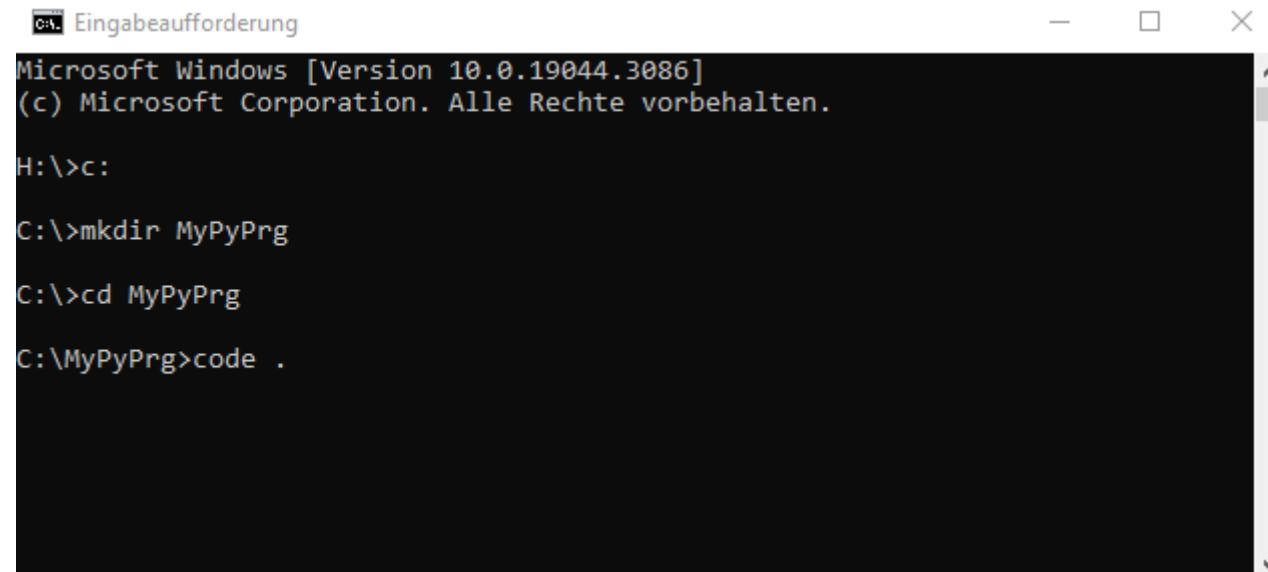
- Miniconda, ist eine kleine Version von Anaconda
<https://www.anaconda.com/download/success#miniconda>

Achtung:
Anaconda oder Miniconda am besten in ein höheres Directory installieren:
z.B. C:\temp\Miniconda

Setup

Anlegen eines Projekts

1. Öffnen Sie cmd



```
Eingabeaufforderung
Microsoft Windows [Version 10.0.19044.3086]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

H:\>c:

C:\>mkdir MyPyPrg

C:\>cd MyPyPrg

C:\MyPyPrg>code .
```

2. Gehe nach C:\

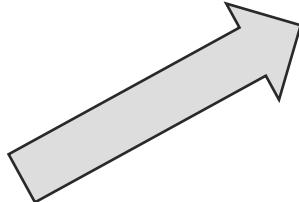
3. Erstellen ein dir

```
C:\> mkdir myPyPrg
C:\>mkdir MyPyPrg
C:\>cd MyPyPrg
C:\MyPyPrg>code .
```

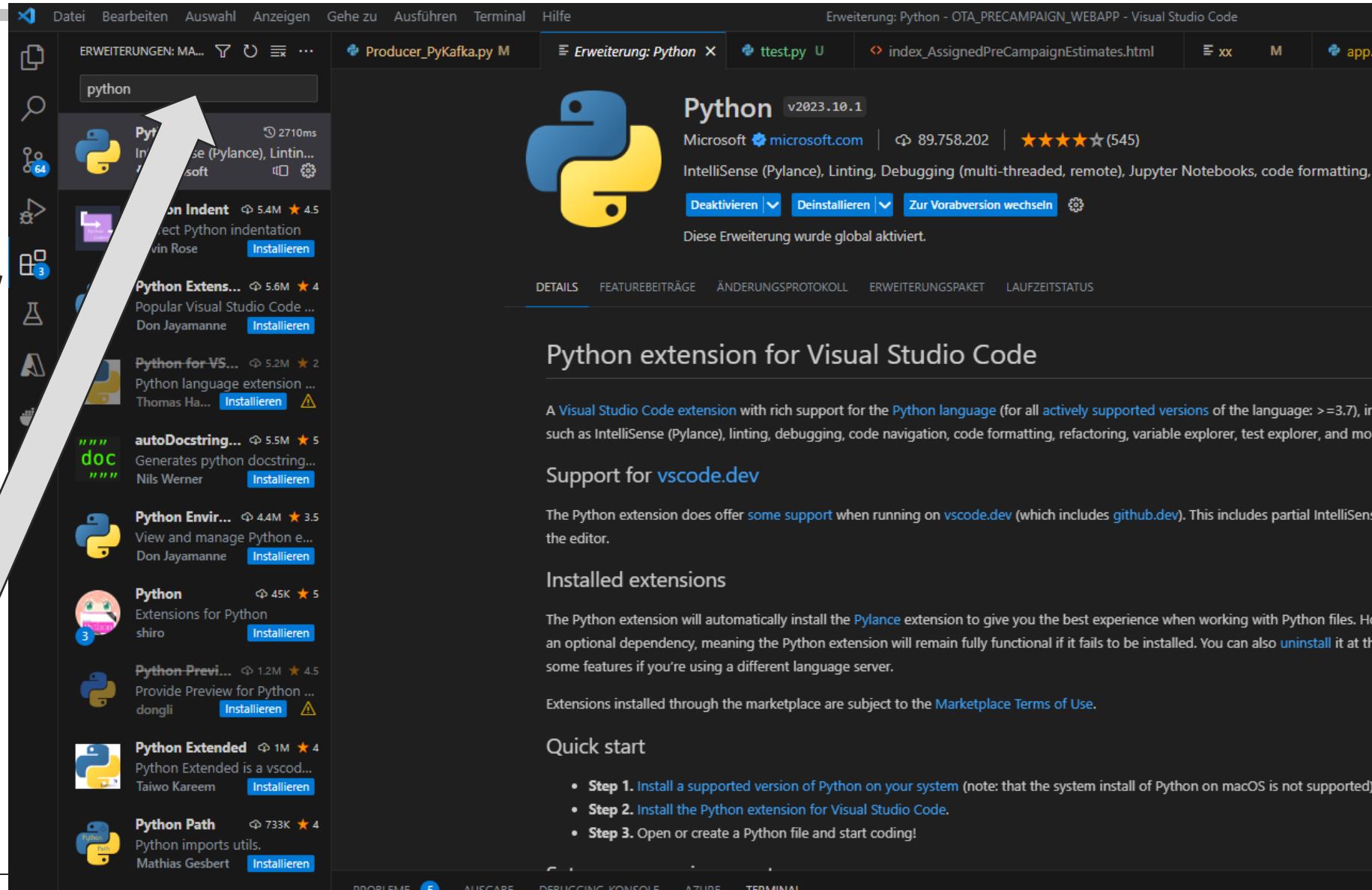
Code Setup

Install Python Extension

1. Selektiere 5 Icon



2. Filter Python
3. Installation von
Python Extension



The screenshot shows the Visual Studio Code interface with the Python extension installed. The top navigation bar includes Datei, Bearbeiten, Auswahl, Anzeigen, Gehe zu, Ausführen, Terminal, and Hilfe. The title bar indicates the extension is Python - OTA_PRECAMPAIGN_WEBAPP - Visual Studio Code. The main area shows a list of extensions under 'ERWEITERUNGEN: MA...', with 'python' selected. The Python extension by Microsoft is highlighted, showing it's version 2023.10.1, has 89.758.202 downloads, and a 5-star rating. It provides IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, and code formatting. Buttons for Deaktivieren, Deinstallieren, and Zur Vorabversion wechseln are shown. Below the extension details, sections for DETAILS, FEATUREBEITRÄGE, ÄNDERUNGSPROTOKOLL, ERWEITERUNGSPAKET, and LAUZEITSTATUS are visible. A large Python logo is displayed on the right.

Python v2023.10.1

Microsoft [microsoft.com](#) | 89.758.202 | ★★★★☆(545)

IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting...

Deaktivieren Deinstallieren Zur Vorabversion wechseln

Diese Erweiterung wurde global aktiviert.

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.7), including such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorer, and more.

Support for vscode.dev

The Python extension does offer some support when running on [vscode.dev](#) (which includes [github.dev](#)). This includes partial IntelliSense support in the editor.

Installed extensions

The Python extension will automatically install the [Pylance](#) extension to give you the best experience when working with Python files. However, Pylance is an optional dependency, meaning the Python extension will remain fully functional if it fails to be installed. You can also [uninstall](#) it at the moment if you no longer need it or if it's causing issues. Some features may be disabled or have limited functionality if Pylance is not installed, but most core features should still work.

Extensions installed through the marketplace are subject to the [Marketplace Terms of Use](#).

Quick start

- Step 1. Install a supported version of Python on your system (note: that the system install of Python on macOS is not supported)
- Step 2. Install the Python extension for Visual Studio Code.
- Step 3. Open or create a Python file and start coding!

Code Setup

Select Python Interpreter

1. Press Strg+Shift+P

2. Filter python

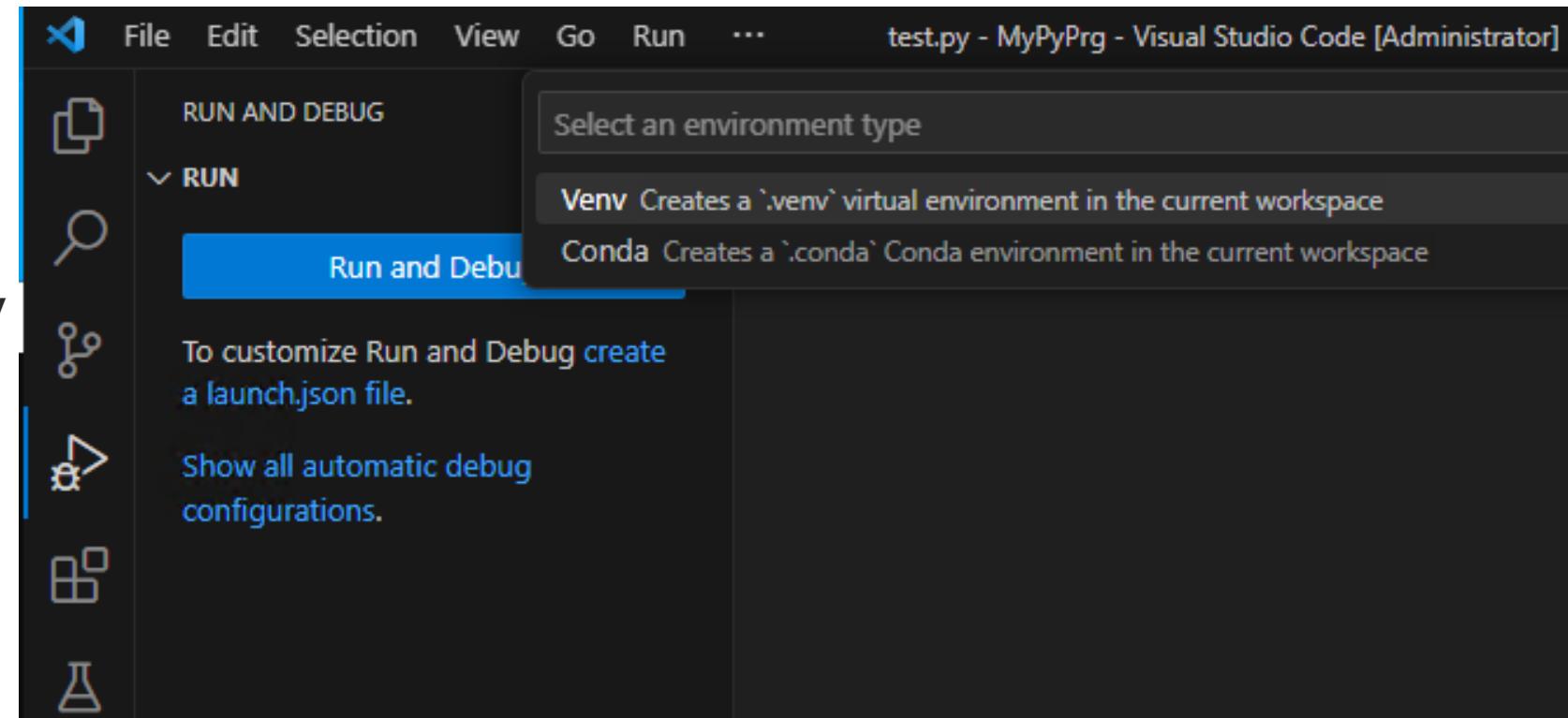
The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** Datei, Bearbeiten, Auswahl, Anzeigen, Gehe zu, Ausführen, Terminal, Hilfe.
- Explorer View:** Shows a tree structure of files and folders. Opened files include "Producer_PyKafka.py", "ttest.py", and "diff.py".
- Code Editor:** Displays the content of "diff.py".
- Command Palette:** A floating window titled "Python: Select Interpreter" is open, showing a list of available interpreters. The first item in the list is "Python: Interpreter auswählen".
- Code Content:** The "diff.py" code is as follows:

```
dir_local = os.path.dirname(os.path.abspath(__file__))
dir_final = pd.read_excel(dir_local + "E_HU_01.xlsx")
df_newEstimates = pd.read_excel(dir_local + "E_HU_02.xlsx")
notin=0
innewEstimates=0
df_fehlen=pd.DataFrame()
for index, row in df_final.iterrows():
    df_new = df_newEstimates[df_newEstimates['FIN'] == row['FIN']]
    if len(df_new) > 1:
        print(df_new)
    if df_new.empty:
        #print(row.to_dict())
        df_new2 = df_final[df_final['FIN'] == row['FIN']]
        df_new2=pd.DataFrame(row.to_dict(), index=[0])
        #print(df_new2)
        df_fehlen=df_fehlen.append(df_new2)
        print(row.FIN, " not in new Estimates")
        #print(df_fehlen)
        notin+=1
    else:
        #print(row.FIN, " included new Estimates")
        innewEstimates+=1
print("Enthalten: ", innewEstimates)
```

Create Venv

1. Press **Strg+Shift+P**
2. Create Virtuel Env
3. Select „Venv“
4. Check if python is selected with new Venv Path



Code Setup

Select Python Interpreter

1. Press Strg+Shift+P

2. Filter python

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** Datei, Bearbeiten, Auswahl, Anzeigen, Gehe zu, Ausführen, Terminal, Hilfe.
- Explorer View:** Shows a tree structure of files and folders. Opened files include "Producer_PyKafka.py", "ttest.py", and "diff.py".
- Code Editor:** Displays the content of the "diff.py" file.
- Command Palette:** A floating window titled "Python: Select Interpreter" is open, showing a list of available interpreters. The first item in the list is "pyth".
- Status Bar:** Shows "diff.py - OTA_PR...".

```
diff.py - OTA_PR...
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

Content of diff.py:

```
ur_final=pd.read_excel(dirlocal+E_HU)
df_newEstimates=pd.read_excel(dirlocal+E_HU)
notin=0
innewEstimates=0
df_fehlen=pd.DataFrame()
for index,row in df_final.iterrows():
    df_new = df_newEstimates[df_newEstimates['FIN'] == row['FIN']]
    if len(df_new) > 1:
        print(df_new)
    if df_new.empty:
        #print(row.to_dict())
        df_new2 = df_final[df_final['FIN'] == row['FIN']]
        df_new2=pd.DataFrame(row.to_dict(),index=[0])
        #print(df_new2)
        df_fehlen=pd.concat([df_fehlen,df_new2])
        print(row.FIN," not in new Estimates")
        #print(df_fehlen)
        notin+=1
    else:
        #print(row.FIN," included new Estimates")
        innewEstimates+=1
print("Enthalten: ", innewEstimates)
```

IDE – Entwicklungsumgebung

Visual Studio Code



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of files under "PYTHON". The file "06_factorial2.py" is selected.
- Editor:** Displays the Python code for "06_factorial2.py".

```
n = 12
count = 0
# while-Schleife: solange Kriterium erfüllt ist, wird Schleife durchlaufen
while count < 10:
    # The next is a smart version of print(str(n1) + ' , ')
    n = n + count
    print(n)
    count = count + 1
```
- Terminal:** Shows the command-line output of running the script.

```
PS D:\Lehre\Sem 1\Python> cd 'd:\Lehre\Sem 1\Python'; ${env:PYTHONIOENCODING}='UTF-8'; ${env:PYTHONUNBUFFERED}='1'; & 'c:\Users\heuser\Anaconda3\python.exe' 'c:\Users\heuser\.vscode\extensions\ms-python.python-2018.7.1\pythonFiles\PythonTools\visualstudio_py_launcher.py' 'd:\Lehre\Sem 1\Python' '1603' '34806ad9-833a-4524-8cd6-18ca4aa74f14' 'RedirectOutput,RedirectOutput' 'd:\Lehre\Sem 1\Python\06_factorial2.py'
12
13
15
18
22
27
33
40
48
57
```
- Status Bar:** Shows the current file as "Python: Current File (Python)" and the Python version as "Anaconda 5.2.0".

IDE – Entwicklungsumgebung

Visual Studio Code



Explorer,
Suchen,
Quellcode-
verwaltung,
Debuggen,
Erweiterungen

Datei-Explorer

Einstellungen

The screenshot shows the Visual Studio Code interface with the following components visible:

- Explorer:** On the left, a sidebar titled "EXPLORER" lists various Python files in a tree view.
- Code-Editor:** The main central area displays a Python script named "06_factorial2.py". The code implements a factorial function using a while loop.
- Terminal:** At the bottom, the terminal window shows the command line output of running the script.

Code-Editor

Ausgaben,
Debugging-
Konsole,
Terminal,
Probleme.

IDE – Entwicklungsumgebung

Visual Studio Code: Debugging



Haltepunkt



```
06_factorial2.py *  
1 n = 12  
2 count = 0  
3 # while-Schleife: solange Kriterium erfüllt ist, wird Schleife durchlaufen  
4 while count < 10:  
5     # The next is a smart version of print(str(n1) + ' , ')  
6     n = n + count  
7     print(n)  
8     count = count + 1
```

Im Debug-Modus lassen sich mit der Maus vor der Zeilenummerierung Breakpoints (Haltepunkte) setzen. Der Haltepunkt erscheint als roter Punkt vor der Zeile (in unserem Beispiel Zeile 6; s. oben).

Der Debugger wird anschließend gestartet.

Das Skript stoppt in Zeile 6.

An diesem Punkt angelangt, können Variablen mit Hilfe von Debuggen > Überwachen überwacht werden. In unserem Beispiel sollen dies die Variablen count und n sein (s. nächste Folie).

IDE – Entwicklungsumgebung

Visual Studio Code: Debugging



Aktueller Codeverlauf

```
06_factorial2.py x
1 n = 12
2 count = 0
3 # while-Schleife: solange Kriterium erfüllt ist, wird ...
4 while count < 10:
5     # The next is a smart version of print(str(n1) + ...
6     n = n + count
7     print(n)
8     count = count + 1
```

VARIABLEN

- Local
 - name : ' main '
 - __doc__: None
 - __package__: None
 - __loader__: None
 - __spec__: None
 - __file__: 'd:\\Lehre\\Sem...'
 - __cached__: None
 - __builtins__: {'Arithmeti...

ÜBERWACHEN

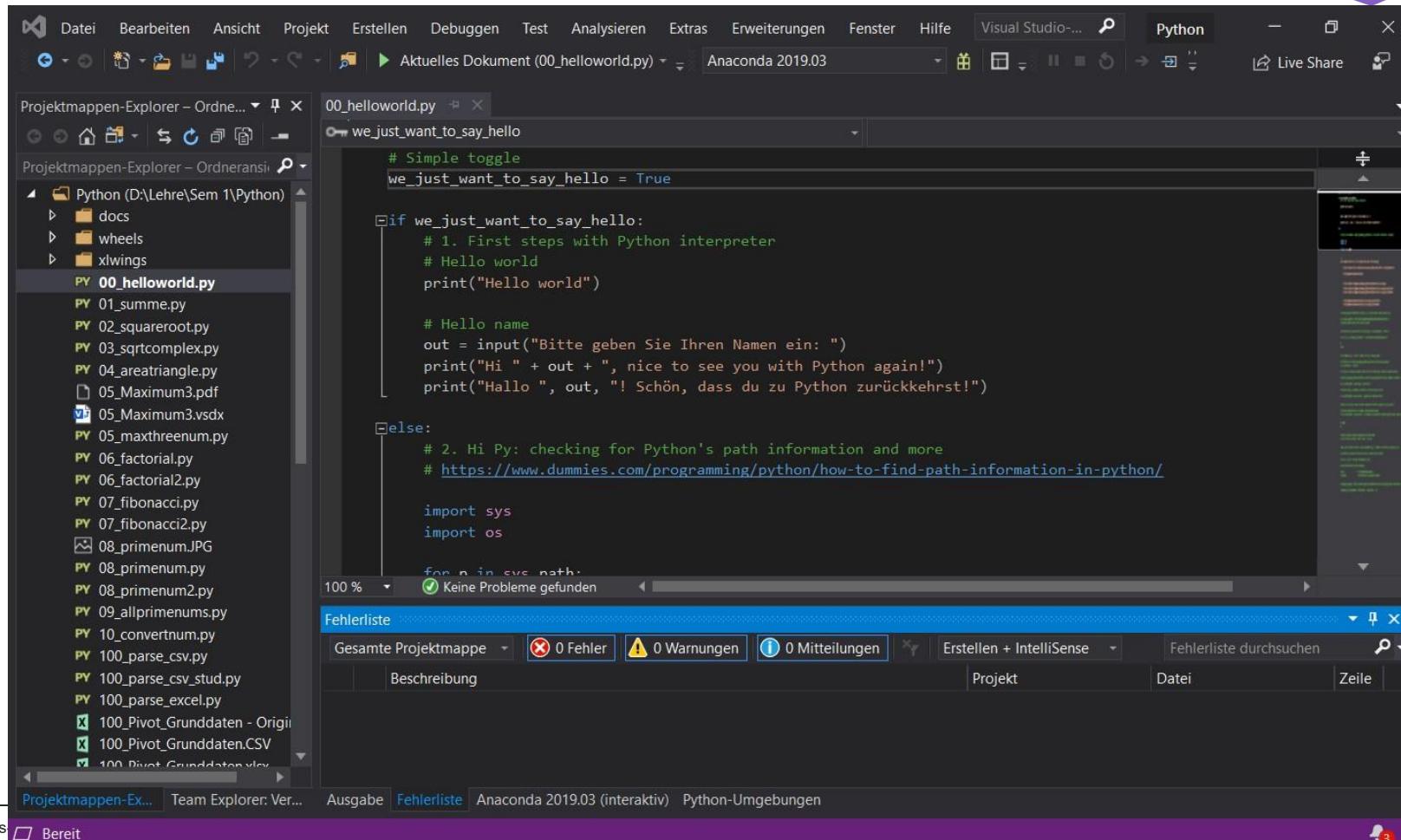
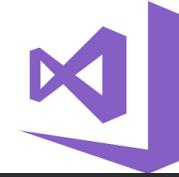
count: 1
n: 12

Überwachte Variablen →

Mit Debuggen > Prozedur-/Einzelschritt lässt sich das Programm sukzessive weiter bewegen. Dabei lassen sich die Inhalte der überwachten Variablen zum aktuellen Skriptverlauf (gelber Pfeil; in unserem Beispiel Codezeile 4) auf der linken Seite überwachen.

IDE – Entwicklungsumgebung

Alternative IDE: Visual Studio Enterprise



The screenshot shows the Visual Studio Enterprise interface. The left sidebar displays a 'Projektmappen-Explorer' with a tree view of files and folders, including '00_helloworld.py' which is selected. The main area is a code editor showing Python code for a 'Hello World' application. The code includes an if-else block that prints 'Hello world' or asks for a name and greets it. Below the code editor is a 'Fehlerliste' (Error List) panel showing '0 Fehler', '0 Warnungen', and '0 Mitteilungen'. At the bottom, there are tabs for 'Projekt', 'Datei', and 'Zeile'.

```
# Simple toggle
we_just_want_to_say_hello = True

if we_just_want_to_say_hello:
    # 1. First steps with Python interpreter
    # Hello world
    print("Hello world")

    # Hello name
    out = input("Bitte geben Sie Ihren Namen ein: ")
    print("Hi " + out + ", nice to see you with Python again!")
    print("Hallo " + out + "! Schön, dass du zu Python zurückkehrst!")

else:
    # 2. Hi Py: checking for Python's path information and more
    # https://www.dummies.com/programming/python/how-to-find-path-information-in-python/

    import sys
    import os

    for p in sys.path:
```

