

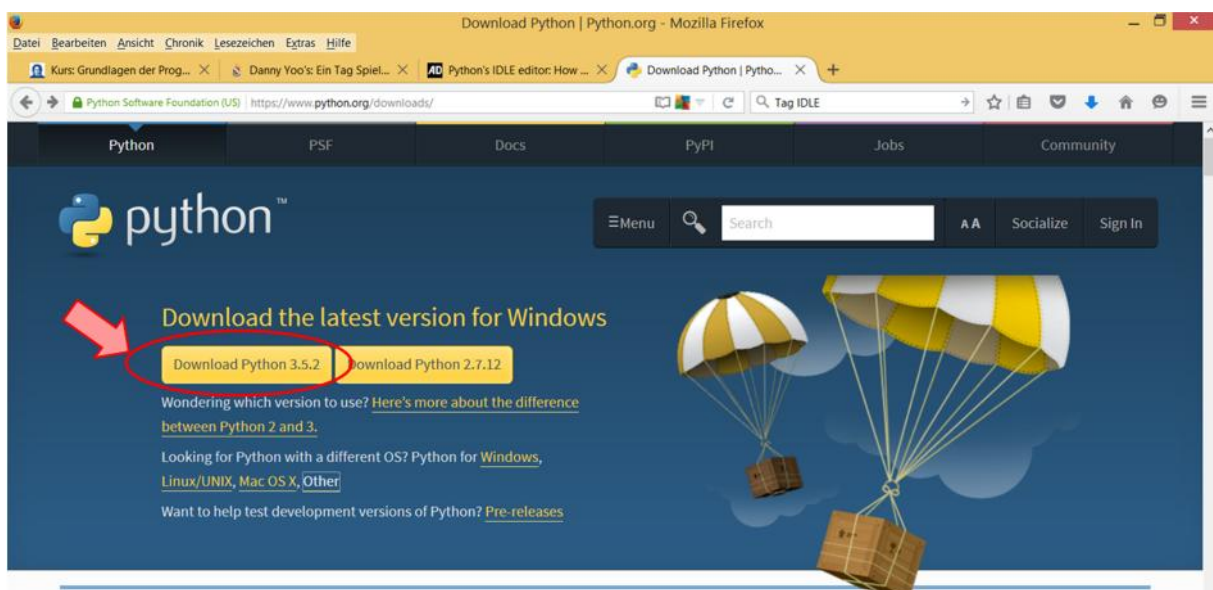
Danny Yoo's: Ein Tag Spielerei mit IDLE

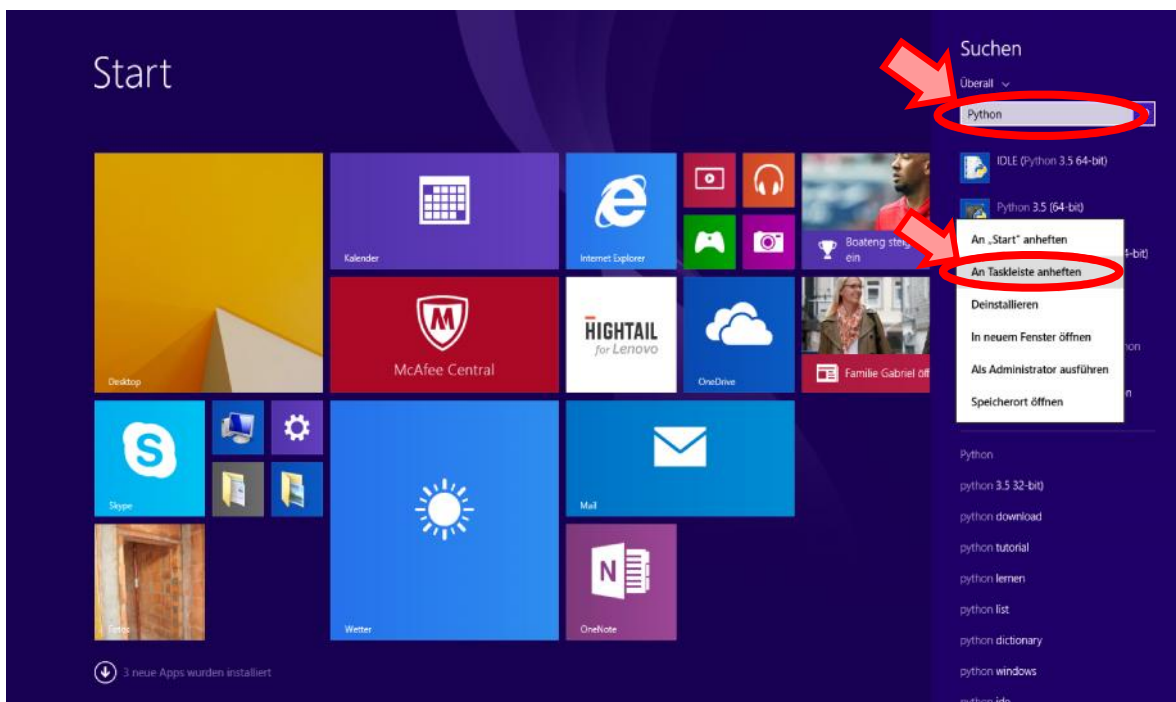
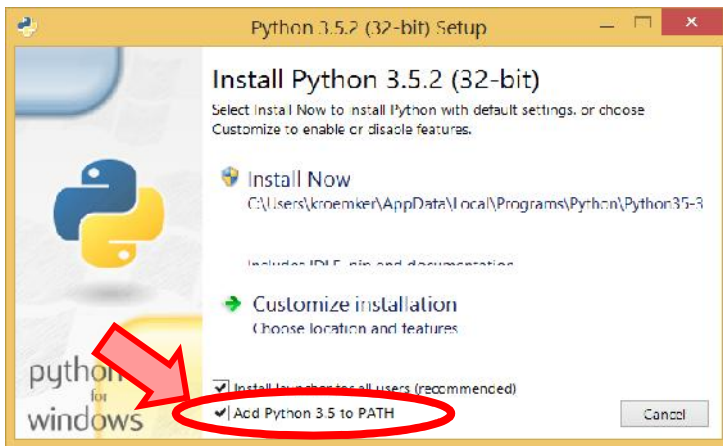
(German [translation](#) by Gregor Lingl) (*Detlef Krömker hat diese Vorlagen zur Nutzung an der Goethe-Uni in PRG1 überarbeitet, insbesondere bezüglich Python 3.x und Windows 8. – Die Sprache ist manchmal sehr, sehr einfach und flapsig, sorry dafür. Wichtig ist aber, dass es manchen hilft. DKs Ergänzungen sind kursiv geschrieben.*) Hier findet man das Original „One Day of IDLE Toying“: https://hkn.eecs.berkeley.edu/~dyoo/python/idle_intro/index.html. Die erste Version ist immerhin schon 15 Jahre alt!

Diese Seite soll neuen Benutzern von Python helfen, die sich vielleicht ein bisschen desorientiert fühlen. Ein Frage, die einem da einfallen mag, ist: „ok, wir haben Python installiert... ummm... was nun?“

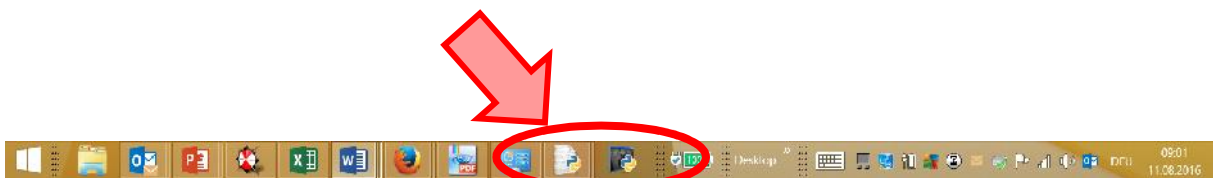
Vielleicht ist es nett einen "visuellen" Führer zu haben, um die anfänglichen Ängste zu reduzieren. Dafür ist diese Seite gedacht. Der Plan ist, eine kleine Session mit IDLE durchzumachen: mit dem **Integrated Development Environment**, der integrierten Entwicklungsumgebung. IDLE wurde entworfen, um einen einfachen Weg zur Verfügung zu haben, die Sprache zu erkunden. Während dieser Session werde ich ein paar täppische Fehler machen, bloß um zu zeigen, was man zu erwarten hat, wenn die Dinge nicht ganz glatt gehen.

(DK: Ganz kurz zur **Installation**: Python steht für sehr viele Plattformen zur Verfügung, Sie laden die Python-Programmierungsumgebung und zwar Version 3.5.2 am besten direkt von der Python Software Foundation: <https://www.python.org/downloads/>. Jetzt ein paar Hinweise, die allerdings nur für Windows 8 gelten. Klicken Sie einfach auf **Download Python 3.5.2**. Dies startet einen Download einer „Installations.exe“, die Sie bitte ausführen. Die Defaults sind durchaus vernünftig: Stören Sie sich bitte nicht daran, dass die „Automatik“ ggf. die 32 Bit Version installiert, obwohl Sie einen 64-Bit-Rechner haben (Es funktioniert alles!). Vergessen Sie bitte NICHT die Auswahlbox „Add Python 3.5 to PATH“ anzuhaken, dass erspart später viel Arbeit.



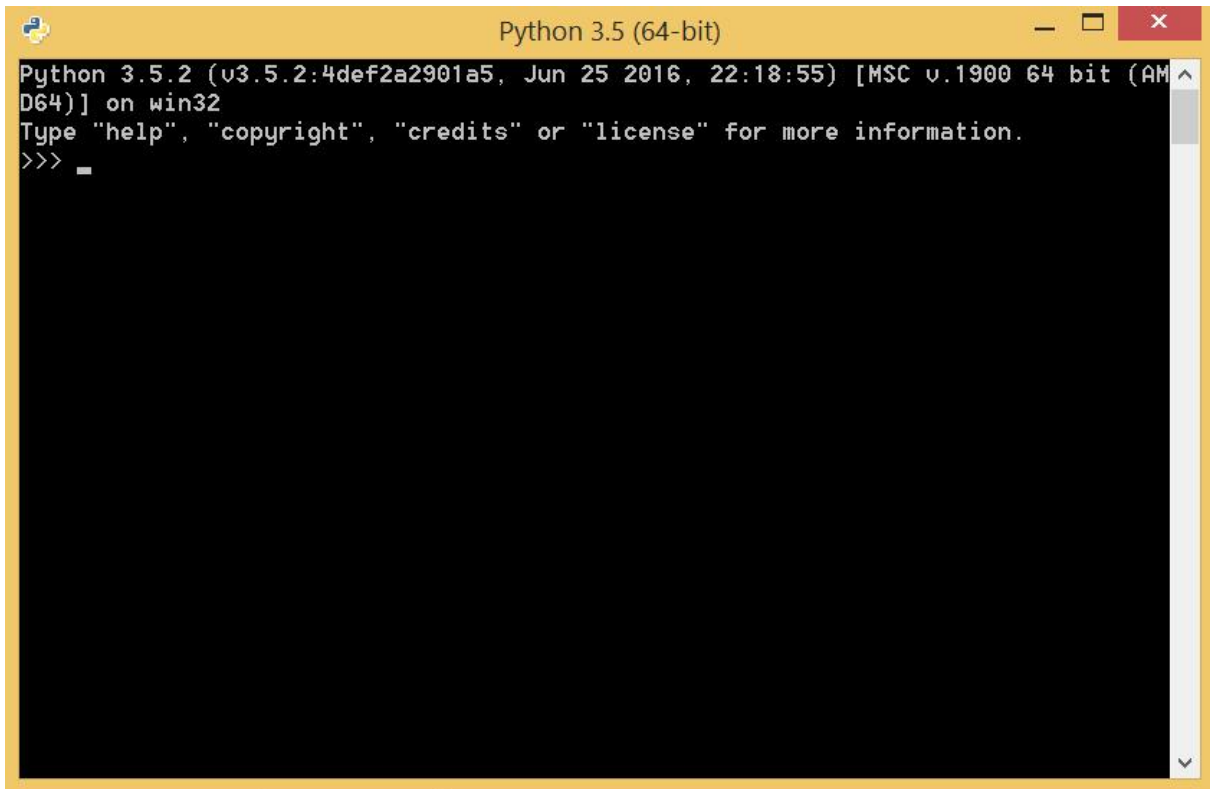


Um nach der Installation Python zu starten, gehen Sie am einfachsten auf die (Windows Kachel-Seite, suchen dort nach Python und heften Sie beides, den Interpreter (schwarzes Logo) und die IDLE (eines der weißen Logos) durch rechten Mausklick an Ihre Taskleiste an, weil Sie diese beiden Programme häufig benötigen. Das Ergebnis sollte sein:



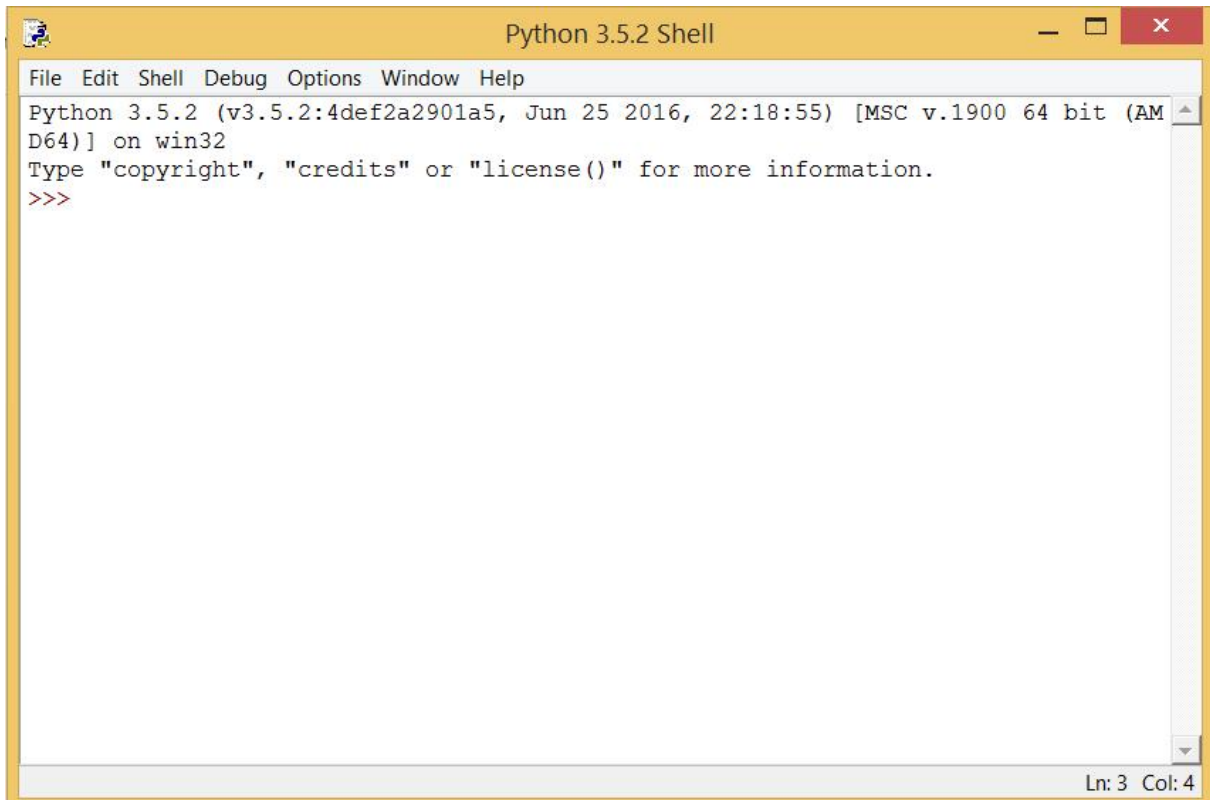
(Sorry, je nachdem, wie Ihr System aussieht, können die Installationsschritte sehr verschieden sein, Versuchen Sie, die Installation hinzubekommen.)

Klicken Sie als erstes einmal auf das „schwarze Logo und starten Sie damit den Python Interpreter:



```
Python 3.5 (64-bit)
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Wenn Sie jetzt das weiße Python Logo klicken, sehen wir, dass auf „großartige“ Weise ein neues Fenster aufgeht:



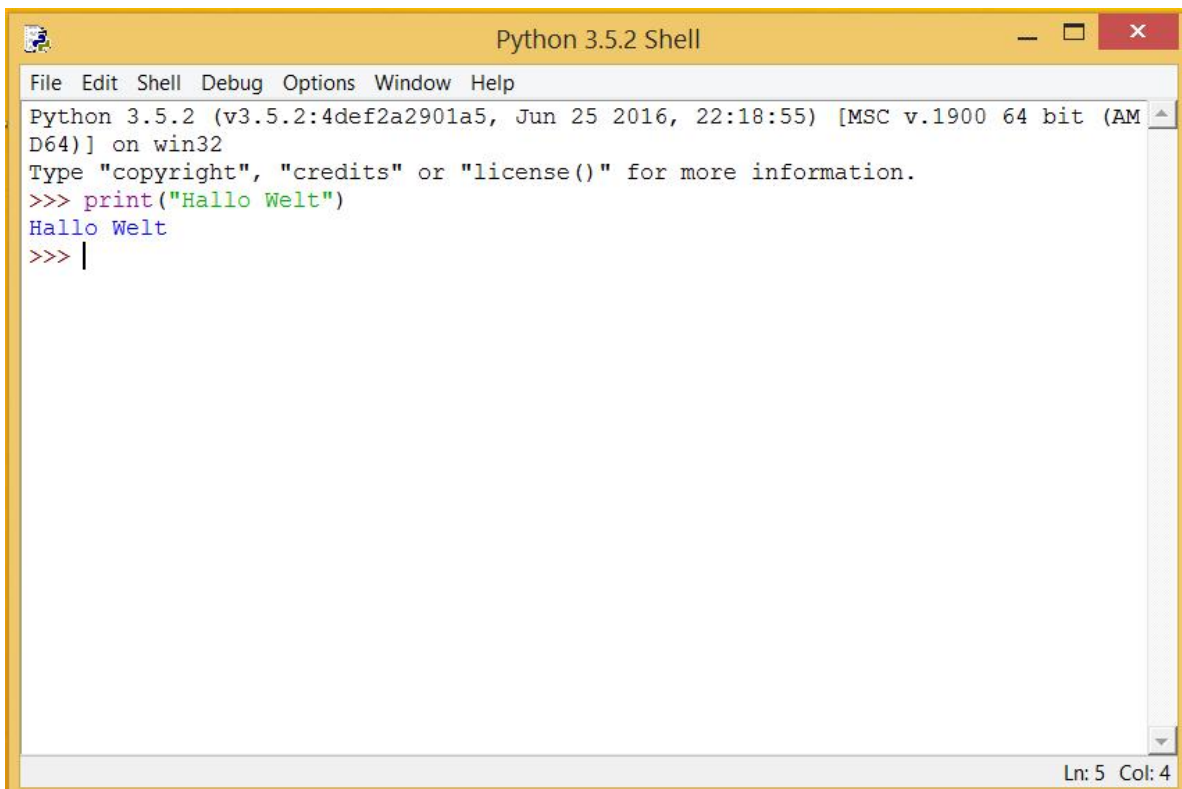
```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Ln: 3 Col: 4

Wenn Sie die Meldungen in diesen beiden Fenstern vergleichen, sehen Sie fast keine Unterschiede, lediglich einmal auf schwarzem Grund und einmal auf weißen. Tatsächlich ist dies auch dasselbe Programm, der Python Interpreter.

Dass weiße Fenster ist das **Hauptfenster von IDLE**, und was wir gerade vor uns sehen, wird das **Interpreter-Fenster (von IDLE)** genannt. Es meldet sich mit dem Fenster-Titel Python 3.5.2 Shell. Der Interpreter erlaubt uns Kommandos direkt in Python einzugeben, uns sobald wir ein Kommando eingeben (*mit Return abschließen*), wird Python das Kommando ausführen, und uns das Ergebnis davon ausgeben. Wir werden dieses Interpreter-Fenster häufig benutzen, wenn wir Python „erforschen“: es ist sehr praktisch, weil wir unsere Ergebnisse unmittelbar heraus bekommen. Wenn's dir hilft, dann stelle dir das als einen sehr leistungsfähigen (Taschen-) Rechner vor.

Wir wollen jetzt was probieren! Aus „traditionellen Gründen“ wollen wir Python dazu bringen, die unsterblichen Worte "Hallo Welt" zu sagen.



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hallo Welt")
Hallo Welt
>>> |
```

Diese '>>>' Zeichen haben für uns die Bedeutung der Aufforderung, etwas einzugeben. Python ist bereit ein neues Kommando einzulesen und zeigt uns das so visuell an. Wir bemerken auch, wenn wir Kommandos eingeben (*und mit RETURN abschließen*), dass Python seine Ausgabe uns unmittelbar danach ausgibt.

*print() ist eine sogenannte Funktion, durchaus mit Funktionen (d.h. Abbildungen) in der Mathematik vergleichbar sind. Gekennzeichnet sind diese durch ihren Namen und dem runden Klammerpaar. Im Gegensatz zur Mathematik können Funktionen in der Programmierung allerdings auch keine Argumente haben und/oder auch keinen Funktionswert zurückliefern. Definieren tut man eigene Funktionen durch ein spezielles Schlüsselwort `def name()`: wobei `name` den Namen (oder Bezeichner wie `print`) angibt. Noch eins: In lila markierte Namen, wie `print` sind im Interpreter vordefinierte Funktionen, sogenannte *Python Builtins*.*

Wie wir sehen, werden die verschiedenen Wörter und Sätze unterschiedlich eingefärbt: Dies ist eine der Unterstützungen von IDLE um Tippfehler zu vermeiden oder einfach zu erkennen.

IDLE erkennt bestimmte Sprachelemente von Python und markiert diese farblich: Die wichtigsten sind:

Kommentare beginnen immer mit einem #

Python Keywords (Schlüsselworte)

Python Definitionen

Python Zeichenketten (Strings)

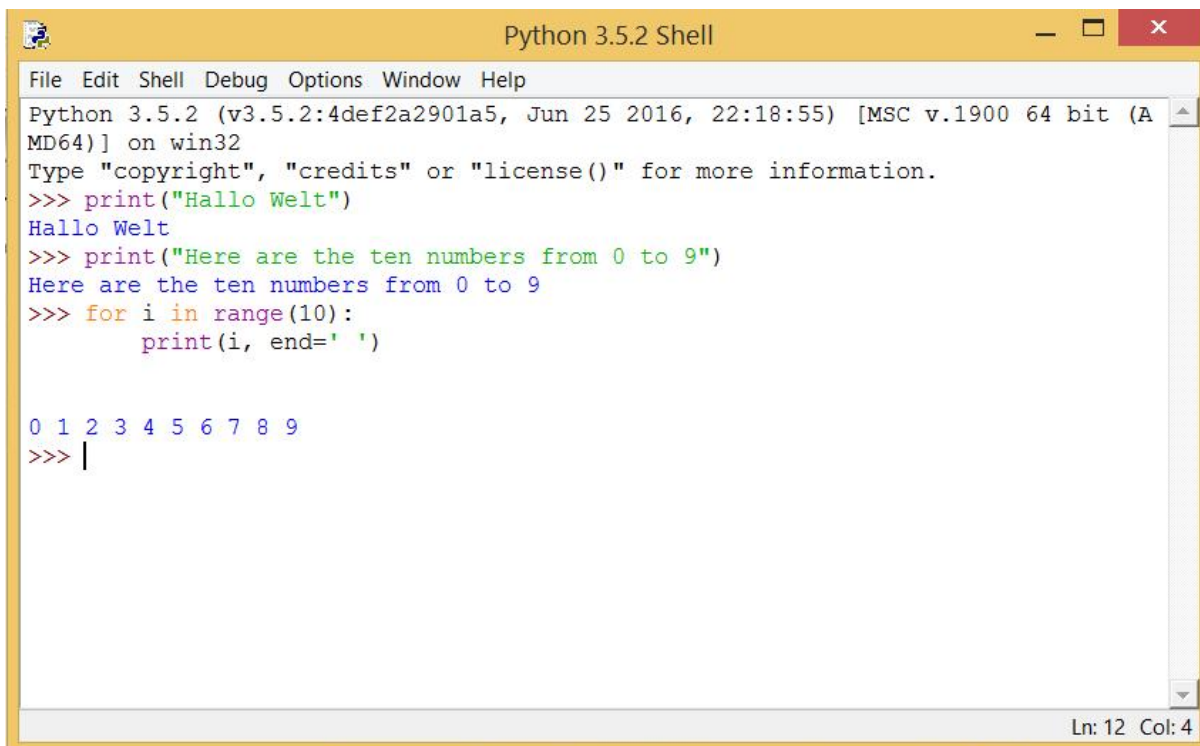
Python Builtins (spezielle vordefinierte Funktionen)

sonstiger Programmtext

Es ist nicht schlimm, wenn Sie im Moment noch nicht alles verstehen. Wir werden dies Zug um Zug kennenlernen. Man nennt IDLE einen so genannten „Syntax-gesteuerter Editor“, d.h. IDLE kennt die Syntax von Python und nutzt dies. Dies dient der Unterstützung des Programmierers.

Noch zwei Kleinigkeiten: Im Python Interpreter-Fenster sind ALLE Programm-Ausgaben blau markiert. Kommentare (rot markiert) sind Anmerkungen des Programmierers für andere Programmierer, sind für den Interpreter aber gänzlich ohne Bedeutung – werden überlesen. Sie gelten jeweils vom # bis zum Ende der Zeile – die folgende Zeile wird dann wieder interpretiert.

Ok, das schaut ja noch ganz einfach aus. Probieren wir noch ein paar Kommandos. Im folgenden Bild



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hallo Welt")
Hallo Welt
>>> print("Here are the ten numbers from 0 to 9")
Here are the ten numbers from 0 to 9
>>> for i in range(10):
    print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
>>> |
```

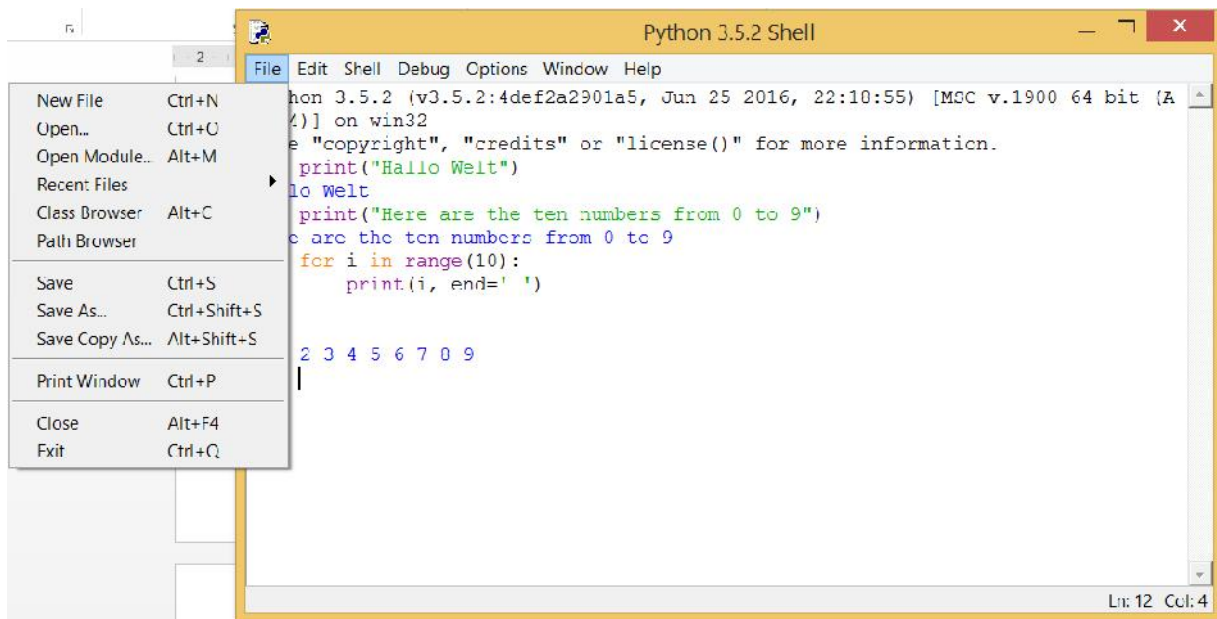
sehen wir, was herauskommt, wenn wir ein paar Kommandos ausführen lassen. Mach dir noch keine Sorgen über die exakten Regeln für das Schreiben von Programmen: die Idee ist jetzt, dass wir mit Python experimentieren können, indem wir einfach Kommandos eingeben. Wenn das nicht funktioniert, dann können wir unsere Fehler ausbessern und es noch einmal versuchen.

Wenn du bis hierher gekommen bist, dann weißt du nun genug um anfangen zu können mit Python herumzuspielen! Schlag einfach eins von den Tutorials der [Python For Beginners](#) Web-Seite auf und begib dich mit dem Interpreter auf Forschungsreise. Kein Zeitlimit!
grins

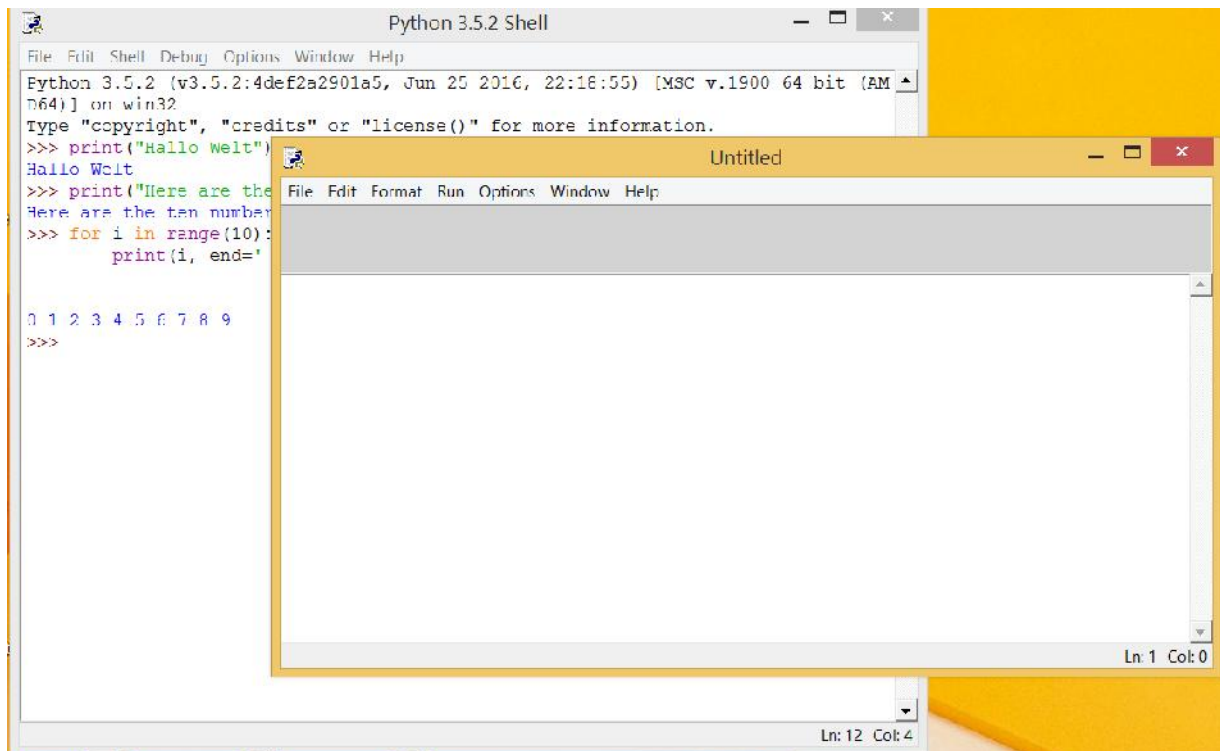
Nachdem wir nun lange genug herumgepaddelt sind, würden wir gerne fragen: ok, das ist ja ganz nett, aber wenn wir Python schließen und es dann von neuem starten, wie bringen wir dann den Computer dazu, sich daran zu erinnern, was wir eingetippt haben?

Die Lösung ist ein wenig subtil: wir können nicht direkt abspeichern, was im Interpreter-Fenster ist, weil das sowohl unsere Eingaben wie auch die Antworten des Systems enthält. Wir würden gerne eine Datei herstellen, die nur unsere Kommandos enthält, die wir dann auch auf der Festplatte abspeichern können. Wenn wir dann Lust dazu haben, können wir später die Datei öffnen und Python die Kommandos ausführen lassen. So sparen wir uns die Zeit um das ganze Zeug immer wieder neu einzutippen.

Versuchen wir es. Starten wir mit einer leeren Fläche, indem wir ein neues Fenster aufmachen (*New File*).

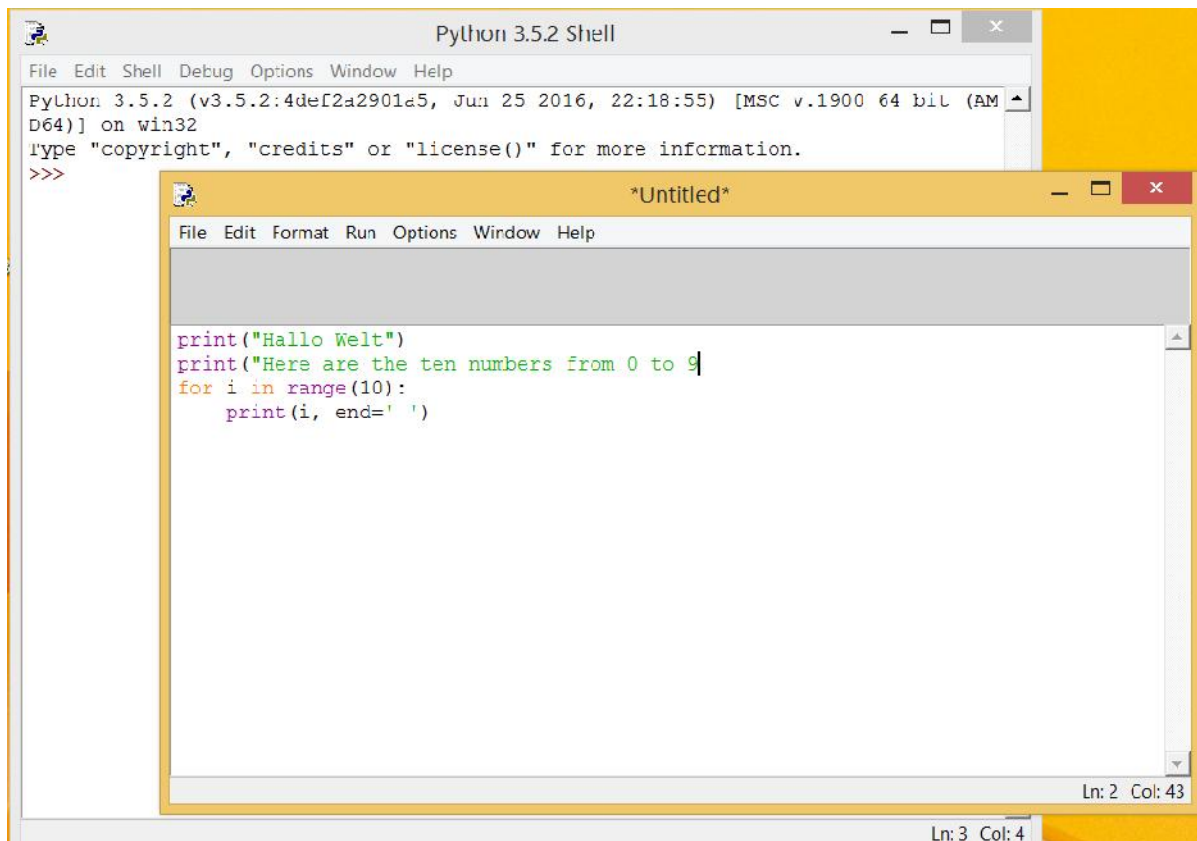


Hier ist das Ergebnis von diesem Menü-Kommando:



Wir bemerken, dass nichts in diesem neuen Fenster ist. Das heißt, dass diese Datei nur für unsere Kommandos gedacht ist. Python wird sich nicht mit seinen Antworten einmischen, das heißt, zumindest so lange nicht, bis wir es von ihm verlangen. Ich nenne dieses das **"Programm"-Fenster** um es vom Interpreter-Fenster zu unterscheiden. *Von diesen Programm-Fenstern können wir übrigens mehrere öffnen und gleichzeitig offen haben.*

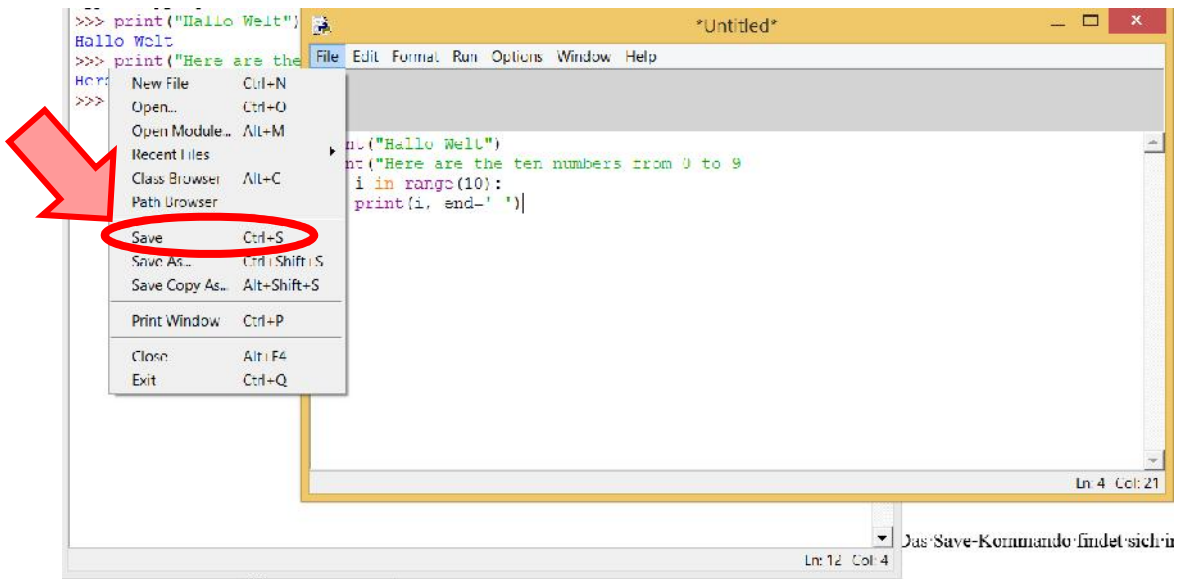
Was wir vorhin wollten, war, einiges von dem Zeug, das wir im Interpreter-Fenster ausprobiert hatten abzuspeichern. Tun wir das, indem wir diese Kommandos eintippen (oder mit copy/paste - kopieren/einfügen - ins Programmfenster übertragen).



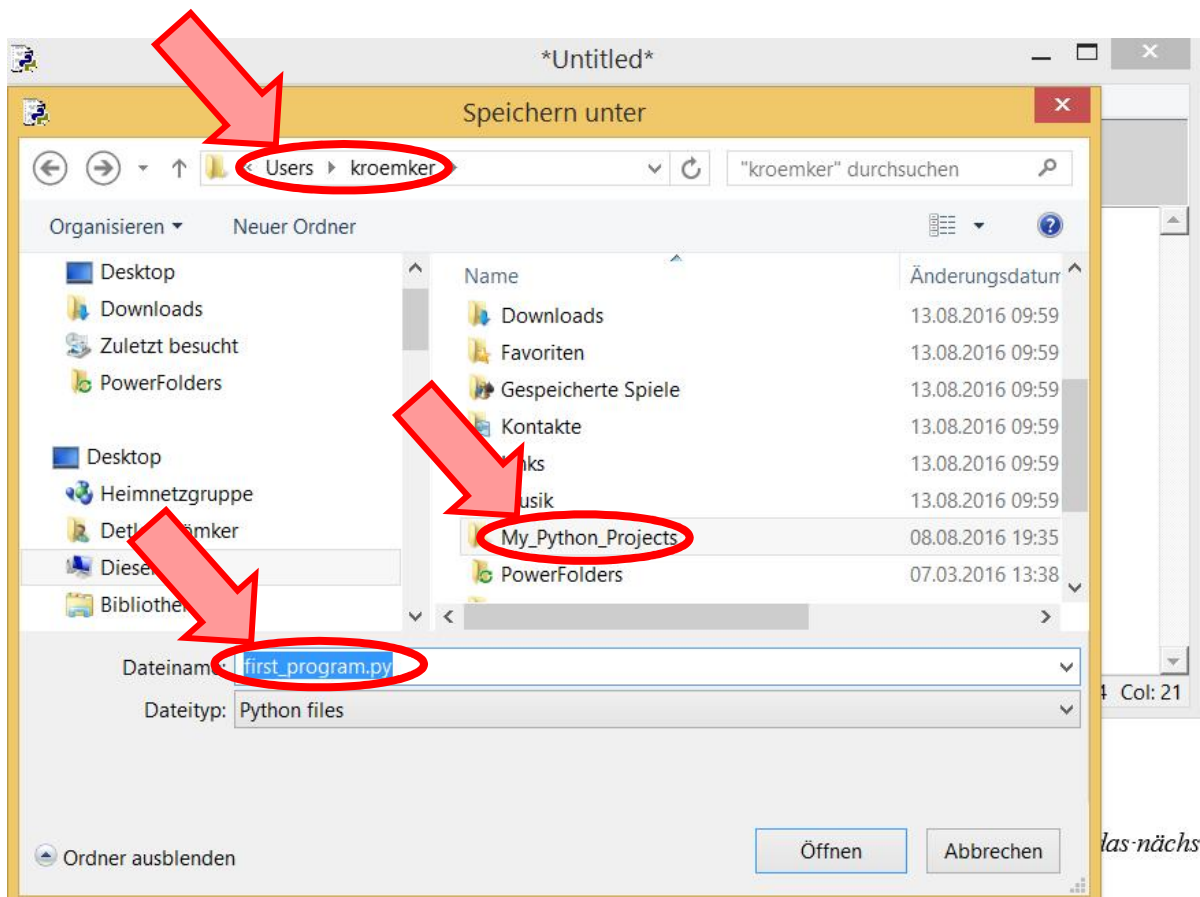
Ok, wir haben das mit kopieren/ausschneiden geschafft. Eine ganz wichtige Sache, die wir beachten müssen, ist, dass wir die '>>>'-Prompts loswerden müssen, denn die gehören nicht zu unserem Programm. Der Interpreter benützt sie ja nur, um uns mitzuteilen, dass wir im Interpreter sind. Aber jetzt editieren wir eine separate Datei, und so müssen wir die künstlichen Einfügungen des Interpreters entfernen.

PS: Übrigens: Die Breite des Programm-Fensters ist genau 80 Zeichen (die aktuelle Spalte sehen Sie unten rechts nach Col: [Column], wenn Sie sie nicht verändern ... länger sollte auch eine Zeile im Programmtext nicht sein.

Wir wollen die Datei nun sichern (abspeichern). Das Save-Kommando findet sich im „File-Menü“:

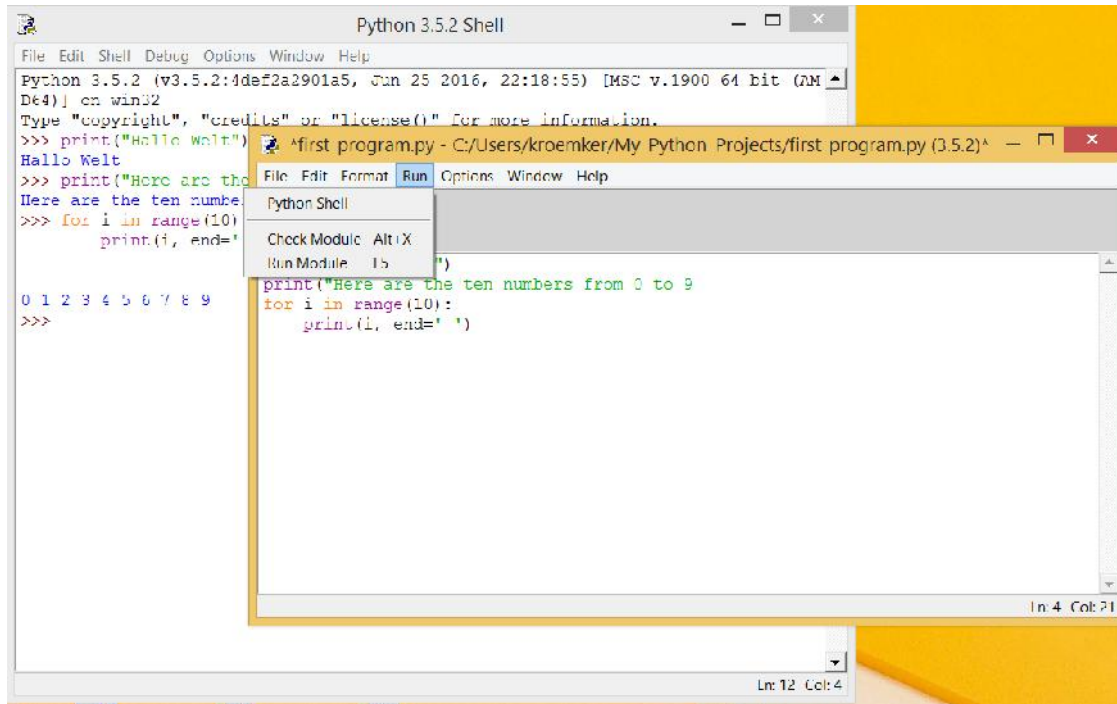


Achtung: Je nachdem, wie Sie Python installiert haben, kann es beim Speichern zu Konflikten kommen, derart, dass Sie nicht genügend Rechte haben, um ein Speichern an der von IDLE vorgeschlagenen Stelle durchzuführen. In diesem Fall folgen Sie am besten den Vorschlägen und speichern das Programm unter Ihrem Account, z.B. bei mir in dem Ordner \Users\kroemker\My_Python_Projects\ unter dem Dateinamen first_program.py. Hierzu habe ich den Ordner „My_Python_Projects“ extra angelegt, um da auch später alle Programmierprojekte zu speichern.



Das Python Programm bitte immer mit der Dateierdung **.py** speichern, die aber normalerweise, d.h. wenn das nächste Feld Dateityp auf „**Python files**“ steht auch automatisch ergänzt wird.

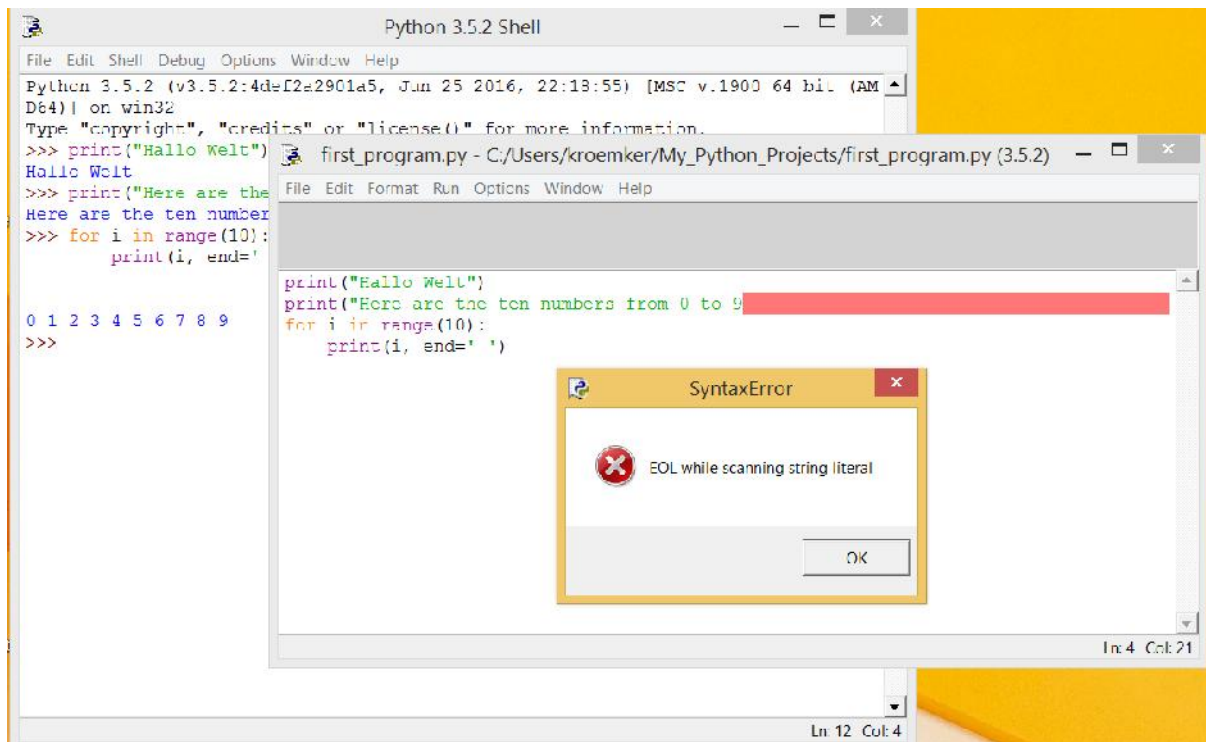
Jetzt, wo wir das Programm gesichert haben, entsteht die Frage: Wie können wir es nun ausführen? Wenn wir die Menüs in unserem Programm-Fenster anschauen,



sehen wir in dem Drop-Down-Menue „Run“ die Menü-Option "**Run Module**", und das ist genau das, was wir tun wollen. Was wir sehen möchten, ist, dass Python ein Kommando des Programms nach dem anderen ausführt und die Ergebnisse jeweils gleich ins Interpreter-Fenster schreibt.

*IDLE nennt unsere Programme hier **Module**. Grundsätzlich nutzt man in der Programmierung Module, um größere Programme zu strukturieren und in Teile (die Module) zu zerlegen, die z.B. separat testbar sind. Unser Programm besteht aus einem Modul und insofern lassen wir einen Modul laufen: Übrigens: In Python ist jedes abgespeicherte Programm(teil) ein Modul!*

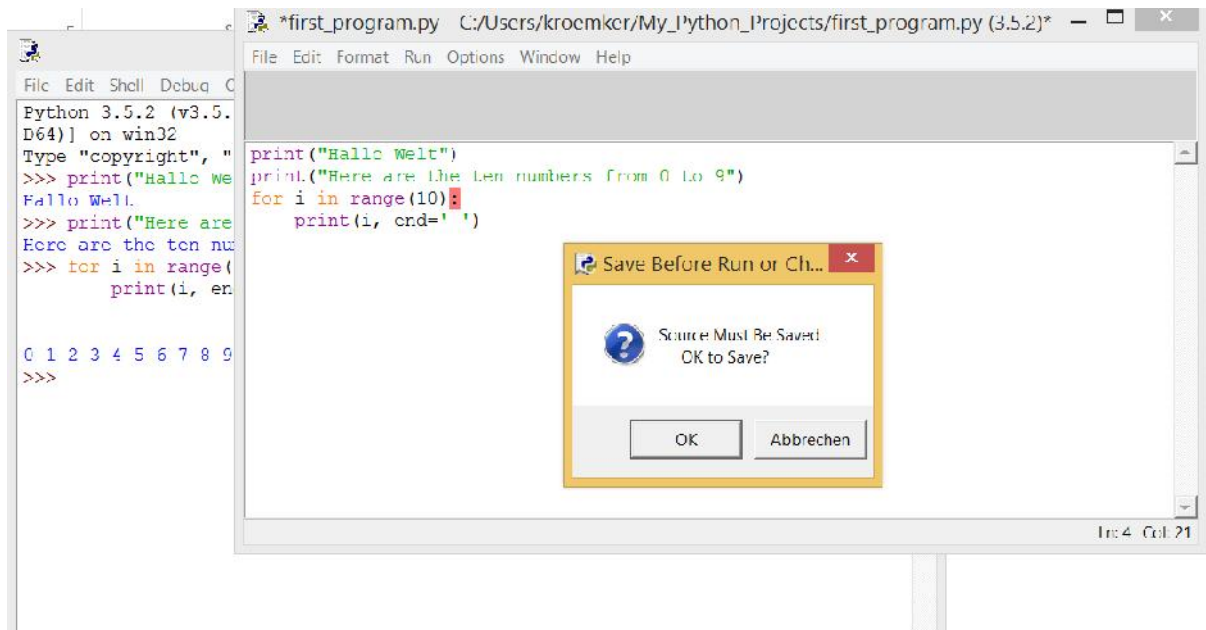
So nebenbei, eine Sache die hier noch zu beachten ist, ist die, dass ich hier einen **Tippfehler** machte. Ich habe nicht genau das hineinkopiert, was ich vorher im Interpreter-Fenster geschrieben hatte. Hat das irgendeine Folgen?



Ooops. Hier ist ein Beispiel von etwas, das Python einen **Syntax-Fehler** nennt ("SyntaxError"). Python erkennt, dass wir einen Tippfehler produziert haben, und nennt uns den Fehler, damit wir einen genaueren Blick auf unser Programm werfen. Die Leute, die Python entworfen haben, meinten, es wäre besser, dass Python auf Fehler aufmerksam macht anstatt herumzuraten, was der Programmierer gemeint haben könnte. Das ist das Prinzip von 'Klarheit gegen Beiläufigkeit'. Es gibt bestimmte Regeln, denen Python gehorcht, die angeben was richtig ist und was verdächtig aussieht. Je besser wir die Sprache können, desto mehr Gefühl bekommen wir für diese Regeln. Und auch wenn dir das verdächtig vorkommt, ja, das ist eine Art von Grammatik. *grins*

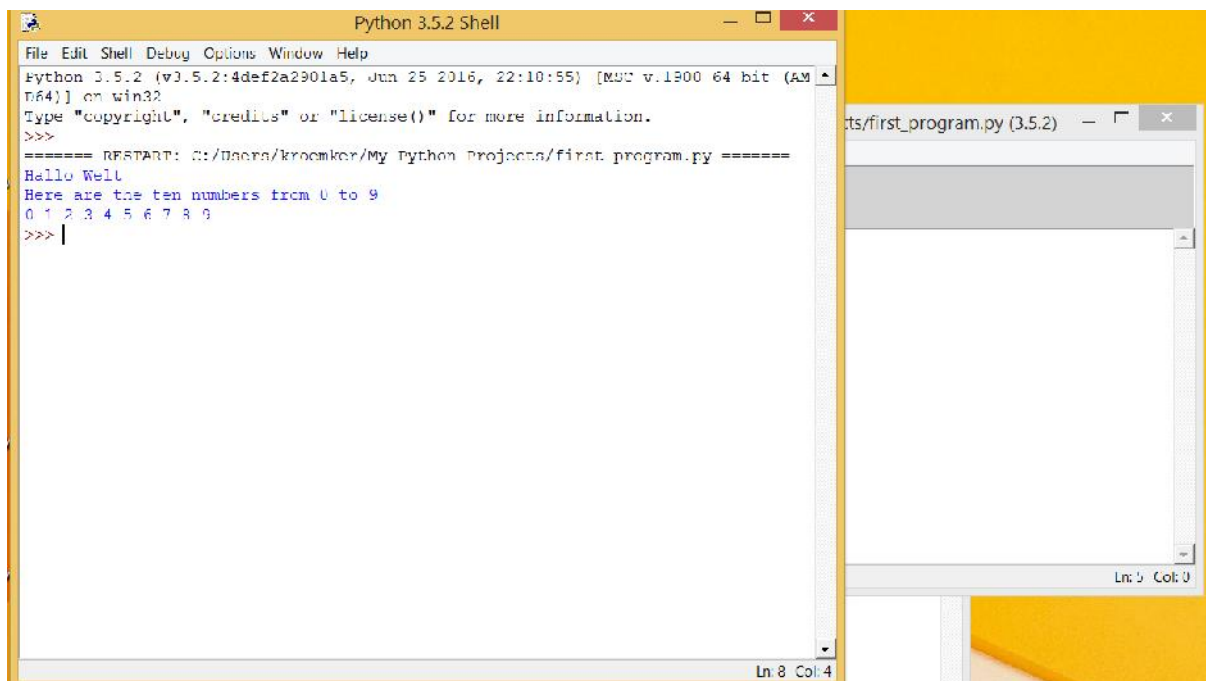
Python ist oft „schlau genug“ um uns direkt auf das Problem hinzuweisen, und hier sagt es uns, „*EOL (heißt „End of Line“) while scanning string literal*“, d.h. dass wir etwas am Ende der Zeile etwas vergessen haben. In diesem Fall müssen wir ein zusätzliches Anführungszeichen und die Klammer zu anfügen. Tun wir das also gleich.

Ok, wir können sagen, wir haben diesen dummen Tippfehler ausgebessert. Versuchen wir nochmals das Programm laufen zu lassen.



Noch ein Haken! Aber nicht so kompliziert, nur dumm (???). Ein Punkt, der uns vielleicht manchmal auf die Nerven geht, ist, dass IDLE möchte, dass wir jedes Programm-Fenster sichern bevor wir das Programm laufen lassen. Es soll sicherstellen, dass wir unser Programm in der letzten Version auch wirklich abspeichern bevor wir beginnen, das Programm laufen zu lassen und das geht einfach in dem wir „OK“ drücken.

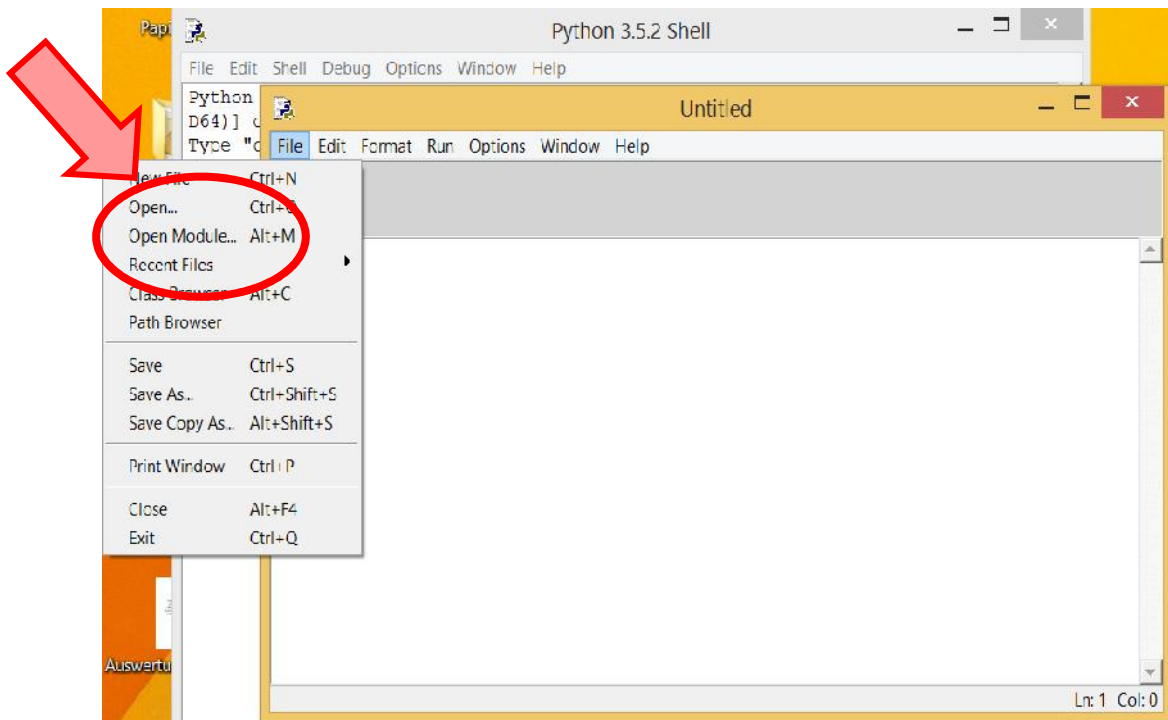
Man sagt ja, das dritte Mal wirkt Wunder, also versuchen wir noch einmal, es laufen zu lassen. Es sollte jetzt hoffentlich gut gehen.



Tatsächlich, das Interpreter-Fenster kommt nach vorn. Es sagt, dass es neu gestartet wurde (===== RESTART: C:/Users/kroemker/My_Python_Projects/first_program.py =====) und gibt die von uns programmierten Zeilen im Interpreter-Fenster aus.

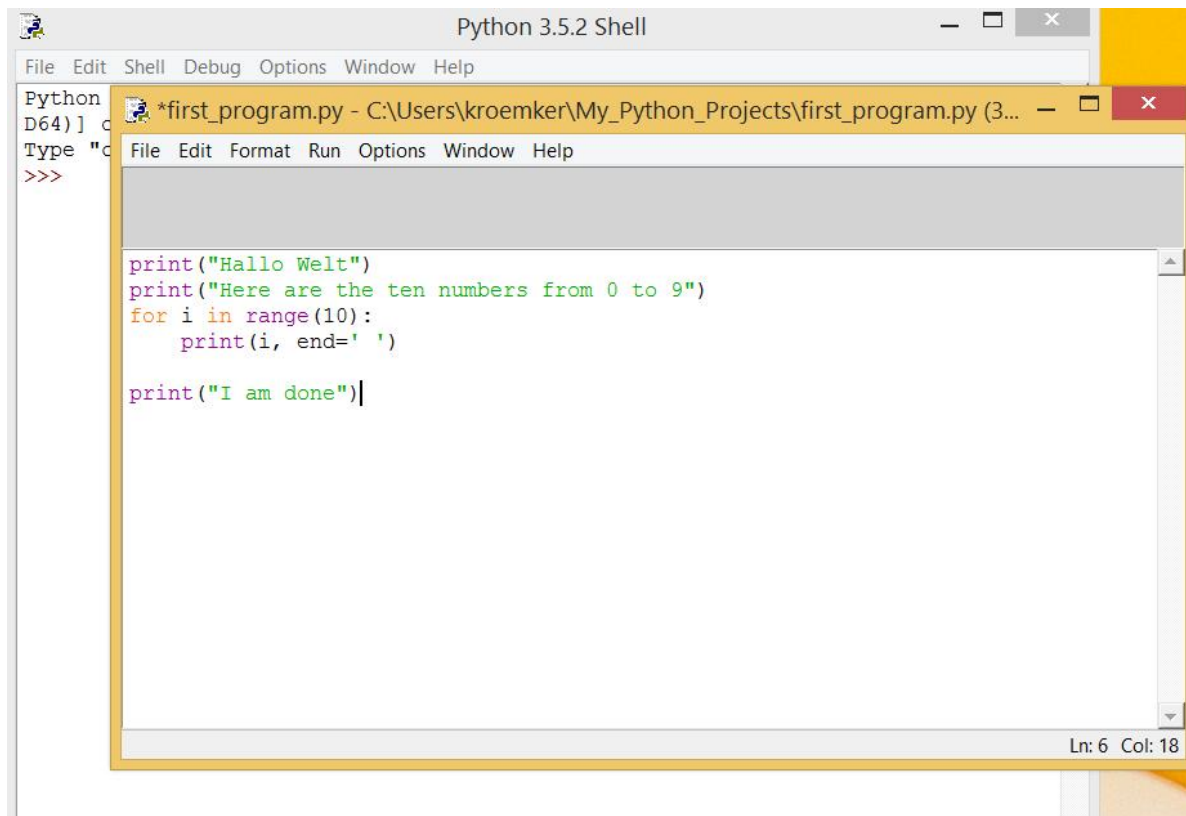
Wenn wir also mit Python spielen, werden wir immer wieder den Modus umschalten, vom Interpreter-Fenster zum Programm-Fenster und zurück. Der Grund dafür ist, dass wir das Interpreter-Fenster wie ein kleines Laboratorium benützen können, in dem wir experimentell ganz kleine Programme ausprobieren können. Wenn wir damit zufrieden sind (oder wenn wir müde sind), können das, was wir gelernt haben, in einer Programm-Datei abspeichern.

Das geht natürlich von der Annahme aus, dass wir diese Datei später wieder öffnen können; es wäre ja ganz dumm, ein Programm zu sichern ohne imstande zu sein, es später wieder zu laden. Schauen wir uns das noch an, und dann hören wir für heute auf. Wir schließen alle Fenster von IDLE und starten es neu. So beginnen wir wieder mit einer leeren Fläche.



Wir finden die Open-Kommandos im File-Menü: Am einfachsten ist es, unter „Recent Files“ dasjenige zu finden, in dem wir zuvor abgespeichert hatten. Später lernen wir noch den genauen Unterschied zwischen Open und Open Module und Recent Files.

Wenn alles gut geht, sehen wir, dass sich in dem Programm-Fenster unser altes Programm öffnet:



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python
D64) ] c
Type "c
>>>

print("Hallo Welt")
print("Here are the ten numbers from 0 to 9")
for i in range(10):
    print(i, end=' ')

print("I am done")|

Ln: 6 Col: 18
```

Wir sind also wieder im Geschäft! Wir sichern unsere getane Arbeit und können sie später wieder öffnen. Das ist zwar nicht gerade weltbewegend, aber es ist entscheiden für alle, die mit Python länger als einen Tag lang herumspielen möchten. *grins*

Das ist eigentlich alles, was wir unbedingt über IDLE wissen müssen, um damit interessante Sachen anstellen zu können. Dieser Führer hat natürlich eine Menge Sachen von IDLE nicht behandelt. IDLE ist viel mehr als nur ein Editor, aber es dauert einige Zeit, alle seine Möglichkeiten zu erforschen. Deshalb hören wir für heute auf. Es gibt eine [IDLE Documentation](#) - Seite, die die fortgeschrittenere Verwendung von IDLE erklärt, für diejenigen, die eine unersättliche Neugier haben.

Es hat mir (*uns!*) Spaß gemacht und ich hoffe es war hilfreich!