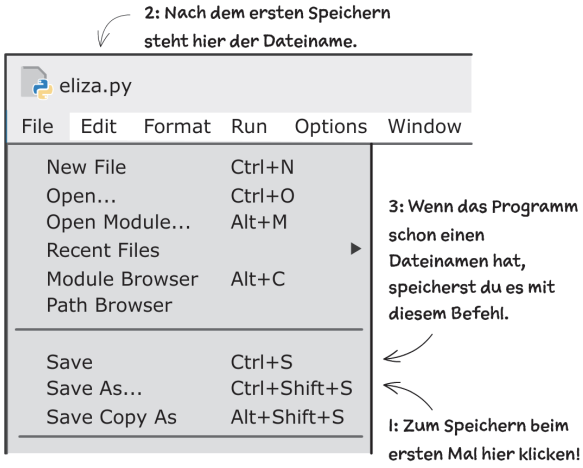


Du bist soweit. Beginnen wir mit dem Programmieren.

1.4 Projekt: Mini-Eliza

In diesem Projekt entwickeln wir in fünf Schritten eine Mini-Version eines Chatbots. Das Programm ist ganz einfach und wird bestimmt keinen Turing-Test bestehen. Dennoch zeigt es einige Features von Chatbots. Falls du noch nie mit Python programmiert hast, wirst du jetzt einige grundlegende Programmiertechniken kennenlernen.

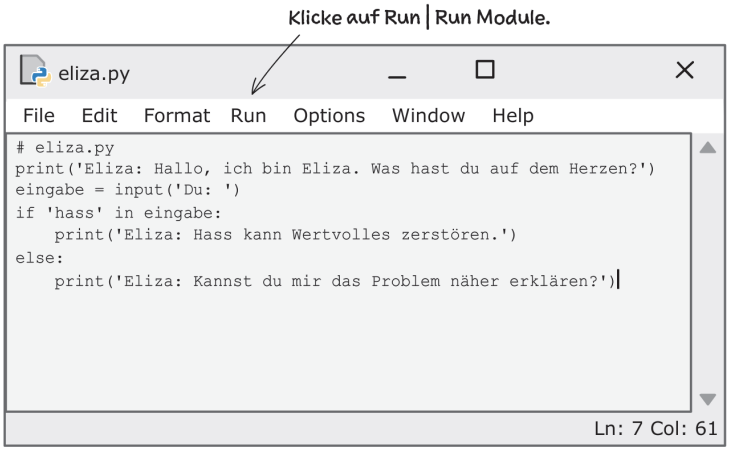
1.4.1 Schritt 1: Auf eine Eingabe reagieren

Ein Gespräch zwischen Mensch und Computer ist ein Wechselspiel von Texten. Das folgende Programm schreibt etwas auf einen Bildschirm, wartet auf eine Benutzereingabe und liefert dann eine passende Antwort. Das ist alles. Aber es ist ja auch nur der erste Schritt auf dem Weg zum Chatbot. Gib den Programmtext in das Editorfenster ein und speichere ihn, indem du im Menü FILE auf den Befehl SAVE klickst. Beachte, dass einige Zeilen um 4 Leerstellen eingerückt sind.

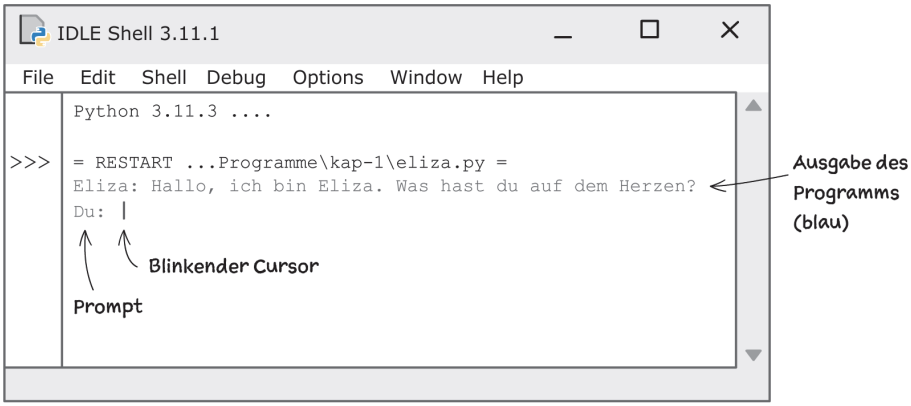
Programm:

```
# eliza.py
print('Eliza: Hallo, ich bin Eliza. Was hast du auf dem Herzen?')
eingabe = input('Du: ')
if 'hass' in eingabe:
    print('Eliza: Hass kann Wertvolles zerstören.')
else:
    print('Eliza: Kannst du mir das Problem näher erklären?')
```

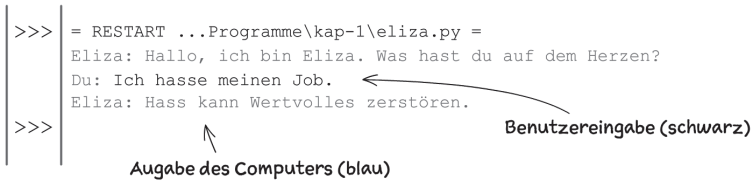
Starte das Programm vom Editor aus. Klicke zuerst in der Menüleiste auf RUN und dann auf den Befehl RUN MODULE.



Es öffnet sich ein Shell-Fenster. Oben stehen einige Informationen zu der Python-Version, die du verwendest. Hinter dem Prompt >>> erscheint eine Meldung, die mit `= RESTART` beginnt. Sie zeigt dir an, dass dein Programm ausgeführt wird. Darunter steht in blauer Schrift die Ausgabe des Programms.



Die zweite Zeile der Ausgabe besteht aus dem `Text Du:` . Hinter dem Doppelpunkt ist ein Leerzeichen und dahinter blinkt der Cursor. Der Computer wartet nun darauf, dass du etwas über die Tastatur eingibst. Den Text vor dem Cursor, also `Du:` (einschließlich Leerzeichen), nennt man *Prompt*. Ein Prompt signalisiert dem Benutzer: »Ich warte auf eine Eingabe.«. Sobald du die Taste `Enter` drückst, wird die Eingabe vom Computer übernommen und das Programm läuft weiter.



So funktioniert es:

Ein Python-Programm ist ein Text, der vom Python-Interpreter ausgeführt werden kann. Ein Programm besteht aus Anweisungen. Gehen wir nun den Programmtext Zeile für Zeile durch.

```
# eliza.py
```

Die erste Zeile ist keine Anweisung. Sie ist ein *Kommentar*, der für Menschen bestimmt ist und vom Python-Interpreter ignoriert wird. Ein Kommentar beginnt mit einem Hashtag # und kann so vom Python-Interpreter erkannt werden. Kommentare sollen die Lesbarkeit eines Programms verbessern. Oft wird eine Programmpassage stichwortartig erklärt. In diesem Fall wird der Name der Programmdatei angegeben. Alle Programmbeispiele in diesem Buch beginnen mit einem solchen Kommentar. So kannst du die Programme in den Ordnern des Downloadmaterials leicht wiederfinden.

```
print('Eliza: Hallo, ich bin Eliza. Was hast du auf dem Herzen?')
```

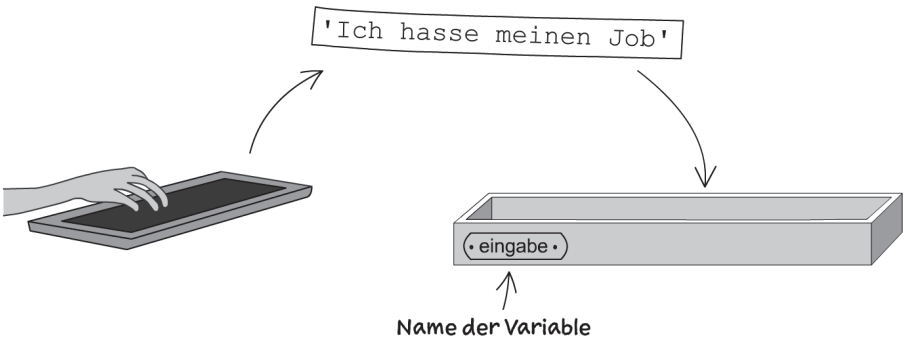
Diese Anweisung bewirkt, dass auf dem Bildschirm der Begrüßungstext ausgegeben wird. Die Anweisung ist ein Aufruf der Funktion `print()`. Dieser Aufruf ist so aufgebaut: Zuerst kommt der Name der Funktion, dahinter dann in Klammern der String `'Eliza: Hallo, ich bin Eliza.'`. Ein *String* ist eine Folge von beliebigen Zeichen, die durch Anführungszeichen eingerahmt sind.

```
eingabe = input('Du: ')
```

Hier wird die Funktion `input()` aufgerufen. Der Wert, der bei einem Funktionsaufruf in Klammern hinter dem Namen einer Funktion steht, heißt *Argument* der Funktion. Hier ist das Argument der String `'Du: '`.

Der Aufruf der Funktion bewirkt Folgendes: Auf dem Bildschirm wird der Prompt `Du:` ausgegeben – ohne die Anführungszeichen. Dann wartet der Computer. Der Benutzer des Programms kann nun über die Tastatur Zeichen eingeben. Sobald die Taste Enter gedrückt worden ist, wird ein String, der aus den eingegebenen Zeichen zusammengesetzt wird, der Variablen `eingabe` zugewiesen. Eine *Varia-*

ble kann man sich als Speicher für Daten vorstellen. Das Bild veranschaulicht an einem Beispiel, was bei der Zuweisung passiert.

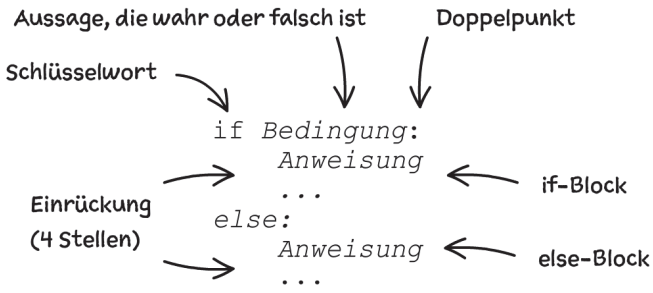


In dem Bild wird die Variable als Behälter dargestellt, der einen String als Inhalt aufnimmt. Der Clou ist: Über den Namen der Variablen kann man auf ihren Inhalt zugreifen. Das machen wir gleich in der nächsten Anweisung.

```
if 'hass' in eingabe:
    print('Eliza: Hass kann Wertvolles zerstören.')
else:
    print('Eliza: Kannst du mir das Problem näher erklären?')
```

Hier haben wir eine zusammengesetzte Anweisung, die über vier Zeilen geht. Es handelt sich um eine *if-else-Anweisung* (wenn-dann-sonst-Anweisung). Die Abbildung zeigt, wie eine if-else-Anweisung grundsätzlich aufgebaut ist. Die wichtigsten »Bauteile« sind

- eine *Bedingung*, d.h. eine Aussage, die wahr oder falsch sein kann,
- der *if-Block*, das sind Anweisungen, die ausgeführt werden, wenn die Bedingung erfüllt ist, also wahr ist,
- der *else-Block*, das sind Anweisungen, die ausgeführt werden, wenn die Bedingung nicht erfüllt ist, die Aussage also falsch ist.

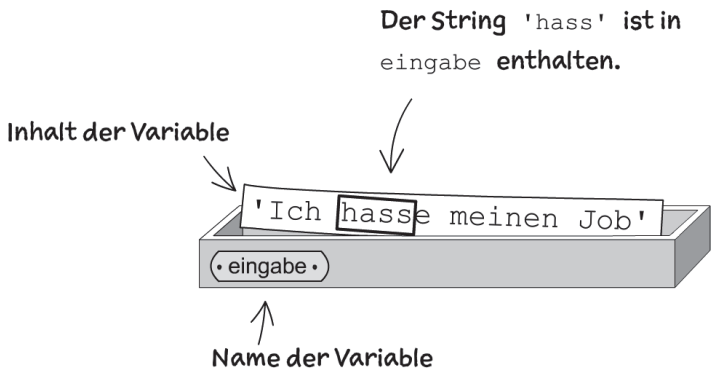


Beachte, dass if-Block und else-Block etwas eingerückt sind, typischerweise um 4 Stellen. Die Schlüsselwörter `if` und `else` müssen genau untereinanderstehen.

In diesem Fall lautet die Bedingung:

```
'hass' in eingabe
```

Das folgende Bild veranschaulicht an einem Beispiel, was damit gemeint ist.



Wenn die Bedingung erfüllt ist, wird die Anweisung

```
print('Eliza: Hass kann Wertvolles zerstören.')
```

ausgeführt. Ansonsten wird

```
print('Eliza: Kannst du mir das Problem näher erklären?')
```

ausgeführt. Das Programm untersucht also die Eingabe und gibt einen passenden Text aus. Im Prinzip machen wir Menschen das auch, wenn wir uns mit jemandem unterhalten.

EVA-Prinzip

Computerprogramme sind oft nach dem EVA-Prinzip aufgebaut. Der Benutzer gibt z.B. über die Tastatur Daten ein (Zahlen, Texte). Der Computer verarbeitet die Daten und gibt ein Ergebnis in lesbarer Form auf dem Bildschirm aus.