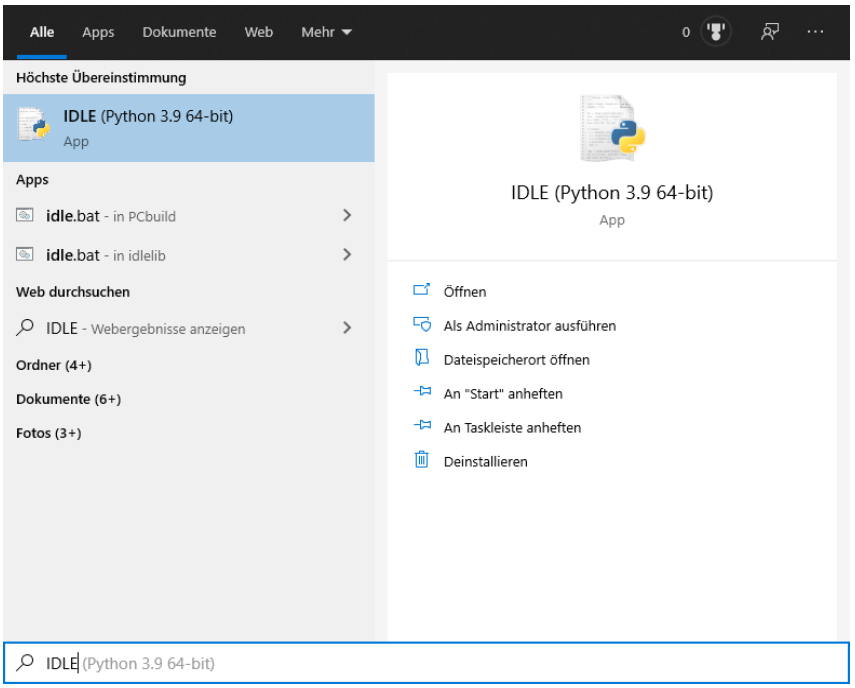
Danny Yoo's: Ein Tag Spielerei mit IDLE

(Originaldokument: http://www.hashcollision.org/hkn/python/idle_intro/index_ger.html)

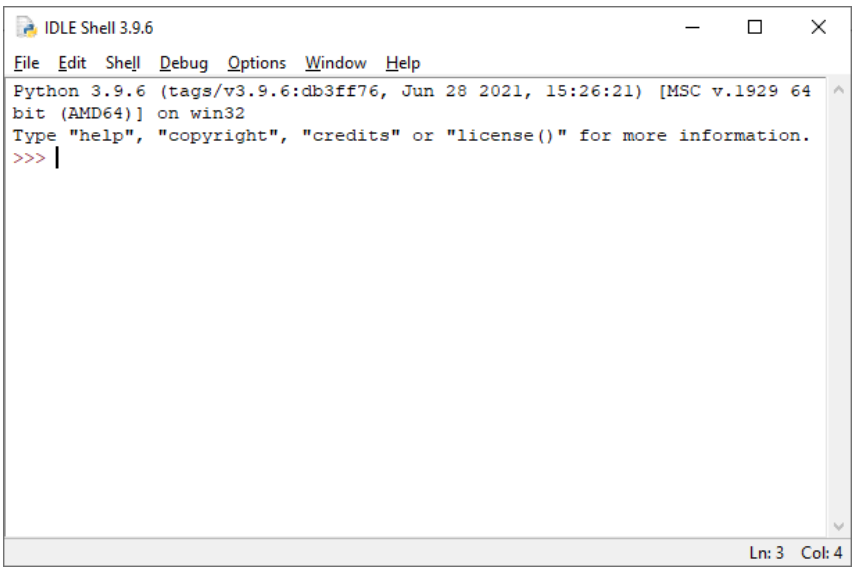
Dieses Dokument soll neuen Benutzern von Python helfen, die sich vielleicht ein bisschen desorientiert fühlen. Ein Frage, die einem da einfallen mag, ist: ok, wir haben Python installiert... ummm... was nun?

Vielleicht ist es nett einen "visuellen" Führer zu haben, um die anfänglichen Ängste zu reduzieren. Dafür ist dieses Dokument gedacht. Der Plan ist, eine kleine Session mit IDLE durchzumachen: mit dem **I**ntegrated **D**evel**o**pment **E**nvironment, 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.

Ok, angenommen, wir haben Python schon installiert. (Wenn nicht, dann können wir http://python.org besuchen und den neuesten Python Interpreter herunterladen). Zuerst möchten wir ihn nun starten! Wir können das erreichen, indem wir IDLE öffnen, das sich im Start-Menü in der gerade erzeugten Python-Programm-Gruppe finden sollte.

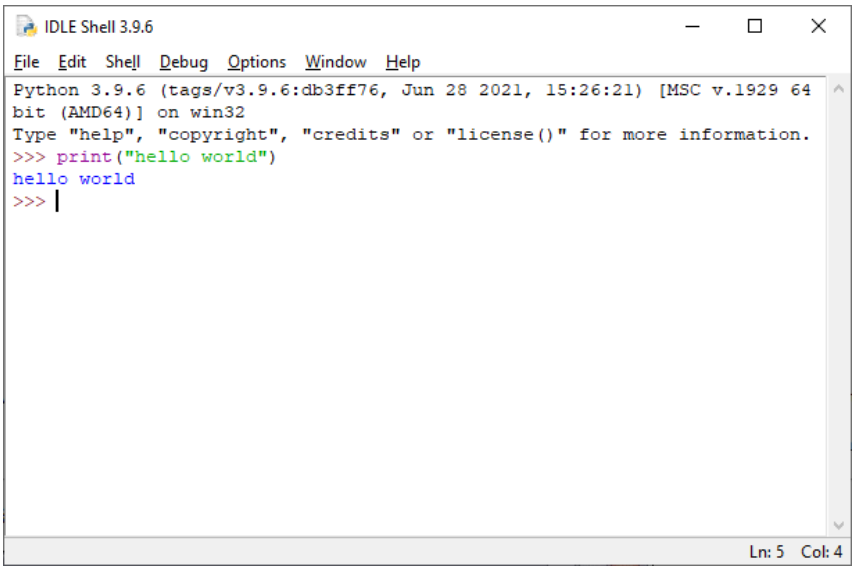


Und wir sehen, dass auf großartige Weise ein neues Fenster auf geht



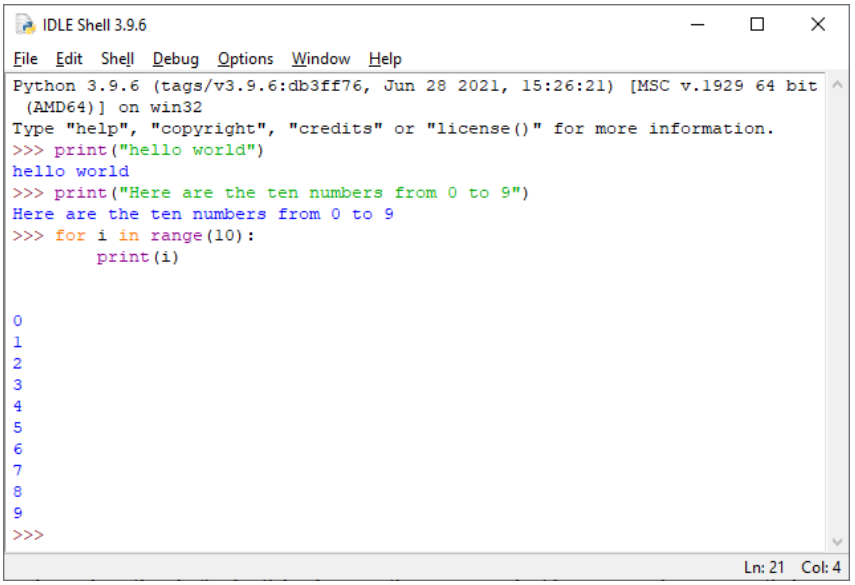
Das ist das Hauptfenster von IDLE, und was wir gerade vor uns sehen, wird das Interpreter-Fenster genannt. Der Interpreter erlaubt uns Kommandos direkt in Python einzugeben, und sobald wir ein Kommando eingeben, wird Python das Kommando ausführen, und uns das Ergebnis davon ausgeben. Wir werden dieses 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



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, dass Python seine Ausgabe uns unmittelbar danach ausgibt.

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



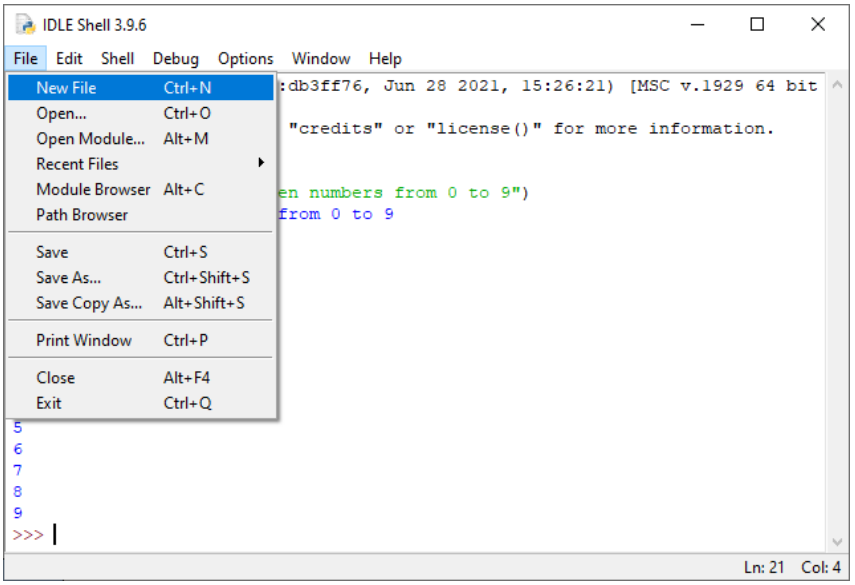
sehen wir, was herauskommt, wenn wir ein paar Kommandos ausführen lassen. Mach dir noch keine Sorgen über die exakten Regeln für das Schreiben von Programmen: die Idee ist jetzt, dass wir mit Python experimentieren können, indem wir einfach Kommandos eingeben. Wenn das nicht funktioniert, dann können wir unsere Fehler ausbessern und es nocheinmal 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 Webseite auf und begib dich mit dem Interpreter auf Forschungsreise. Kein Zeitlimit! *grins*

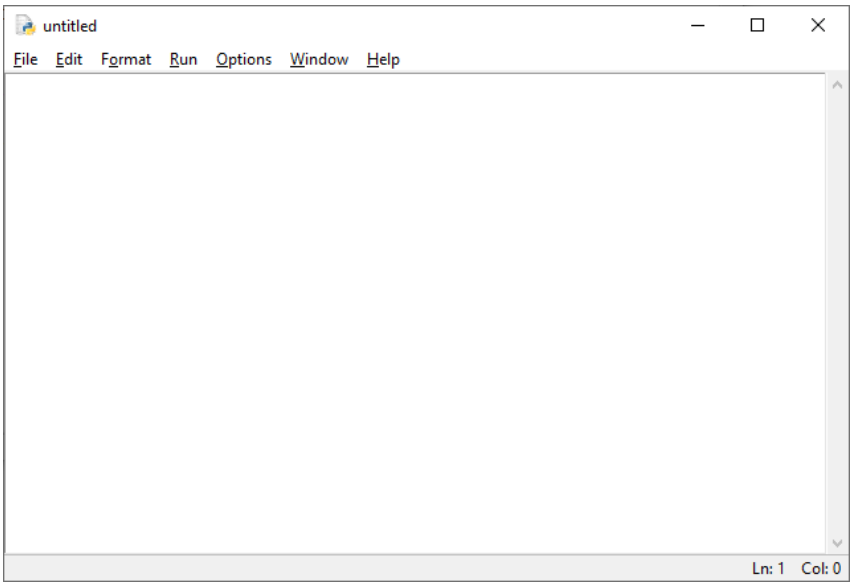
Nachdem wir nun lange genug herumgepaddelt sind, würden wir gerne fragen: ok, das ist ja ganz nett, aber wenn wir Python schließen und es dann von Neuem starten, wie bringen wir dann den Computer dazu, sich daran zu erinnern, was wir eingetippt haben?

Die Lösung ist ein wenig subtil: wir können nicht direkt abspeichern, was im Interpreterfenster ist, weil das sowohl unsere Eingaben wie auch die Antworten des Systems enthält. Wir würden gerne eine Datei erstellen, 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.

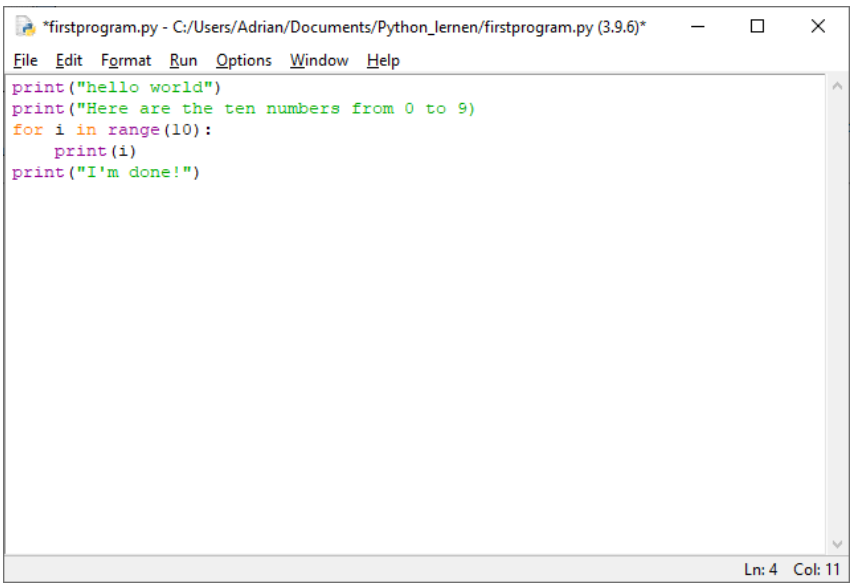


Hier ist das Ergebnis von diesem Menü-Kommando:



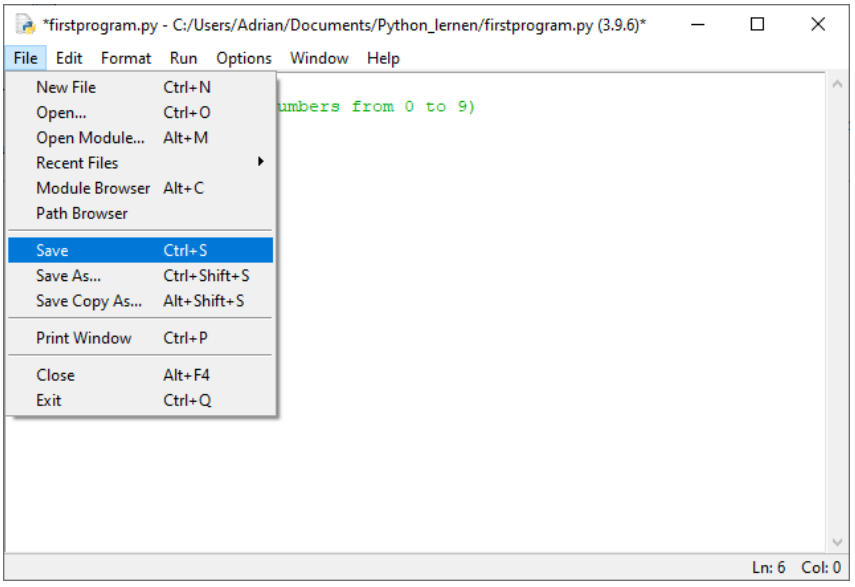
Wir bemerken, dass nichts in diesem neuen Fenster ist. Das heisst, dass diese Datei nur für unsere Kommandos gedacht ist. Python wird sich nicht mit seinen Antworten einmischen, das heißt, zumindest so lange nicht, bis wir es von ihm verlangen. Ich nenne dieses das "Programm"-Fenster um es vom Interpreter-Fenster zu unterscheiden.

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).

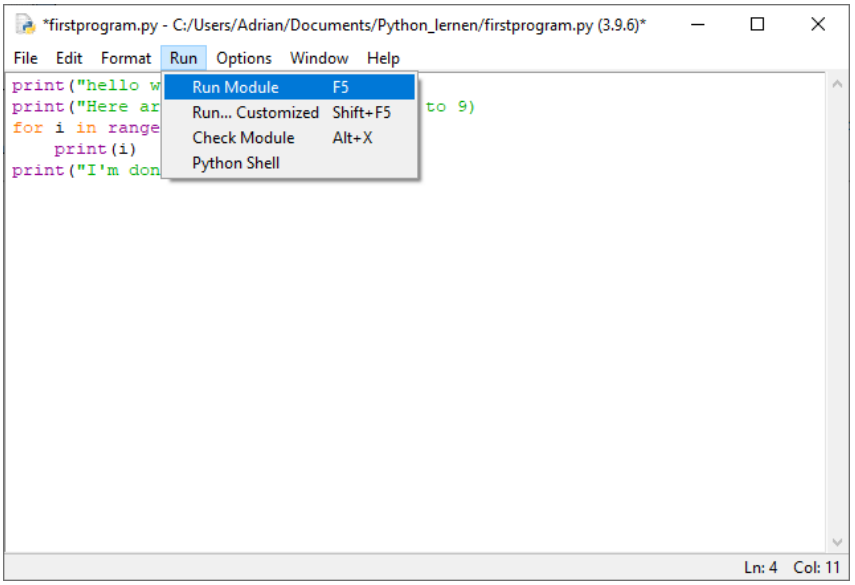


Ok, wir haben das mit kopieren/ausschneiden geschafft. Eine ganz wichtige Sache, die wir beachten müssen, ist, dass wir die '>>>'-Prompts loswerden müssen, denn die gehören nicht zu unserem Programm. Der Interpreter benutzt sie ja nur, um uns mitzuteilen, dass wir im Interpreter sind. Aber jetzt editieren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

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

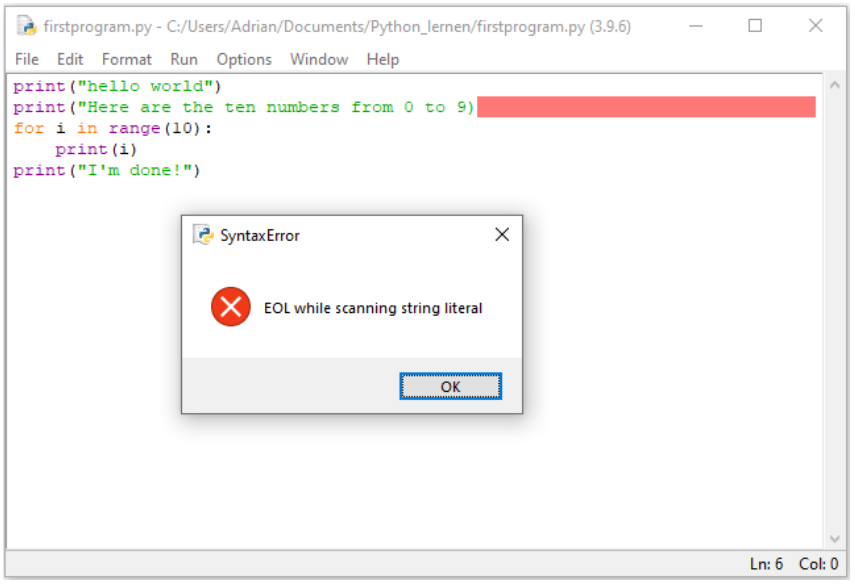


Jetzt, wo wir das Programm gesichert haben, wie können wir es nun ausführen? Wenn wir die Menüs in unserem Programm-Fenster anschauen,



sehen wir eine Menü-Option "Run Module", und das ist genau, 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.

So nebenbei, eine Sache die hier zu beachten ist, ist die, dass ich hier einen Tippfehler machte. Ich habe nicht genau das hineinkopiert, was ich vorher im Interpreterfenster geschrieben habe. Hat das irgendwelche Folgen?

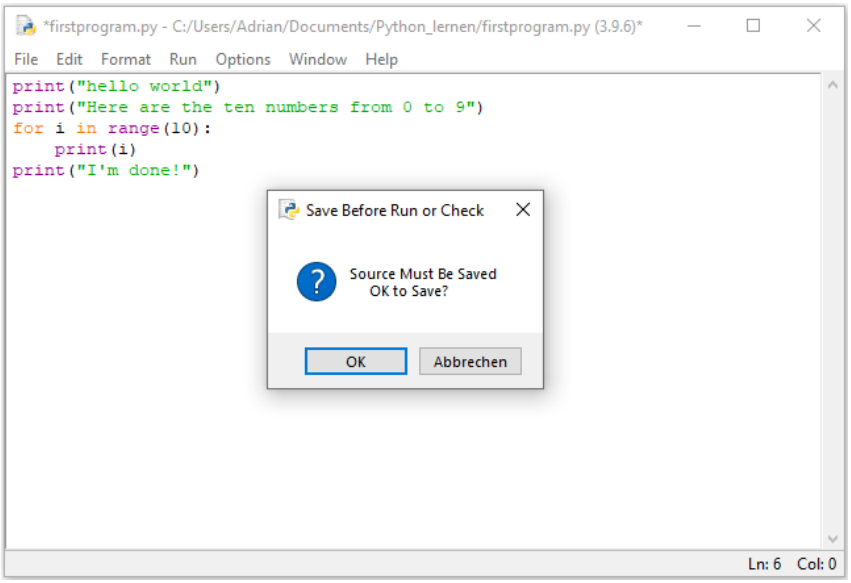


Ooops. Hier ist ein Beispiel von etwas, das Python einen Syntax-Fehler nennt. ("syntax error"). Python sieht, dass wir einen Tippfehler produziert haben, und warnt uns, damit wir einen genaueren Blick auf unser Programm machen. Die Leute, die Python entworfen haben, meinten, es wäre besser, dass Python 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 ausschaut. 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, dass wir etwas am Ende der Zeile vergessen haben. In diesem Fall müssen wir ein zusätzliches Anführungszeichen einfügen. Tun wir das also gleich.

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



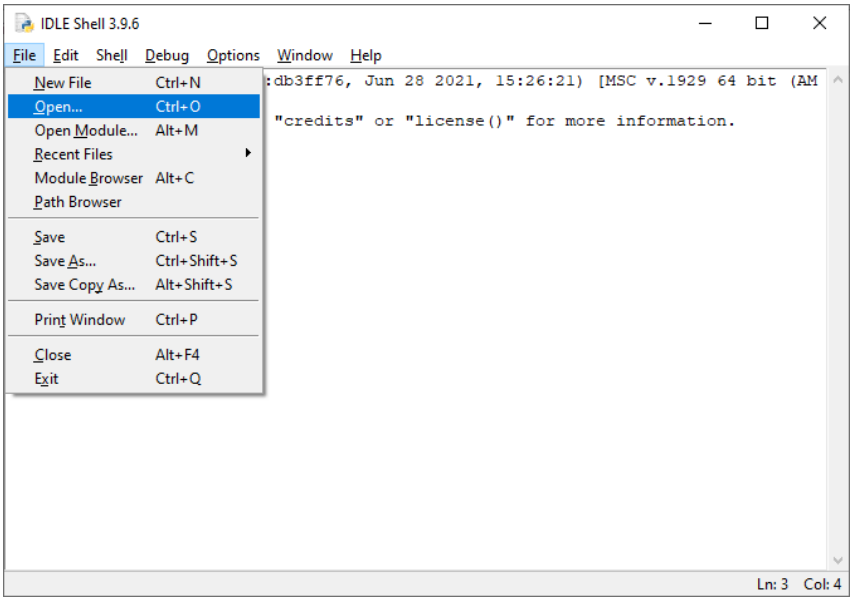
Noch ein Haken! Aber nicht so kompliziert, nur dumm. Ein Punkt, der uns vielleicht manchmal auf die Nerven geht, ist, dass IDLE möchte, dass wir jedes Programm-Fenster sichern bevor wir das Programm laufen lassen; das ist einfach eine Sache des Benutzer-Interfaces. Es soll sicherstellen, dass wir unser Programm auch wirklich abspeichern bevor wir beginnen, das Programm laufen zu lassen.

Versuchen wir noch einmal, es laufen zu lassen. Es sollte jetzt hoffentlich gut gehen.

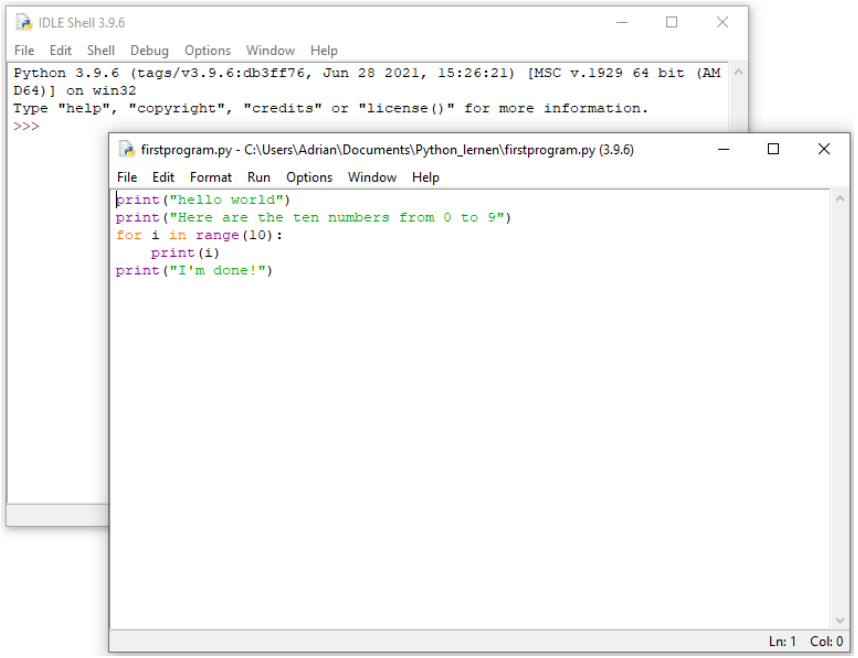
Wenn wir also mit Python spielen, werden wir immer wieder den Modus umschalten, vom Interpreter-Fenster zum Programm-Fenster und zurück. Der Grund dafür ist, dass wir das Interpreterfenster wie ein kleines Laboratorium benützen können, in dem wir experimentell ganz kleine Programme ausprobieren können. 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 das Open-Kommando im File-Menü:



und wenn alles gut geht, sehen wir, dass sich ein neues Programmfenster öffnet:



mit unserem alten Programm. Wir sind also 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 sich 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 Spaß gemacht und ich hoffe es war hilfreich!