



Bild: Andreas Martini

Codefabriken

Die Python-Entwicklungsumgebung für Sie

Entwicklungsumgebungen stellen die richtigen Werkzeuge bereit, um schnell und effizient zu programmieren. Sie sollten mehrere davon kennen, denn sie sind auf verschiedene Bedürfnisse zugeschnitten und Python ist unglaublich vielseitig.

Von Pina Merkert, Wilhelm Drehling und Jan Mahn

Um Python zu programmieren, reicht ein einfacher Texteditor, wie ihn jedes Betriebssystem an Bord hat. Aber schon nach wenigen Zeilen wünscht man sich mehr Komfort. So unterschiedlich wie die Einsatzbereiche für Python sind auch die Entwicklungsumgebungen. Wir stellen sechs Programme vor, die die Arbeit mit Python-Code immens erleichtern. Sie fragen sich, wie so ein aufgebohrter Texteditor beim Programmieren hilft? Hier ein Beispiel aus dem Profi-Alltag:

Am Anfang klang `BackendController` wie ein guter Name für die Klasse. 14 Commits später produziert sie aber Objekte, die ihrerseits die Anwendungslogik umsetzen. Beim Programmieren ist diese Ver-

wandlung mit kleinen Schritten einfach so passiert und im Kontext des gesamten Programms auch sinnvoll. Nun wäre der Name `BackendControllerFactory` aber sinnvoller. Mit einem simplen Texteditor wäre es aber eine Menge Arbeit, den Klassennamen in allen Dateien zu ändern, wo sie vorkommt.

Solch ein Refactoring geht mit einer mächtigen Entwicklungsumgebung (IDE) mit zwei Mausklicks. Eine so praktische Automatik ist keine Kleinigkeit, weil die IDE den Code wie ein Compiler lesen (parsen) und anhand der Syntax entscheiden muss, wo die Klasse überall verwendet wird. Wer mehr als nur ein paar Zeilen programmiert, weiß den Aufwand aber zu schätzen: Für die Verständlichkeit des

Codes spielt es eine entscheidende Rolle, wie Variablen, Funktionen und Klassen benannt sind. Das ist nicht nur für andere Entwickler sinnvoll – man muss den eigenen Code ja auch immer wieder lesen und verstehen.

Das Beispiel zeigt einen der sehr unterschiedlichen Fälle, in denen eine IDE ihre Stärken ausspielen kann. Data-Scientists werden wahrscheinlich auch ohne Umbenenn-Automatik auskommen, wollen dafür aber mit ein paar Tastendrücken Diagramme zeichnen. Raspi-Bastler brauchen dagegen vor allem einen Startknopf, um Programme ganz schnell testen zu können. So unterschiedliche Bedürfnisse erfüllt nicht ein Programm allein. Und es ist völlig legitim, Vorlieben für bestimmte IDEs zu haben. Um dieser Vielseitigkeit gerecht zu werden, stellen wir im Folgenden mehrere Entwicklungsumgebungen vor: von einfach bis vollgepackt, von schnell installiert bis browserbasiert mit lokalem Webserver.

Interaktive Konsole

Wenn Sie nur mal schnell eine Zeile tippen wollen, ist es unnötig aufwendig, erst eine Datei zu erstellen, die Sie danach ohnehin wieder löschen müssen.

Wenn Sie in einer Konsole nur `python` ohne weitere Parameter eintippen, startet eine interaktive Konsole. Die führt jede mit Enter bestätigte Zeile direkt aus. Da der Interpreter den Code liest, auswertet, ausgibt und das Ganze beliebig wiederholt, wird die interaktive Konsole auch „REPL“ genannt (read, evaluate, print, loop). Python eignet sich damit als Taschenrechner, weil die Ergebnisse jeder Berechnung in der Konsole sofort erscheinen. Die interaktive Konsole bewährt sich auch beim separaten Testen kleiner Codeschnipsel. Standardmäßig ist sie aber schmucklos und ohne Code-Vervollständigung. Letztere würde schon während des Tippens Angebote machen, wie die Code-Zeile sinnvollerweise weitergeht, was Tipparbeit spart und Fehler vermeidet.

Der Befehl `pip install ipython` installiert die IPython-Konsole, mit der interaktives Python-Tippen viel angenehmer ist. Starten Sie die Konsole einfach mit `ipython` ohne Parameter und profitieren Sie von buntem Code, der sich mit Tab vervollständigen lässt.

Jupyter Notebooks

Nicht jeder will in Python umfangreiche Programme schreiben – manchmal (zum

Beispiel in Wissenschaft und Lehre) will man auch visualisieren und am Code dokumentieren, wie man zu einem Ergebnis gekommen ist. Dafür sind Jupyter-Notebooks optimal.

Wenn Sie die Software mit `pip install jupyter` installiert und mit `jupyter notebook` gestartet haben, öffnet sich ein Browserfenster, das die Web-App des von Jupyter gestarteten lokalen Webserver auf Port 8888 anzeigt. Jupyter nutzt aus Sicherheitsgründen noch ein Token in der URL, die der Befehl öffnet. Sollte das Browserfenster nicht wie erwartet starten, können Sie die URL mit Token in der Konsole kopieren und in die Adressleiste einfügen. In der Übersichtsseite wählen Sie danach oben rechts „New/Python 3“ und erstellen ein neues Notebook.

Mit „Notebook“ bezeichnet Jupyter ein Dokument bestehend aus formatiertem Text – auch mit Bildern, wenn Sie möchten – und Code-Schnipseln. Jede grau hinterlegte Code-Zelle können Sie mit Umschalt+Enter ausführen, Konsolenausgaben und Rückgabewerte landen dann darunter. Ein solches Dokument ist nicht nur ideal, um Code zu erklären, es erlaubt auch schnelle Experimente ähnlich wie auf der interaktiven Konsole.

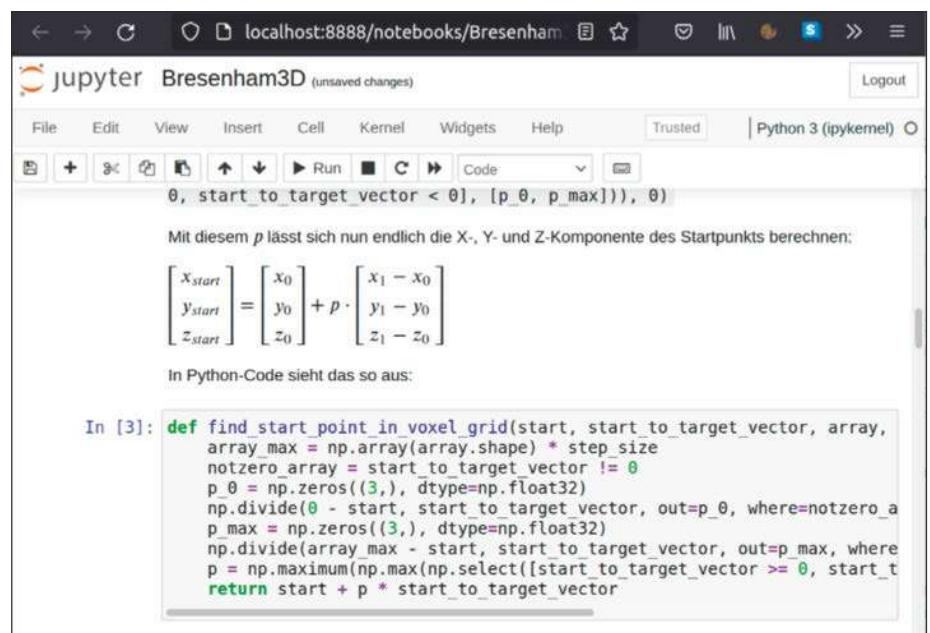
Der größte Vorteil besteht aber in der Fähigkeit des Browsers, neben Text auch dynamisch generierte Grafiken anzeigen zu können. Ein Beispiel: Mit der magischen Zeile `%matplotlib` schalten Sie Inline-Diagramme mit der gleichnamigen

c't kompakt

- Texteditoren mit Syntaxhervorhebung erleichtern den schnellen Einstieg in Python.
- Einzelne Zeilen Python kann man sehr schnell auf der interaktiven Konsole und noch schöner in Jupyter-Notebooks ausprobieren. Letztere zeichnen auch ruckzuck Diagramme und Bilder.
- Für große Projekte gibt es mächtige Entwicklungsumgebungen wie VSCode und PyCharm, die Code verstehen, vervollständigen, testen und auf häufige Fehler hinweisen.

Bibliothek frei. Gibt die letzte Zeile einer Code-Zelle ein Diagramm-Objekt zurück, zeichnet es der Browser direkt unter die Zelle. Das kann auch ein interaktives Diagramm wie der 3D-Plot in unserem Artikel über den Bresenham-Algorithmus sein [1]. Das Einbetten von Diagrammen geht viel schneller als in einer Tabellenkalkulation.

Dank der übersichtlichen Code-Struktur, meist mit Erklärungstext zwischen den Zellen und der sofort sichtbaren Diagramme, sind Jupyter-Notebooks das Lieblingswerkzeug vieler Statistiker und fast aller KI-Forscher. Google bietet das



Jupyter-Notebooks kombinieren Text mit im Browser ausführbarem Code. Diagramme lassen sich direkt ins Dokument zeichnen.

Konzept unter dem Namen „Colab“ als Clouddienst an, falls die Berechnungen lieber gleich auf einem gemieteten Server laufen sollen.

Texteditor mit Extras

Für kleine Skripte und die meisten Raspibasteleien reicht fürs Programmieren ein Texteditor. Der sollte Schlüsselwörter, Zahlen und Strings hervorheben, damit die Augen Strukturen im Code leichter wiederfinden können. So ein Syntax-Highlighting (auch Syntaxhervorhebung genannt) ist die Mindestanforderung für jeden Code-Editor. Manche vorinstallierte Editoren wie Gedit unter Gnome können das einfach so. Unrühmliche Ausnahmen sind der Standard-Editor von macOS und Notepad unter Windows (Notepad++ behebt den Makel).

Ein Editor speichert nur die Python-Datei mit der Endung .py. Zum Ausführen öffnet man einfach ein Konsolenfenster und tippt `python dateiname.py`. Python nutzt intern zwar ähnlich wie Java einen Bytecode; in der Praxis bekommt man die Datei mit diesem Code aber nicht zu Gesicht. Python wirkt wie eine interpretierte Sprache, der Compileraufruf entfällt.

Eine nahe liegende Ergänzung für einen Code-Editor ist ein Startknopf. Dann muss man nicht ins Konsolenfenster wechseln, wenn man das Skript testen will. Spyder ist ein auf dem Raspi vorinstallierter Editor, der so einen Knopf eingebaut hat. Nachinstallieren kann man ihn auf allen Systemen.

Es ist verlockend, mit einem extrem einfachen Editor loszulegen. Wer allerdings absehen kann, dass die Programmierprojekte einige Hundert Codezeilen lang werden, sollte von Anfang an auf eine mächtigere IDE setzen, um sich nicht zweimal in eine neue Programmoberfläche einarbeiten zu müssen.

Sublime

Sublime ist ein Texteditor, der viele Programmiersprachen unterstützt – darunter auch Python. Praktischerweise bringt der Editor Syntax-Hervorhebungen für Python und einen Ausführen-Knopf in Form eines „Build“-Befehls gleich mit. Der Editor ist zwar kostenpflichtig, aber die Test-Version läuft zeitlich unbegrenzt. Unter macOS moniert Sublime manchmal fehlende Pakete von Xcode, bevor er den Code ausführt. Auf allen anderen Betriebssystemen geht es nach der Installation direkt los.

Sublime funktioniert bereits mit den Werkseinstellungen hervorragend, aber mit ein paar zusätzlichen Paketen schneiden Sie den Editor auf sich zu. Die Pakete sind Open Source und lassen sich direkt über das integrierte „Package Control“ installieren. Klicken Sie dafür in der Menüleiste auf „Tools“ und anschließend auf „Command Palette“. Tippen Sie „Install Package Control“ ein und bestätigen Sie mit Enter.

Mit dem „Build“-Befehl führt Sublime Python-Skripte direkt aus und schreibt `print()`-Ausgaben in ein integriertes Fens-

ter. In dem nimmt er aber keine Eingaben für `input()` entgegen. Abhilfe schafft das Plug-in „SublimeREPL“. Es ergänzt einen zusätzlichen „Build“-Befehl, der ein Fenster öffnet, das auch Eingaben annimmt. Zusätzlich können Sie mit dem Plug-in in Sublime auch eine interaktive Python-Konsole öffnen.

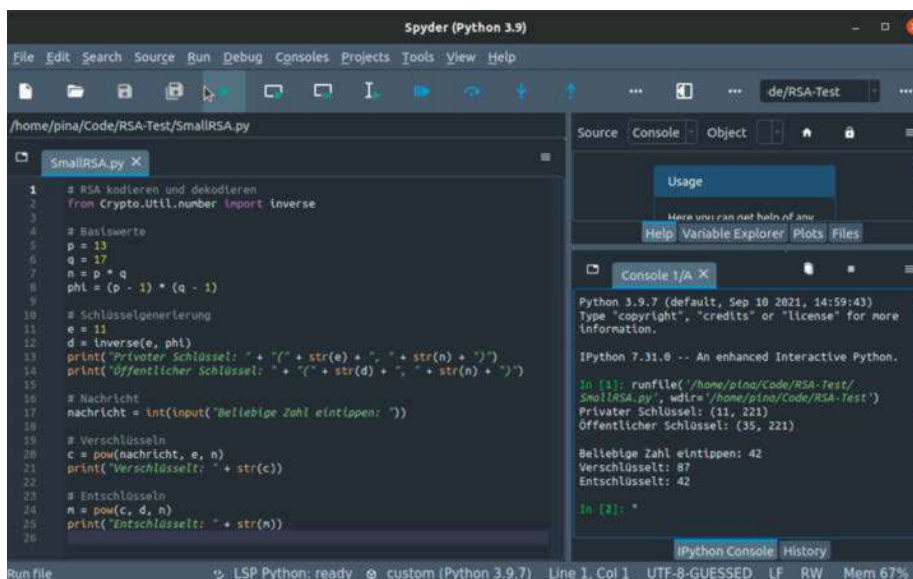
Das mächtigste Plug-in fehlt aber noch: Anaconda. Das Plug-in zur Python-Distribution ergänzt Sublime mit Funktionen einer vollwertigen IDE (Integrated Development Environment). Das Anaconda-Plug-in zeigt Fehler im Code schon beim Tippen an und bringt unter anderem eine Autovervollständigung mit.

Erwähnenswert ist noch das Plug-in „BracketHighlighter“. Vor allem bei langen Code-Dateien mit vielen Klammern hilft das Werkzeug bei der Übersicht, da es alle offenen Klammern an der linken Liste farblich markiert. Sublime hat trotz vieler Plug-ins nicht alle Funktionen einer IDE, weshalb sich der Editor vor allem für Programmierer eignet, die ein übersichtliches Interface und Geschwindigkeit schätzen.

Visual Studio Code

Visual Studio Code, abgekürzt VSCode oder VSC, ist innerhalb von wenigen Jahren zum populären Universalwerkzeug für unterschiedlichste Entwickler in vielen Sprachen und Betriebssystemen (Windows, Linux und macOS) geworden. Es stammt wie sein Namensverwandter „Visual Studio“ aus dem Hause Microsoft, hat aber wenig mit diesem fast namensgleichen Schwergewicht aus der Windows-Welt gemein. VSC wird als Open-Source-Software entwickelt und ist vor allem deshalb so beliebt, weil es modular konstruiert ist. Zur Python-IDE wird die Software, indem man eine Python-Erweiterung installiert. Den Marketplace für Erweiterungen findet man links in der Menüleiste hinter dem Symbol mit den vier Bauklötzen.

Die populärste (aber nicht die einzige) Python-Erweiterung heißt schlicht „Python“ und kommt ebenfalls vom Microsoft. Auf deren Autovervollständigung (über die Visual-Studio-Code-Funktion IntelliSense) möchte man nach wenigen Zeilen nicht mehr verzichten. Beim Tippen bekommt man Funktions- und Variablenamen vorgeschlagen und zu Funktionen auch direkt ein Pop-up mit einer Erklärung der Parameter. Auch nützlich: In der unteren Leiste kann man schnell



Spyder ist auf dem Raspi vorinstalliert und bringt neben Code-Highlighting auch einen Play-Knopf.

zwischen mehreren auf dem System installierten Python-Versionen umschalten. Und wer gern mit Jupyter-Notebooks arbeitet, kann die dank dieser Erweiterung direkt in der IDE ausführen – der Wechsel in den Browser fällt weg. Interessant für Fortgeschrittene: Eingebaut sind auch ein Python-Debugger, ein Linter (ein Werkzeug, das stilistische und syntaktische Fehler findet), Hilfsmittel fürs Refactoring sowie Steuerelemente für Unit-Tests.

Visual Studio Code ist damit einerseits für Einsteiger eine gute und kostenlose Alternative zum schnöden Texteditor, gleichzeitig aber auch ein Werkzeug, für das viele Profis kostenpflichtige IDEs liegenlassen. Insbesondere, wenn man neben Python noch andere Programmier- oder Auszeichnungssprachen editiert, konfiguriert man sich mit VSC und passenden Erweiterungen eine Arbeitsumgebung für alle Programmierer-Lebenslagen. Ein Highlight für Nutzer von GitHub: Betätigt man im Browser in einem Repository auf GitHub die Punkt-Taste auf der Tastatur, öffnet man das Repo in einer vollwertigen VSC-Instanz im Browser. Das klappt sogar unterwegs auf dem Tablet.

PyCharm

Wer große Python-Projekte entwickelt und bei der IDE keine Kompromisse machen will, sollte sich PyCharm ansehen. Die kommerzielle Entwicklungsumgebung von JetBrains parst den Code und weist mit sogenannten „Inspections“ schon beim Tippen auf Fehler, Probleme und unsauberen Stil hin. Außerdem bietet sie extrem mächtige Werkzeuge fürs Refactoring bis hin zum Ändern von Variablennamen innerhalb von Strings. Interaktive Konsole, Test-Runner (siehe [3]) und Virtualenvs sind nahtlos in die Programmoberfläche integriert. Im Debugger-Modus setzt man mit einem Klick Unterbrechungspunkte neben die Code-Zeile und PyCharm zeigt in einem Fenster, mit welchen Werten Variablen belegt sind.

PyCharm gibt es kostenlos in einer Community Edition ohne Supportanspruch. Gegenüber der als Abomodell vertriebenen Professional Edition fehlen einige nicht überlebenswichtige Features. So versteht nur die kostenpflichtige Version automatisch die Importpfade eines Django-Projekts und bei clientseitigem JavaScript-Code stellt sich die kostenlose Version etwas dümmel an. Beide Versionen öffnen die gleichen Projektdateien, sodass ein Upgrade jederzeit möglich ist, falls man die Profi-Features braucht.

Die IDE ist in Java programmiert und braucht selbst auf schnellen Rechnern einige Sekunden zum Starten. Sie benötigt so viel RAM, dass Notebooks mit 8 Gigabyte manchmal auf die langsame Festplatte ausweichen (swappen) müssen, um genug Arbeitsspeicher freizuschaffen. Nach dem langsamen Start reagiert die IDE aber prompt auf Eingaben.

Um alle Features von PyCharm aufzuzählen, fehlt hier der Platz (die stehen alle auf der Webseite; zu finden über ct.de/yzmj). Hier nur ein paar Highlights: Strg+Klick auf einen Funktionsnamen führt zur Definition der Funktion. Das klappt genauso auch mit Bibliotheken. Ein Rechtsklick auf eine Variable und im aufploppenden Menü die Auswahl „Find Usages“ öffnet eine Übersicht mit allen Stellen, an denen die Variable benutzt wird. Die Übersicht ist sogar nach Typ sortiert. Mit Funktionen und Klassen funktioniert das auch. Der Menüpunkt „Code/Reformat Code“ formatiert eine ganze Datei nach den in den Projekteinstellungen festgelegten Vorlieben. Selektiert man Codeabschnitte, kann man auch nur den selektierten Teil automatisch forma-

Fachwissen für die neue Arbeitswelt



244 Seiten · 24,90 €
ISBN 978-3-86490-863-7



296 Seiten · 34,90 €
ISBN 978-3-86490-877-4



212 Seiten · 26,90 €
ISBN 978-3-86490-839-2



300 Seiten · 34,90 €
ISBN 978-3-86490-877-4

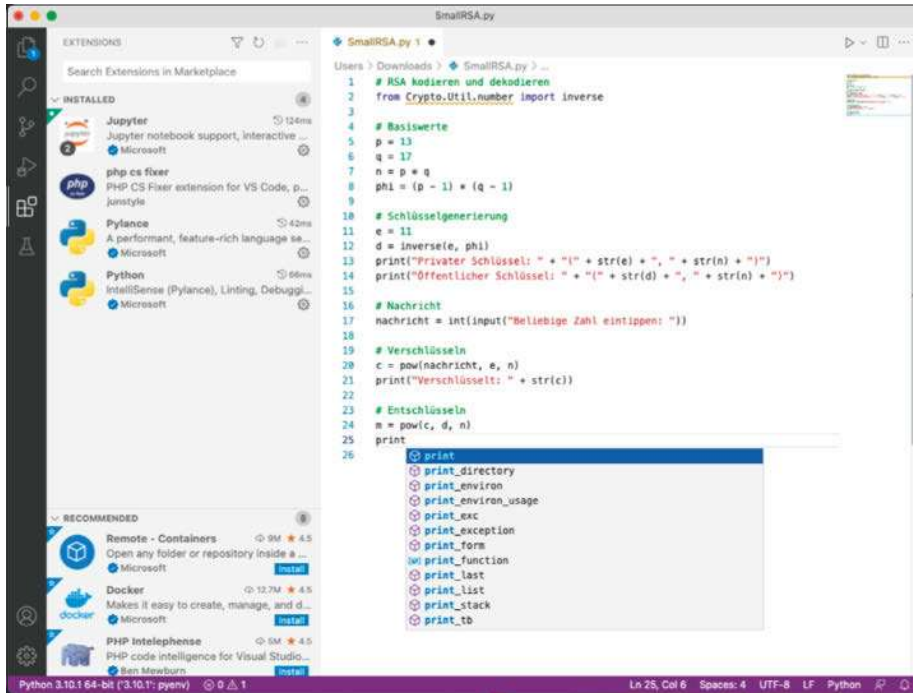


356 Seiten · 36,90 €
ISBN 978-3-86490-798-2



2. Auflage · 176 Seiten · 26,90 €
ISBN 978-3-86490-883-5

Bundle up!
Print & E-Book nur auf
www.dpunkt.de



Die Stärke von Visual Studio Code liegt in den Erweiterungen. Mit den richtigen wird die Universal-IDE zur gut ausgestatteten Python-Entwicklungsumgebung.

tieren lassen. Schön umgesetzt ist auch die Variablen-Liste in der rechten Hälfte vom Debugger-Fenster: Sie zeigt die Datentypen mit kleinen Symbolen an. Da Python nicht statisch typisiert ist, kann man hier schnell erkennen, wenn Variablen falsch belegt sind. Ähnlich übersichtlich sind die grünen Häkchen beim Ausführen von Unittests, mit denen man fehlschlagende


Tests viel schneller erkennt als in der Konsolenausgabe des Test-Runners.

Fazit

Entwicklungsumgebungen lassen sich nicht gut nach objektiven Kriterien testen. Die Vorlieben verschiedener Entwickler sind zu unterschiedlich und oft ist ein einzelnes Feature nicht wichtig genug, um

dafür einen ganz neuen Workflow zu lernen. Dennoch lohnt es sich, immer mal wieder einen Blick über den Tellerrand zu wagen.

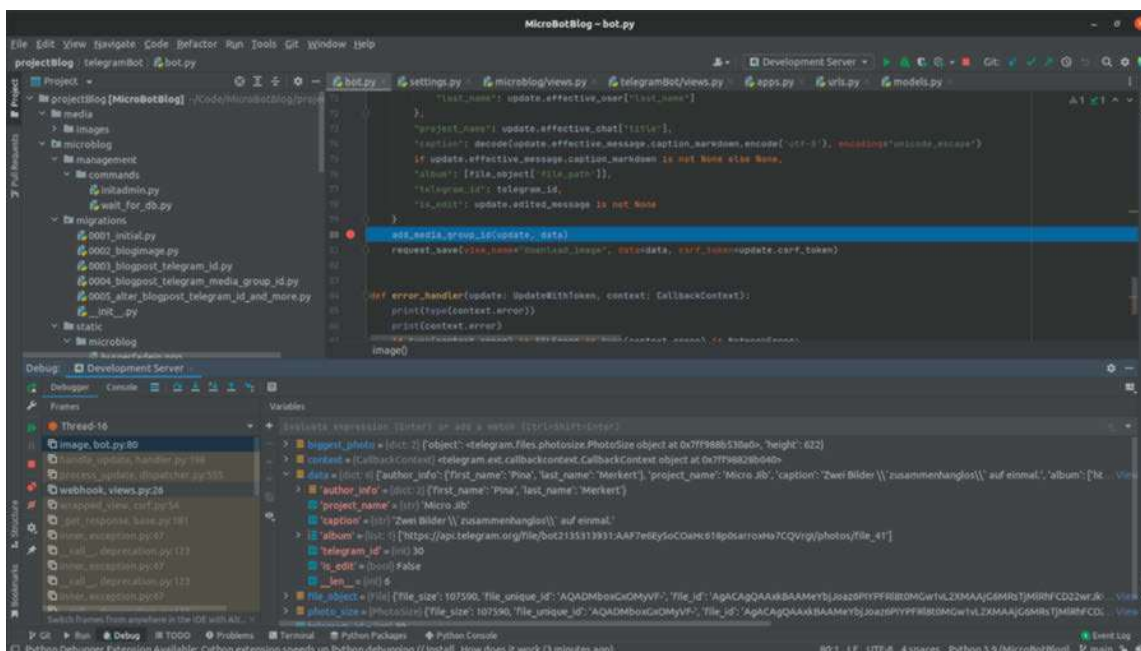
IDEs sind Programme von Entwicklern, die sehr genau verstanden haben, welche Werkzeuge einem Entwickler die Arbeit erleichtern. Sie nehmen einem die repetitiven Aufgaben ab und sie warnen auch erstaunlich frühzeitig vor Fehlern. Das kann das Programmieren enorm beschleunigen und viele Stunden beim Debugging einsparen.

Wunder darf man sich von einer IDE aber nicht erwarten: Keine IDE wird einem jemals abnehmen, eine gute Idee für einen Algorithmus zu erfinden. Und eine IDE sollte einem auch nicht vorschreiben, wie man den Code zu strukturieren hat. Programmieren ist eine mental anspruchsvolle Aufgabe für kreative Menschen. Die IDE ist nur das Werkzeug, damit diese herausfordernde Aufgabe möglichst viel Spaß macht und man – so schnell es geht – zu einem funktionierenden Programm kommt. (pmk@ct.de) 

Literatur

- [1] Pina Mertk, Bresenham 3D, Algorithmus für Linien in Voxelgittern, c't 26/2021, S. 144
- [2] Pina Mertk, Djangos Fundament, Einstieg in serverseitige Webprogrammierung mit Python und Django, c't 1/2022, S. 140
- [3] Pina Mertk, Programmierte Prüfer, Eine Einführung ins automatische Testen mit Python, c't 1/2020, S. 152

Download-Links: ct.de/yzmj



PyCharm zeigt hier das Django-Projekt aus [2]. Links oben listet es die zum Projekt gehörenden Dateien und Ordner auf, rechts daneben sind in mehreren Reitern Code-Dateien geöffnet. Die untere Bildschirmhälfte nimmt das Debugger-Fenster ein, das rechts den Inhalt der lokalen Variablen zeigt.