

Python

Einführung in die objektorientierte Programmierung



Monty Python's Flying Circus

Python

- Objektorientierte Programmiersprache.
- Klare, einfache Syntax.
- Strukturierte Programmierung.
- Interpretative Sprache.
- Plattformunabhängig.
- Download unter <https://www.python.org/>

Geschichte

- Anfang 1990er Jahre von Guido von Rossum am Centrum voor Wiskunde en Informatica entwickelt.
- Benennung nach der britischen Comedy-Gruppe Monty Python.

Versionszweig 2.x

- Python 2.7. ist in diesem Zweig die letzte Version.
- Lebensende 2020. Nur noch Bugfixes.
- Nutzung: Projekte, die in einer alten Version geschrieben sind. Verwendung von Paketen, die nicht mit Version 3.x kompatibel sind.

Versionszweig 3.x

- Beginn mit Python 3.0 ab dem 3. Dezember 2008. Aktuelle Version 3.6.
- Keine Kompatibilität zu Versionszweig 2.x
- Informationen zur Portierung von Version 2.x nach 3.x:
<https://docs.python.org/3/howto/pyporting.html>

Einsatzmöglichkeiten

- Web-Programmierung (CGI, Django, TurboGears, Zope...).
- Textverarbeitung.
- GUIs / Dialogfenster entwickeln (Tkinter, ...).
- Prototypen-Erstellung für Machbarkeitsstudien.
- Nutzung als eingebettete Skriptsprache.
- Siehe <http://www.python.org/about/success/>.

Literatur

- Michael Weigend: Python 3. Lernen und professionell anwenden. mitp
- Johannes Ernesti & Peter Kaiser: Python 3: Das umfassende Handbuch. Rheinwerk
- Bernd Klein: Einführung in Python 3. Hanser
- Arnold Willemer: Python. Der Sprachkurs für Einsteiger und Individualisten.
- Luigi Lo Iacono & Stephan Wiefling & Michael Schneider: Programmieren trainieren

Hinweise zu Tutorials etc. im Web

- <https://wiki.python.org/moin/BeginnersGuide/Programmers>
- <http://openbookproject.net/thinkcs/python/english3e/>

Deutschsprachige Tutorials im Web

- <http://www.python-kurs.eu/kurs.php>
- <https://py-tutorial-de.readthedocs.io/de/python-3.3/>
- <https://media.readthedocs.org/pdf/py-tutorial-de/python-3.3/py-tutorial-de.pdf>

Style Guides für Python-Code

- <https://www.python.org/dev/peps/pep-0008/>
- <http://docs.python-guide.org/en/latest/writing/style/>
- <http://www.cs.bu.edu/courses/cs108/guides/style.html>

Entwicklungsumgebungen

- Integrierte Entwicklungsumgebung in Python: IDLE.
- Weitere IDEs siehe <https://wiki.python.org/moin/%20IntegratedDevelopmentEnvironments>.
- Spyder (<https://pypi.org/project/spyder/>) wird häufig bei wissenschaftlichen mathematischen Berechnung genutzt.
- Portable Version für das Betriebssystem Windows: <https://winpython.github.io/>

Programmierung spielerisch lernen

- <http://www.swisseduc.ch/informatik/karatojava/pythonkara/>
- https://wiki.scratch.mit.edu/wiki/Scratch_Wiki_Home
- <https://turtleacademy.com/>

IDLE

- Integrated Development Enviroment.
- Einfache Entwicklungsumgebung für Python.
- Automatische Installation mit der Programmiersprache Python.
- Informationen zu der IDE:
<https://docs.python.org/3/library/idle.html>.

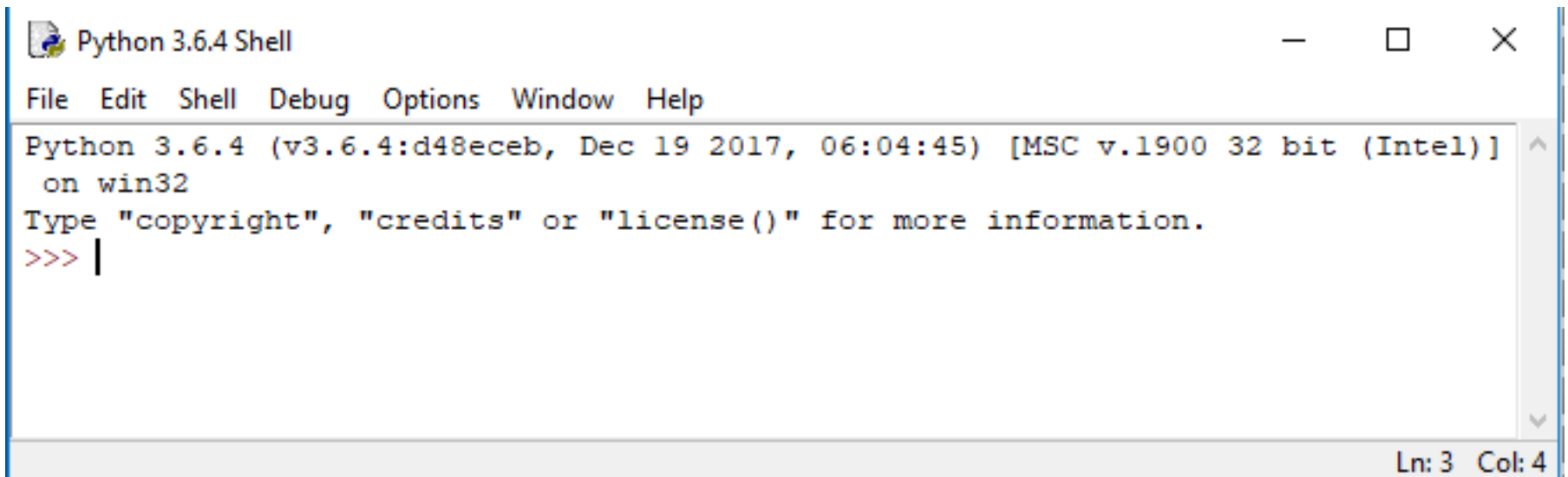
... unter Windows öffnen

- Icon auf dem Desktop.
- Windows 8 und höher: Suchen ...
- Doppelklick auf `...\Python\Python[version]\Lib\idlelib\idle.pyw` im Windows Explorer.

Python-Shell

- Kommandozeilenorientiert. Zeilenweise werden Anweisungen in einer Programmiersprache eingegeben.
- Jede Eingabezeile wird mit einem Prompt gekennzeichnet. In der Shell werden die Zeilen folgendermaßen gekennzeichnet: `>>>`.
- Befehle oder Anweisungen, die der Nutzer in die Shell eingibt, können so interpretiert werden, dass ein Computer sie ausführen kann.

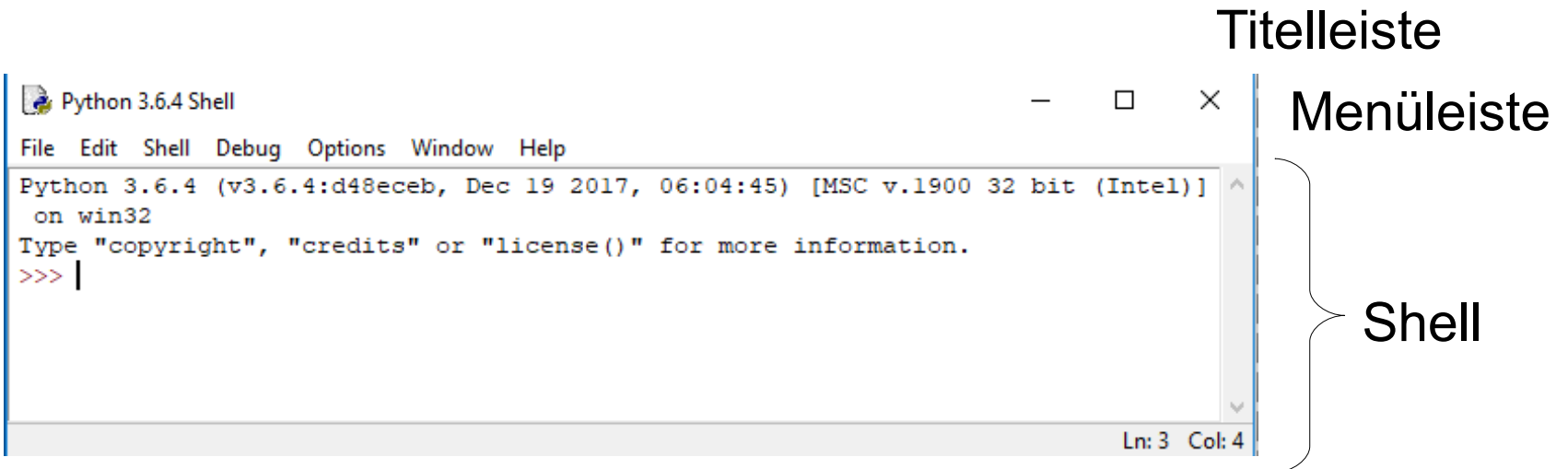
... nach dem Öffnen



```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
  on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 3 Col: 4

Aufbau



Titelleiste

- Das Systemmenü wird am linken Rand angezeigt.
- Die Schaltflächen zum Minimieren, Maximieren und Schließen von IDLE werden am rechten Rand angezeigt.
- Mittig wird die Python-Version eingeblendet.

Menüleiste

- *File* enthält alle Befehle zum Öffnen und Speichern von Dateien.
- *Edit* enthält Befehle zum Bearbeiten von Code.
- *Shell* startet die Arbeitsfläche neu.
- *Debug* zur Fehlersuche im Programm.
- *Options* zur Konfiguration von IDLE.
- *Window*.
- *Help*.

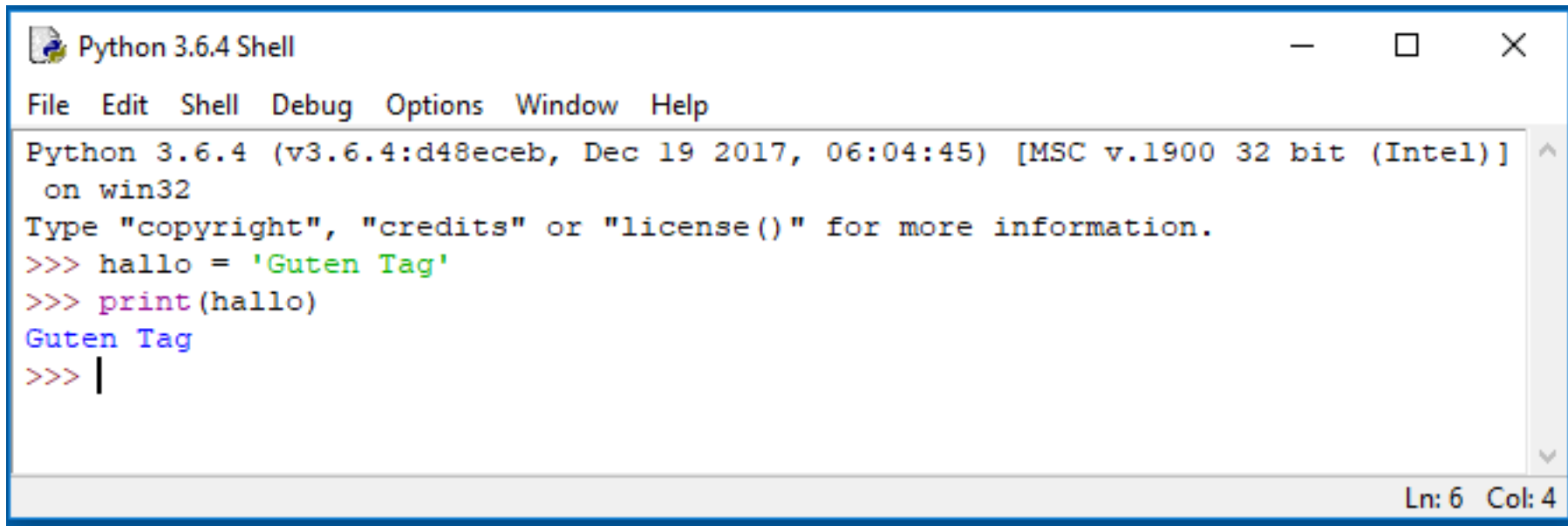
Informationen in der Shell

- Anzeige der aktuell installierten Python-Version.
- Informationen in Abhängigkeit des Betriebssystems.

Prompt

- Am Beginn einer Zeile `>>>`.
- Wenn die Einfügemarke direkt im Anschluss daran blinkt, kann dort ein Befehl eingegeben werden.

Eingabe der Anweisung



The image shows a screenshot of a Python 3.6.4 Shell window. The window title is "Python 3.6.4 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following content:

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> hallo = 'Guten Tag'  
>>> print(hallo)  
Guten Tag  
>>> |
```

The status bar at the bottom right of the window shows "Ln: 6 Col: 4".

... in der Shell

```
>>> print('Guten Tag')
Guten Tag
>>>
```

- An der Position der Einfügemarke (hinter dem Prompt) wird die Anweisung `print('Guten Tag')` eingegeben.
- Die Anweisung wird mit der Eingabetaste (<Return>) abgeschlossen. Sobald die Anweisung abgeschlossen ist, kann diese nicht verändert werden.

Anweisungen

- Befehle (Kommandos) für den Computer in einer bestimmten Programmiersprache.
- Handlungsanweisungen werden mit Hilfe einer bestimmten Syntax beschrieben. Die Syntax wird mit Hilfe der gewählten Programmiersprache festgelegt.
- In der Regel wird eine Anweisung pro Zeile ausgeführt.

... in der Shell ausführen

- Nach Drücken der Eingabetaste wird die Anweisung entsprechend der Syntax interpretiert. Der Befehl wird ausgeführt.
- Anweisungen können Werte berechnen oder Informationen ausgeben. Der berechnete Wert oder Ausgabe werden in der nächsten Zeile in der Shell angezeigt.

Vorheriger und nächster Befehl in der Shell

- ALT+P zeigt den vorherigen Befehl in der History-Liste an.
- ALT+N zeigt den nächsten Befehl in der History-Liste an.
- Hinweis: Mit Hilfe von *Options – Configure IDLE*, Registerkarte *Keys* können die Tastenkombinationen angepasst werden.

Beendigung der Shell

```
>>> exit()
```

- Die Warnmeldung Kill? wird durch Klicken auf die Schaltfläche *OK* bestätigt.
- Das laufende Programm wird beendet. Die Shell wird geschlossen.

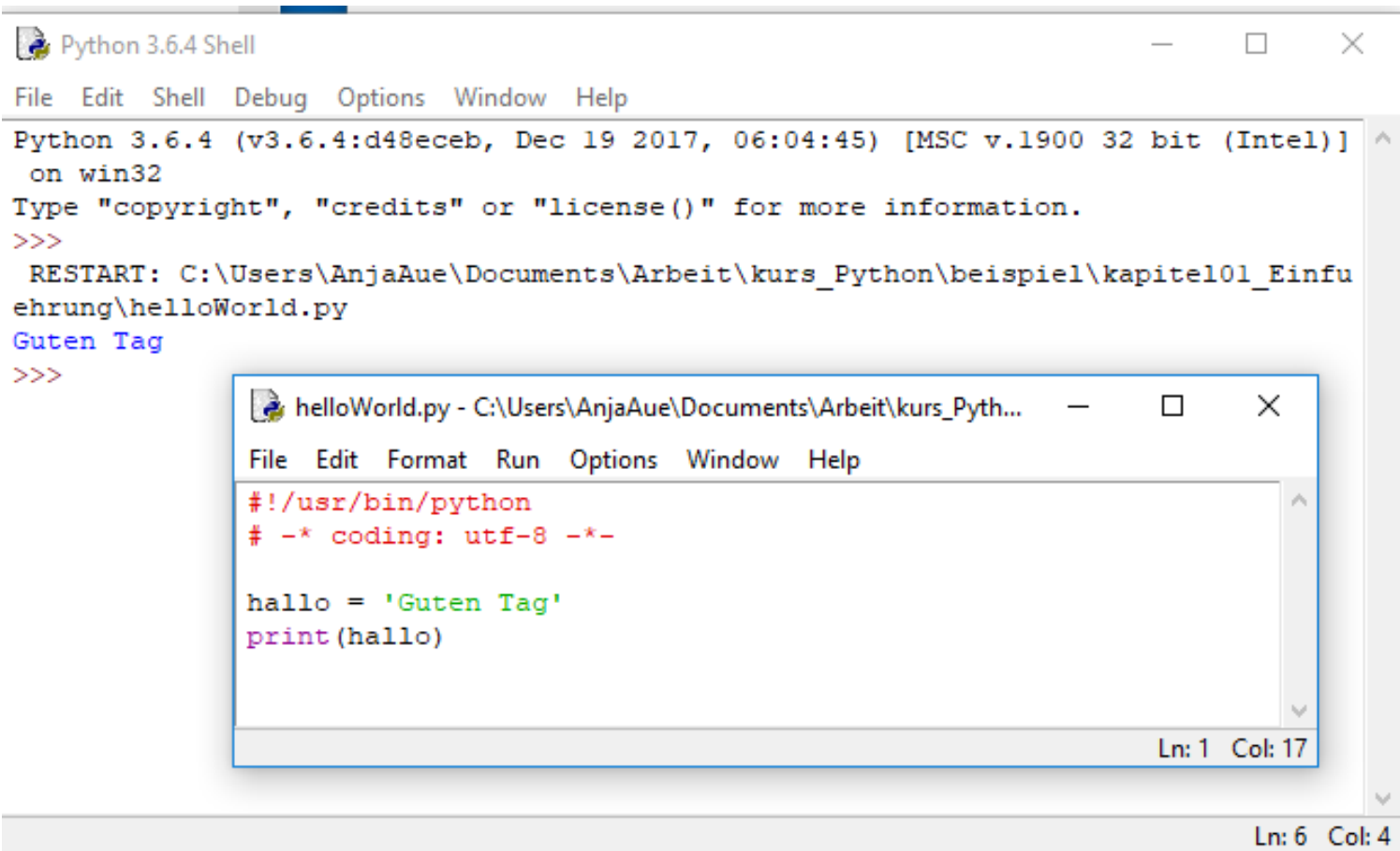
Schlüsselwörter

- Wörter mit einer besonderen Bedeutung in Abhängigkeit der Programmiersprache.
- Reservierte Wörter in einer Programmiersprache.
- Wörter, die in der Shell standardmäßig mit der Farbe „orange“ markiert werden.

... in Python

and	except	lambda	while
as	False	None	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	
def	if	raise	
del	import	return	
elif	in	True	
else	is	try	

Code-Dateien



The image shows a screenshot of a Python 3.6.4 Shell window. The shell window title is "Python 3.6.4 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell output shows the Python version and architecture, followed by a prompt. The user enters a command to restart a Python script: `RESTART: C:\Users\AnjaAue\Documents\Arbeit\kurs_Python\beispiel\kapitel01_Einfuehrung\helloWorld.py`. The output of the script is `Guten Tag`. A smaller window titled "helloWorld.py - C:\Users\AnjaAue\Documents\Arbeit\kurs_Pyth..." is overlaid on the shell window, showing the source code of the script. The code is as follows:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

hallo = 'Guten Tag'
print(hallo)
```

The status bar at the bottom of the shell window shows "Ln: 6 Col: 4". The status bar at the bottom of the code editor window shows "Ln: 1 Col: 17".

... in Python

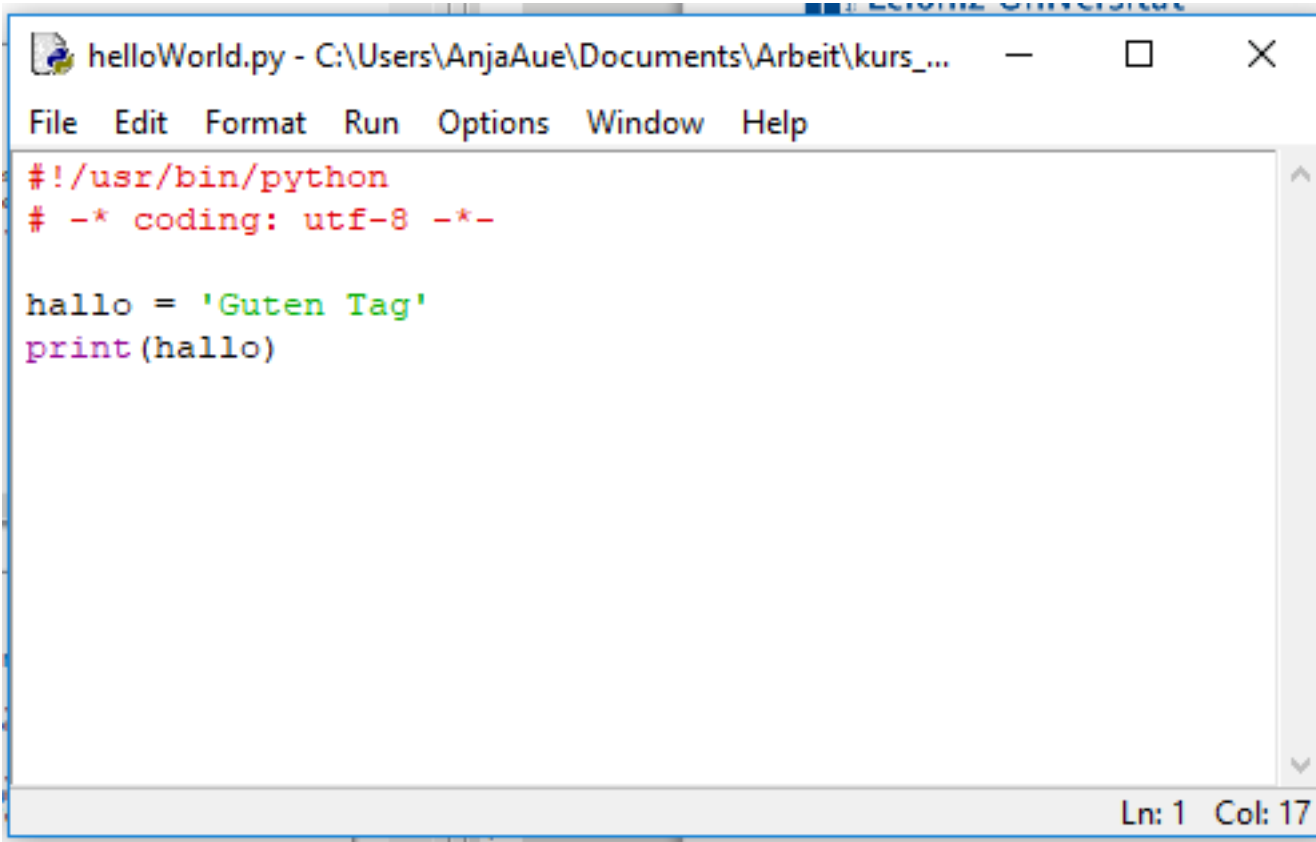
- Logische Zusammenfassung von Anweisungen in Modulen.
- Speicherung von Befehlen.
- Dateien mit der Endung „.py“ oder „.pyw“.

Dateiendung „.py“ oder „.pyw“

- Dateien mit der Endung „.py“ werden als Konsolenprogramme bezeichnet. Diese Art von Dateien können über die Shell ausgeführt werden.
- Dateien mit der Endung „.pyw“ werden bei der Programmierung einer grafischen Oberfläche genutzt. Die Programme öffnen ein Fenster, in dem das Programm läuft.

Aufbau des Editors

Titelleiste



Menüleiste

Codefenster

Titelleiste

- Das Systemmenü wird am linken Rand angezeigt.
- Die Schaltflächen zum Minimieren, Maximieren und Schließen werden am rechten Rand angezeigt.
- Mittig wird der Name und der Speicherort der geöffneten Datei eingeblendet. Falls die Datei neu ist, wird „untitled“ angezeigt.

Menüleiste

- *File*. Alle Befehle zum Öffnen und Speichern von Dateien.
- *Edit*. Befehle zum Bearbeiten von Code.
- *Format*. Codeformatierung mit Hilfe von Einrückungen.
- *Run*. Starten des Programms.
- *Options*. Konfiguration von IDLE.
- *Window*.
- *Help*.

Codefenster

- Pro Zeile wird eine Anweisung angegeben.
- Beschreibung eine Aufgabe mit Hilfe von Python-Befehlen.
- Die Befehle in dem Codefenster werden in einer Datei gespeichert.

... neu anlegen und speichern

- *File – New File*. Der Texteditor zur Eingabe des Programms öffnet sich.
- *File – Save As* speichert den eingegebenen Code unter einem Dateinamen an einem bestimmten Ort auf dem Rechner.
- *File – Close* schließt die Datei.

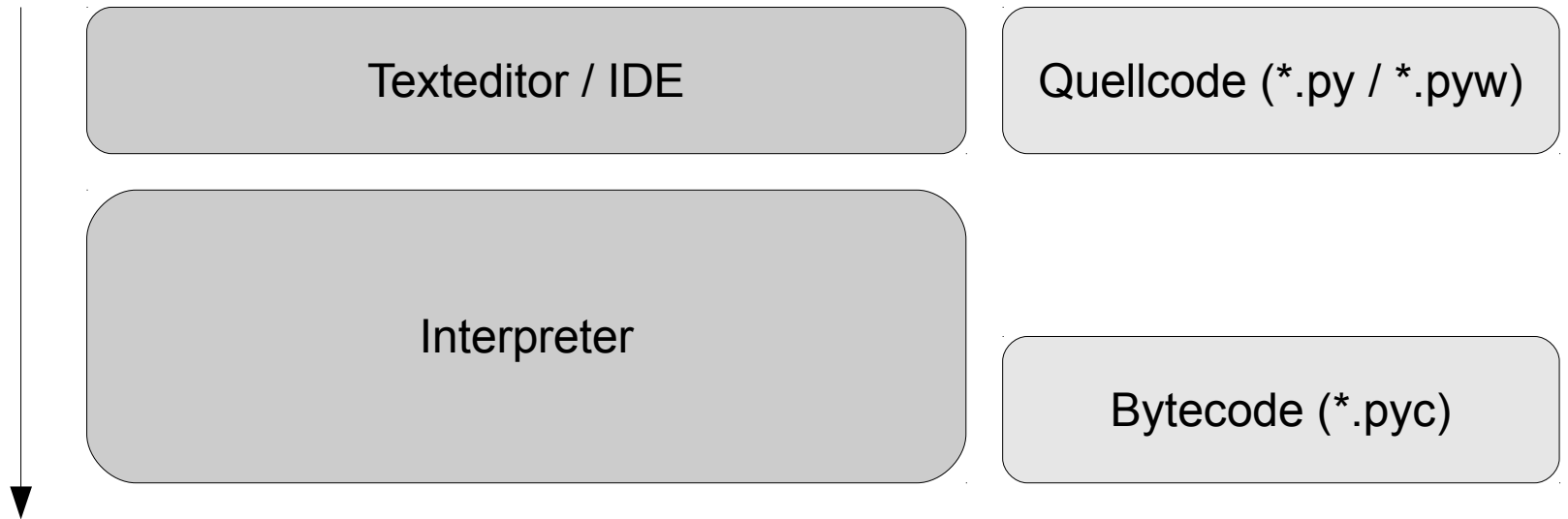
... öffnen und Änderungen speichern

- *File – Open*. Mit Hilfe des Dialogs *Öffnen* wird die gewünschte Datei ausgewählt.
- *File – Save* speichert die Änderungen in der geöffneten Datei.
- *File – Close* schließt die Datei.

Starten des Codes

- *Run – Run Module* oder Taste <F5>.
- Die Datei wird zeilenweise von oben nach unten durchlaufen.
- Die Anweisungen werden interpretiert und ausgeführt.

Architektur



Interpreter

- Der Code wird direkt ausgeführt.
- Zuerst wird der von Menschen lesbare Code automatisiert in Byte-Code umgewandelt. Dieser von der Maschine lesbare Code wird direkt interpretiert.

Angabe zum Interpreter in der Code-Datei

```
#!/usr/bin/python  
  
hallo = 'Guten Tag'  
print(hallo)
```

- In der ersten Zeile wird der Pfad zum Interpreter angegeben.
- Die Zeile beginnt mit `#!`. Die Zeile wird als magic line oder Shebang-Zeile bezeichnet.
- Den beiden Zeichen folgt der Pfad zum Python-Interpreter.

Hinweise

```
#!/usr/bin/python
```

```
hallo = 'Guten Tag'  
print(hallo)
```

- Betriebssystem Windows: Die Angaben sind nicht erforderlich.
- Betriebssystem Unix / Linux: Jede Code-Datei muss mit dieser Zeile beginnen.

Angabe der Python-Version

```
#!/usr/bin/env python3
```

```
hallo = 'Guten Tag'  
print(hallo)
```

- In der ersten Zeile wird der Pfad zum Interpreter angegeben.
- Die Zeile beginnt mit `#!`.
- Das Tool `env` sucht den Pfad entsprechend der `PATH`-Angaben.
- In diesem Beispiel wird nach einer Python-Version ab 3.0 gesucht.

Anzeige der von IDLE genutzten Python-Version

- Menü des Texteditors: *Help – About IDLE*.
- Zuerst wird die verwendete Python-Version angezeigt.
- Darunter wird die verwendete IDLE-Version angezeigt.

Zeichenkodierung der Datei

```
#!/usr/bin/env python3
# coding=utf-8

hallo = 'Guten Tag'
print(hallo)
```

- In der zweiten Zeile kann eine Zeichenkodierung für die Datei angegeben werden. Die Datei muss in dieser Zeichenkodierung gespeichert werden.
- Dem Hash-Zeichen folgt der Begriff `coding`. Dem Begriff wird mit Hilfe des Gleichheitszeichens eine Zeichenkodierung zugewiesen.

Unicode-Zeichensatz

- Das erste Zeichen im Zeichensatz wird mit Hilfe von `'\u0000'` angegeben.
- Die ersten 127 Zeichen des UTF-8-Zeichensatzes sind mit dem ASCII-Zeichensatz identisch.
- Die ersten 256 Zeichen des UTF-8-Zeichensatzes entsprechen dem ISO 8859-1 (Latin 1)-Zeichensatz.
- Standardcodierung ab Python 3.x.
- Siehe <https://unicode-table.com/de/>.

ASCII-Zeichensatz

- American Standard Code for Information Interchange.
- Definition von 128 Zeichen.
- Standardcodierung für die Versionen Python 2.x.
- Siehe <http://www.torsten-horn.de/techdocs/ascii.htm>

Prozedurale Programmierung

```
PI = 3.1415926
```

```
def startKreis(radius, posX = 0, posY = 0):  
    print("Fläche: " + str(getFlaeche(radius)))  
    print("Umfang: " + str(getUmfang(radius)))  
    print("\n")
```

```
def getFlaeche(radius):  
    flaeche = PI * (radius * radius)  
    return flaeche
```

```
def getUmfang(radius):  
    umfang = 2 * PI * radius  
    return umfang
```

Erläuterung

- Strukturierung von Code.
- Funktionen fassen eine sequentielle Abfolge von Befehlen zusammen. Der Code ist wiederverwendbar.
- Daten und Funktionen werden getrennt.

Objektorientierte Programmierung

```

class ClsKreis(object):
    pi = 3.1415926

    def __new__(cls, radius = 1):
        print ("ClsKreis wird erzeugt.")
        instance = super(ClsKreis, cls).__new__(cls)
        return instance

    def __init__(self, radius = 1):
        print (' und wird initialisiert.')
        self.radius = radius

    def __del__(self):
        print (' und stirbt.')

    def setRadius(self, radius):
        self.radius = radius

    def getFlaeche(self):
        flaeche = ClsKreis.pi * (self.radius * self.radius)
        return flaeche
  
```

Erläuterung

- Python: Everything is a object
- Abstraktion von konkreten Objekten aus der realen Welt.
- Definition von Klassen als Vorlage für Objekte.

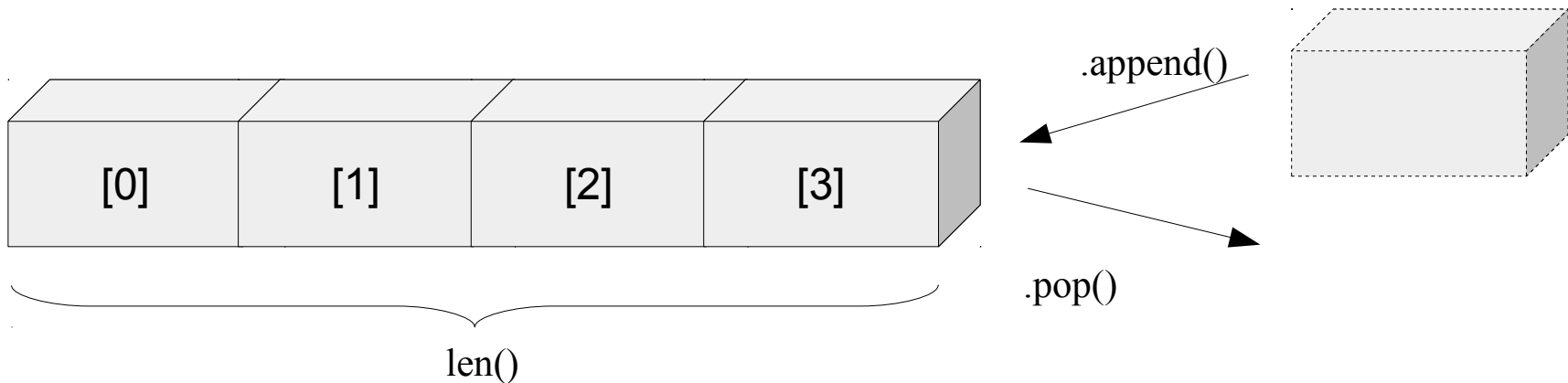
Paradigmen

- Datenkapselung. Die Daten (Attribute) eines Objekts werden nur durch Methoden verändert. Die Daten können nicht unkontrolliert von außen verändert werden.
- Vererbung. Attribute und Methode können von Eltern-Objekten an deren Kind-Objekte vererbt werden. Die Kind-Objekte können diese wiederum verändern oder überschreiben.
- Polymorphie. Austauschbarkeit von Objekten. Bezeichner können verschiedene Datentypen annehmen.

Klasse

- Allgemeine Beschreibung von einem Objekt. Welche Eigenschaften hat ein Objekt? Wie verhält sich das Objekt?
- Bauplan für ein bestimmtes Ding zum Beispiel für eine Liste, ein Wörterbuch etc.
- Vorlage für die Erzeugung eines Objektes aus der Standard-Bibliothek. Benutzerdefinierte Vorlagen.

Klasse „Liste“



- Die Klasse `<class 'list'>` beschreibt allgemein die Funktionalität einer Liste.
- Eine Liste aus Python besteht aus beliebig vielen Elementen. Die Elemente können von einem beliebigen Typ sein. Während der Programmausführung kann eine Liste verändert werden.

Listen

- Beispiele: Einkaufszettel, Temperaturwerte etc.
- Sequenzen, deren Elemente mit Hilfe eines Index angesprochen werden können.
- Eine Behälter, der x Elemente unterschiedlichster Arten enthält oder leer sein kann.
- Mit Hilfe des Befehls `help(list)` werden in der Shell Informationen zu einer Liste angezeigt.

Erstellung eines Objekts

```
farbe = list()
```

- Die Klasse `<class 'list'>` beschreibt allgemein die Eigenschaften und das Verhalten einer Liste.
- In Abhängigkeit des Bauplans „Liste“ wird in diesem Beispiel die leere Liste `farbe` mit Hilfe der Funktion `list()` erzeugt.
- Die Variable `farbe` ist eine Instanz, die auf den Speicherort der erzeugten Liste verweist.

Benutzerdefinierte Klassen

```
class ClsKreis(object):  
    pass
```

- Die Klasse `<<class '__main__.ClsKreis'>` beschreibt die benötigten Funktionalitäten eines Kreises.
- Die Klasse wird in einem Modul (eine Datei mit der Endung „.py“) gespeichert.
- Pro Modul wird eine Klasse definiert.

... nutzen

```
class ClsKreis(object):  
    pass
```

```
>>> myKreis = ClsKreis()
```

- *Run – Run Module* startet das Modul.
- An der Einfügemarke in der Shell wird folgende Anweisung eingegeben: `myKreis = ClsKreis()`.
- Von dem Bauplan „Kreis“ wird das konkrete Objekt `myKreis` erzeugt. Eine Instanz von der Klasse `ClsKreis` wird erzeugt.

Hinweise

- Der Name des Bauplans entspricht dem Name der Klasse.
- Die Groß- und Kleinschreibung der Bezeichner wird beachtet.
- Durch die leeren Klammern direkt im Anschluss an den Namen der Klasse wird eine Instanz mit den Standardeinstellungen erzeugt.

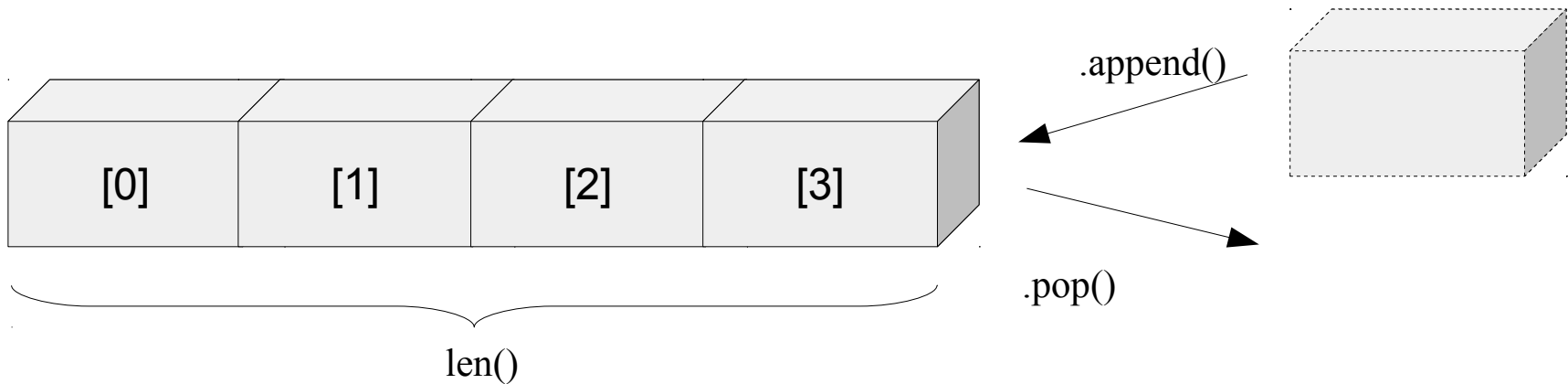
Instanz (Objekt)

- Ein Ding / Exemplar aus der realen Welt..
- Zur Laufzeit wird eine Instanz von einer Klasse erzeugt.
- Kategorisierung von Dingen. Die Dinge von einer Klasse haben alle die gleiche Eigenschaften, aber unterschiedliche Eigenschaften-Werte.
- Die Eigenschaften einer Instanz werden mit Hilfe von Methoden verändert.

Methoden

- Alle Methoden gemeinsam beschreiben die Funktionalität eines bestimmten Dings.
- Methoden sind in der Vorlage (Klasse), auf welche das Objekt basiert, gekapselt.
- Methoden sind Funktionen, die an ein Objekt gebunden sind.
- Lesen und modifizieren in Abhängigkeit von bestimmten Regeln der Attribut-Werte einer Instanz.
- Methoden werden für alle Objekte einer Klasse gemeinsam im Speicher abgelegt.

Methoden einer Liste



```
farbe.append("yellow")  
farbe.pop()
```

Methoden einer Liste

```
farbe.append("yellow")  
farbe.pop()
```

- Auf einer Instanz wird eine Methode angewendet. Methode und Instanz werden mit Hilfe des Punkt-Operators verbunden.
- Die Instanz ist von der Klasse ... In dieser Klasse ist die gewünschte Methode definiert.
- Die Methode `.append` in diesem Beispiel fügt das Element `yellow` an die Liste an. Die Methode `.pop` entfernt das letzte Element in der Liste und gibt dieses zurück.
-

„Konstruktoren“ einer Klasse

```
def __init__(self, radius = 1):  
    self.radius = radius
```

- Spezielle Methoden zur Konstruktion eines Objekts.
- Eine Instanz wird mit Hilfe von `ClsKreis()` oder `list()` erzeugt. Die Methode `__new__` zur Erzeugung des Objekts wird aufgerufen. Anschließend wird die Methode `__init__` für die Initialisierung aufgerufen.
- Beide Methoden können, müssen aber nicht, vom Programmierer überschrieben werden.

Attribute (Member, Instanzvariablen)

- Beschreibung eines Gegenstandes, einer Person, etc.
- Allgemeingültige Beschreibung für einen bestimmten Objekttyp.
- Jedes Objekt einer Klasse hat die gleichen Attribute.
- Jedes Objekt einer Klasse unterscheidet sich aber in mindestens einem Attribut-Wert von allen anderen Objekten.
- Für jedes Objekt werden die Attribute im Speicher abgelegt.

Beispiel: Attribute für ein Kreis

```
def __init__(self, radius = 1):  
    self.radius = radius
```

- In der Klasse `ClsKreis` ist das Attribut „Radius“ definiert.
- Der Attribut-Wert kann nur mit Hilfe von Methoden verändert werden.
- In der Klasse wird mit Hilfe von `self` auf das Objekt verwiesen, welches die Methode aufgerufen hat. Bei der Neuanlage einer Instanz wird auf das neu zu erstellende Objekt verwiesen.
- Der Platzhalter `self` und das Attribut werden mit Hilfe des Punktes verbunden.

Attribute und Methoden

- Attribute werden durch die Initialisierungsmethode `__init__` angelegt und initialisiert.
- Methoden wie zum Beispiel `.getFlaeche()` nutzen Attribute in Ausdrücken.
- Methoden wie zum Beispiel `.setRadius()` verändern die Werte von Attribute.

Lebenszyklus eines Objekts

... erzeugen
und initialisieren:

```
kreis = ClsKreis()
```

... arbeiten:
Methoden aufrufen

```
kreis.setRadius(4)  
kreis.getFlaeche()
```

... zerstören:

```
del kreis
```