Ostfalia
Hochschule für angewandte Wissenschaften

Fakultät Maschinenbau

Labor Ingenieurinformatik I
Teil 1 – Arbeiten mit der IDE

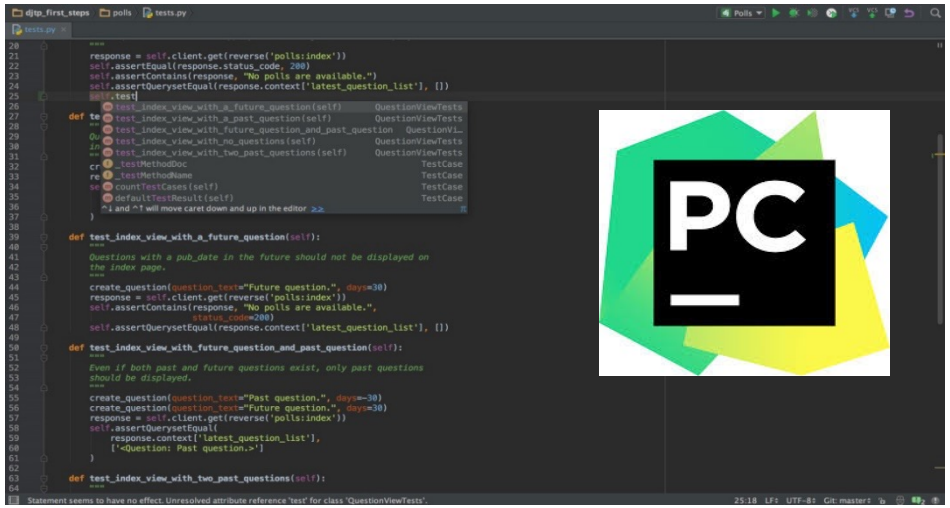
Prof. Dr.-Ing. Udo Triltsch, Prof. Dr.-Ing. M. Strube

Ziele und Organisation der Laborveranstaltung

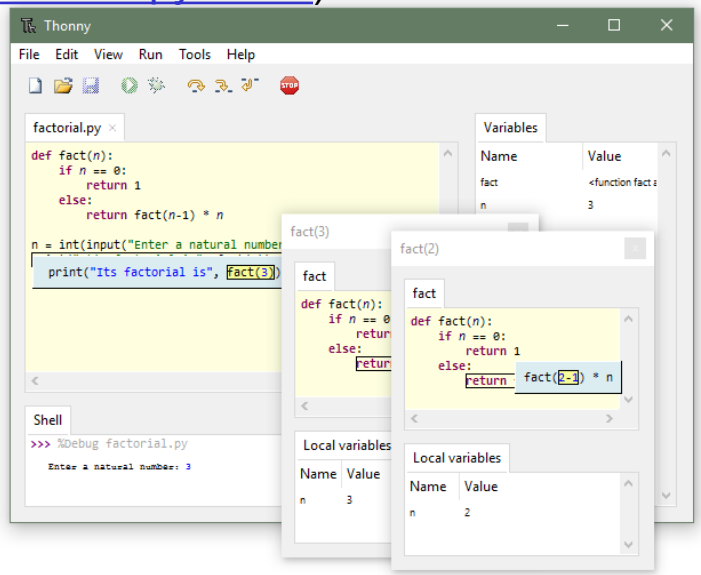
- Die Studierenden lernen ergänzende Inhalte zur Vorlesung Ingenieurinformatik I.
- Die Inhalte und Arbeitstechniken werden im ersten Teil des Semesters in gemeinsamen Laborübungen vorgestellt.
- In der zweiten Semesterhälfte bearbeiten die Studierenden eine Programmieraufgabe in Kleingruppen, um die gelernten Inhalte anzuwenden und zu vertiefen.
- Diese Aufgabe muss bearbeitet werden, um den Leistungsnachweis für diese Veranstaltung zu erbringen.
- Die Note setzt sich aus dem abgegebenen Projekt und einem Kolloquium zusammen. Dabei zählt das Projekt 60 % und das Kolloquium 40 % zu Ihrer finalen Note.

Der Werkzeugkasten des Programmierers - Entwicklungsumgebung / IDE

- Als Entwicklungsumgebung oder IDE (Integrated Development Environment) werden speziell für Programmierer gestaltete Programme bezeichnet, welche die wichtigsten Werkzeuge in einer geschlossenen Umgebung zusammenführen.
- Für Python gibt es z.B. folgende IDE´s:
  - **IDLE** (Integrated Development and Learning Environment): Standard IDE, wird mit dem Python Installer ausgeliefert
  - **Thonny**: Einfache IDE für Einsteiger, die etwas mehr Komfort bietet, als IDLE (https://thonny.org/)
  - **Spyder**: Teil der Anaconda Distribution (https://www.spyder-ide.org)
  - **PyCharm**: Professionelle Python Entwicklungsumgebung (https://www.jetbrains.com/pycharm)



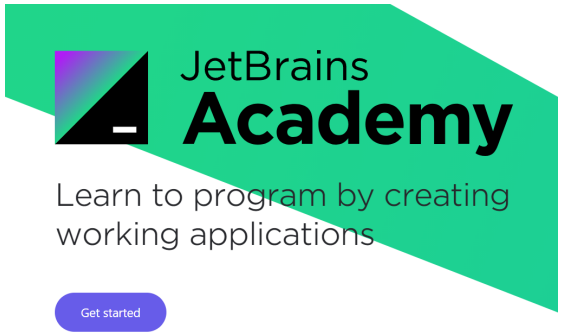
Quelle: www.jetbrains.com/pycharm/

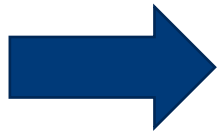


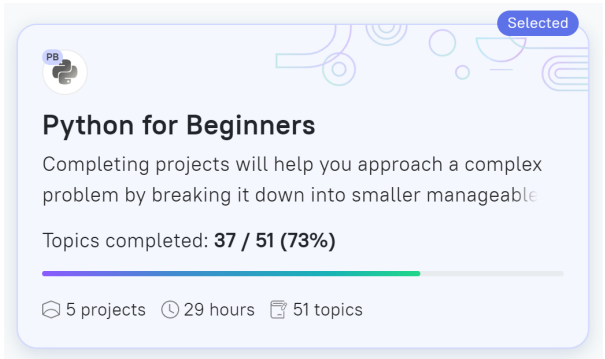
Quelle: www.thonny.org

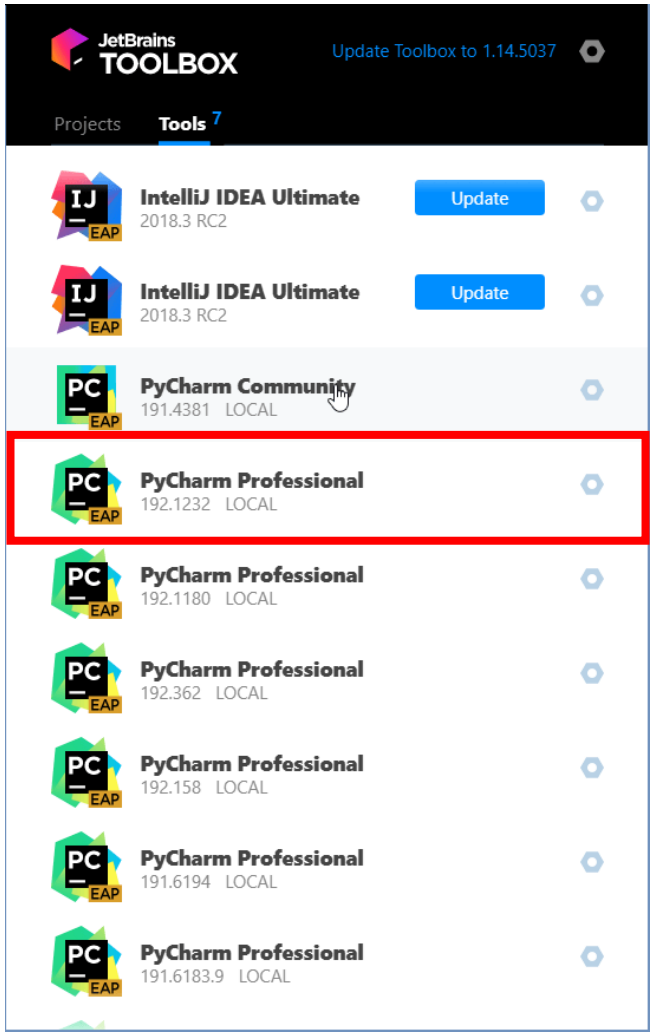
Installation von PyCharm

- Die einfachste Art, die Umgebung lokal zu installieren, ist der Weg über die Toolbox-App.
- Gehen Sie zur Seite von JetBrains: https://www.jetbrains.com/help/pycharm/installation-guide.html#toolbox
- Um die Software zu nutzen, benötigen Sie einen JetBrains Account. Als Studierende der Ostfalia erhalten Sie eine umfassende Lizenz für Produkte der Firma.
- Zusätzlich erhalten Sie Zugang zur JetBrains Academy, einer sehr guten Lernumgebung für das praktische Programmieren in Python.

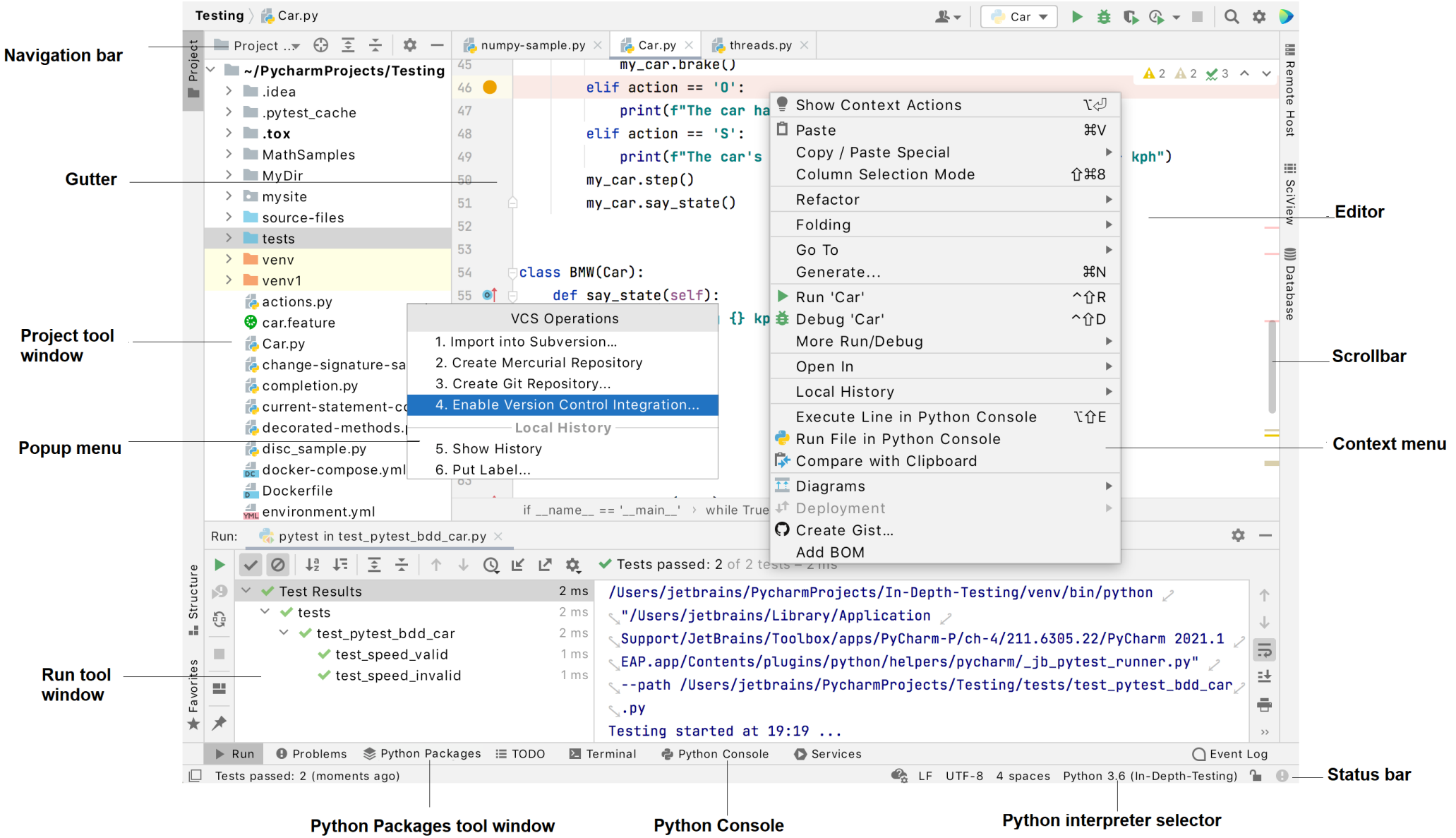






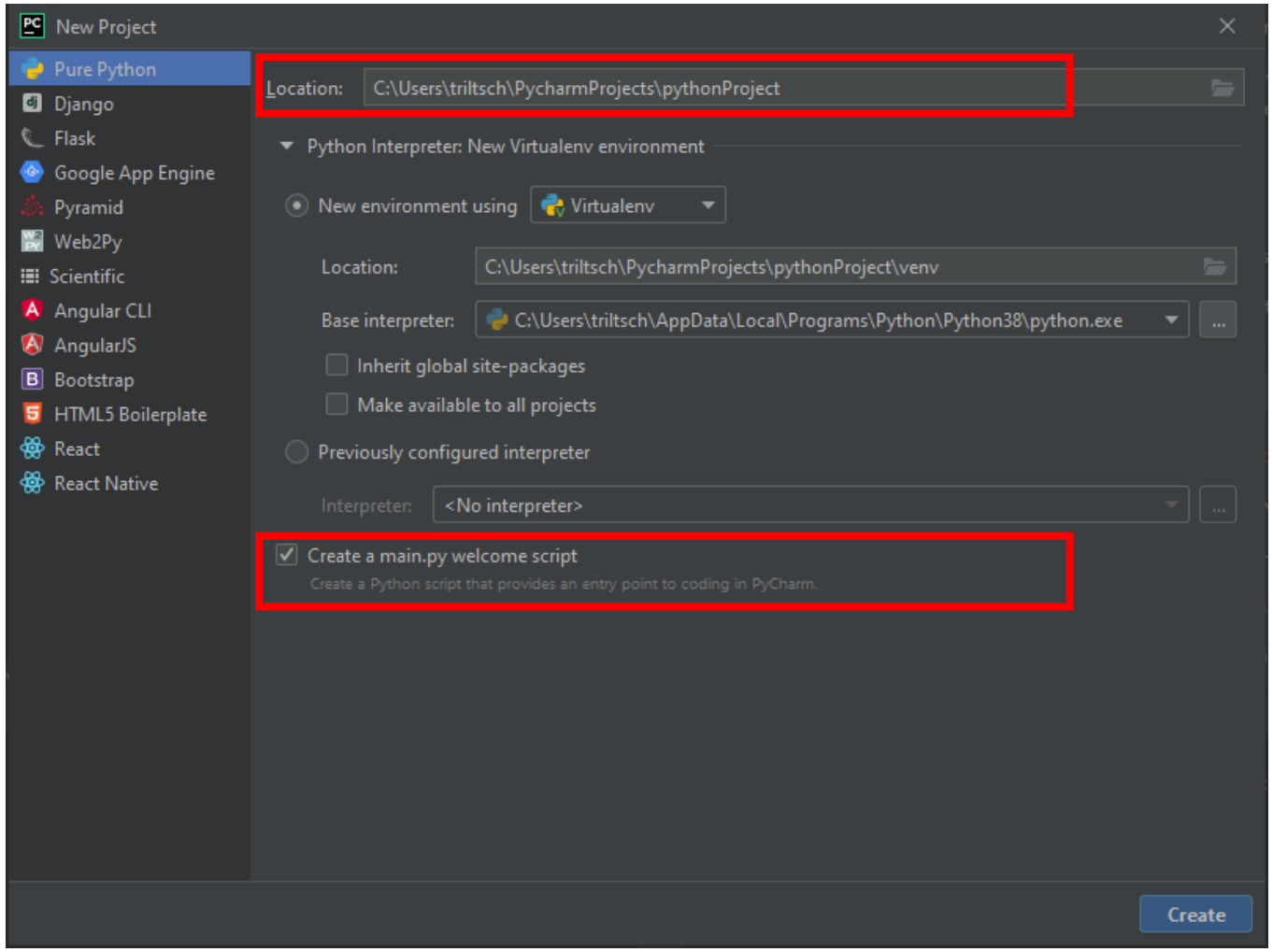


Überblick über die Benutzeroberfläche



Ein Projekt anlegen

- Klicken Sie entweder im Willkommensbildschirm auf „New Project“ oder Wählen Sie im Menü: „File“ → „ New Project“
- Wählen Sie einen Speicherort und einen sinnvollen Namen für Ihr Projekt
- Legen Sie eine neue virtuelle Umgebung an, diese gilt dann nur für dieses Projekt
- Wenn Sie eine Standard-Start-Datei „main.py“ anlegen wollen, lassen Sie den Haken unten gesetzt, ansonsten deaktivieren Sie ihn.
- Drücken Sie „Create“



https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html

Ein erstes Beispiel

Wir versuchen uns an einer sehr einfachen Implementierung des Spiels „Galgenraten“.

Es wird zwei mögliche Ergebnisse geben. Zuerst drucken wir eine Begrüßungsnachricht aus und fordern dann den Spieler auf, das Wort zu erraten, das wir für das Spiel festgelegt haben. Gelingt es dem Spieler, das richtige Wort zu erraten, meldet das Spiel "Du hast überlebt!"; andernfalls wird der Spieler die Nachricht "Du hast verloren!“ sehen.

Zielsetzung:
Frag den Spieler nach einem möglichen Wort. Gib "Du hast überlebt!“ auf dem Bildschirm aus, wenn der Benutzer das Wort erraten hat. Gib "Du hast verloren!“ aus, wenn der Benutzer das Wort falsch erraten hat. Implementieren Sie Hinweise zum Lösungswort, falls der Spieler falsch geraten hat.

Beispiel 1

G A L G E N R A T E N

Rate das Wort: python

Du hast überlebt!

Beispiel 2

G A L G E N R A T E N

Rate das Wort: java

Du hast verloren!

Tipp zur Fallunterscheidung:

```
if (wort == "python"):
    #Code bei Übereinstimmung
else:
    #Code im anderen Fall
```

Arbeiten mit dem Debugger

- Ein Debugger ist ein spezielles Programm bzw. Tool zum Auffinden von Fehlern.
- Debugger gibt es für nahezu alle Programmiersprachen – auch für Python.
- Die Programmausführung kann z.B. an einer definierten Stelle unterbrochen werden, indem man sogenannte **Haltepunkte** (Breakpoints) im Programm gesetzt werden.
- An den Haltepunkten können z.B. Werte von Variablen beobachtet und geändert werden oder der Quellcode kann Zeile für Zeile ausgeführt werden, um das Verhalten zu beobachten.
- Wichtigste Möglichkeiten für die Fehlersuche in Programmen:
  - schrittweises Verfolgen (sogenanntes **Steppen**),
  - gezieltes Anhalten (mit **Haltepunkten** beziehungsweise **Breakpoints**) oder
  - Auswertung einzelner Variablen (**Watch**)
  - gezielte Veränderung von Variablen während der Laufzeit

Quelle: www. keluro.com

Zweites Beispiel – Programmabläufe verstehen mit dem Debugger

- Sie finden in Stud.IP eine main.py Datei mit einem unkommentierten Programm.
- Legen Sie ein neues Projekt an und importieren Sie diese Datei.
- Finden Sie durch die Benutzung des Debuggers heraus, wie das Programm abläuft und was dieses macht!
- Ergänzen Sie Kommentare!
- Wenn Sie herausgefunden haben, was der Algorithmus macht, benennen Sie die Datei um.
  Benutzen Sie dazu das Refactoring-Tool von PyCharm.
- Tipp: geben Sie systematisch Zahlen von 1-25 ein. Welche Ergebnisse beobachten Sie?
- Geben Sie die Null ein, was passiert? Wo liegt der Fehler? Können Sie Ihn verhindern?

```python
# Dieses erste Beispielprojekt soll die grundlegende Arbeit
# mit PyCharm und Debugger demonstrieren.



# Autor: U. Triltsch
# Erstellt am: 04.10.021
# Letzte Aenderung am: 05.10.2021

# Ihre Beschreibung kommt hier hin...
def algorithm(number):

    g = number/2.0;
    g2 = g + 1;
    while(g != g2):
        n = number/ g;
        g2 = g;
        g = (g + n)/2;

    return g;

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    #HIPO- Principle
    #Input
    number = float(input("Please input a number:\n"))
    #Process  --> What is happening here?
    result = algorithm(number)
    #Output
    print("The result of the operation on: ", number, "is: ", result)
```