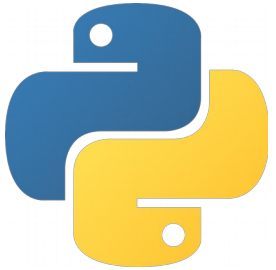
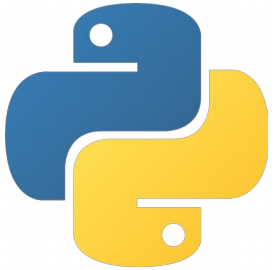
Python Betriebssystem



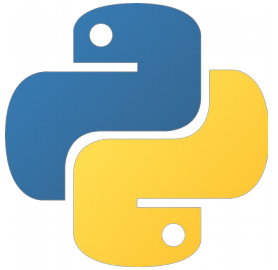
- Zugriff auf das Dateisystem – shutil
  - Verzeichnis- und Dateioperationen
  - copyfileobj(fsrc, fdst[, length])
    - Diese Operation kopiert den Inhalt des zum Lesen geöffneten Dateiobjekts fsrc in das zum Schreiben geöffnete fdst-Objekt
    - Mit dem optionalen Parameter length können Sie dabei die zu verwendende Zwischenspeichergröße in Bytes angeben
    - Ist length positiv, wird die fsrc portionsweise ausgelesen und nach fdst geschrieben, während bei negativem length zuerst der gesamte Inhalt von fsrc in den Speicher gelesen und dann in einem Rutsch nach fdst geschrieben wird
    - Standardmäßig wird ein positiver Wert für length verwendet, den das System wählt

Python Betriebssystem



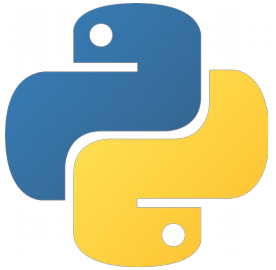
- Zugriff auf das Dateisystem – shutil
  - Verzeichnis- und Dateioperationen
  - copy(src, dst)
    - Diese Operation kopiert die Datei unter dem Pfad src nach dst
    - Der Parameter dst kann dabei einen Pfad zu einer Datei enthalten, die dann erzeugt oder überschrieben wird
    - Verweist dst auf einen Ordner, wird eine neue Datei mit dem Dateinamen von src im Ordner dst erzeugt oder gegebenenfalls überschrieben
    - Dies ist der wesentliche Unterschied zu der Funktion copyfile, die keinen Ordner als Ziel akzeptiert

Python Betriebssystem



- Zugriff auf das Dateisystem – shutil
  - Verzeichnis- und Dateioperationen
  - rmtree(src[, ignore_errors[, onerror]])
    - Hiermit wird die gesamte Verzeichnisstruktur unter src gelöscht
    - Für ignore_errors kann ein Wahrheitswert übergeben werden, der bestimmt, ob beim Löschen auftretende Fehler ignoriert oder von der Funktion, die für onerror übergeben wurde, behandelt werden sollen
    - Wird ignore_errors nicht angegeben, ruft jeder auftretende Fehler eine Exception hervor

Python Betriebssystem

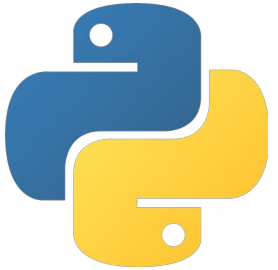


- Zugriff auf das Dateisystem – shutil
  - Verzeichnis- und Dateioperationen
  - rmtree(src[, ignore_errors[, onerror]])
    - Wenn Sie onerror angeben, muss es eine Funktion sein, die drei Parameter erwartet

    1. function – eine Referenz auf die Funktion, die den Fehler verursacht hat. Dies können os.listdir, os.remove oder os.rmdir sein
    2. path – der Pfad, für den der Fehler auftrat
    3. excinfo – der Rückgabewert von sys.exc_info im Kontext des Fehlers

    - Hinweis
      - Exceptions, die von der Funktion onerror geworfen werden, werden nicht abgefangen

Python Betriebssystem

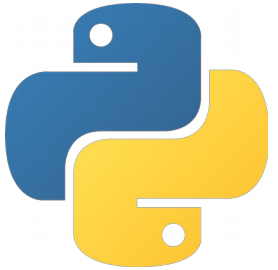


- Zugriff auf das Dateisystem – shutil
  - Archivoperationen

  - Für die folgenden Tests gehen wir davon aus, dass sich im aktuellen Arbeitsverzeichnis ein Verzeichnis namens daten befindet, wie folgt aufgebaut

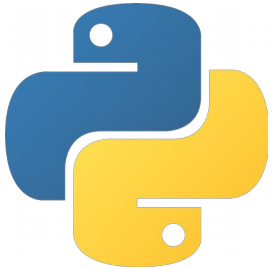
```
daten
    unterordner1
        datei4.txt
        datei5.txt
    unterordner2
        unterordner3
            datei7.txt
    datei1.txt
    datei2.txt
    datei3.txt
```

Python Betriebssystem



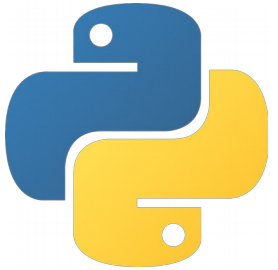
- Zugriff auf das Dateisystem – shutil
  - Archivoperationen
  - make_archive(base_name, format[, root_dir[, base_dir[, verbose[, dry_run[, owner[, group[, logger]]]]]]])
    - Diese Funktion erzeugt ein neues Archiv, das die Dateien im Verzeichnis root_dir enthält
    - Wird root_dir nicht angegeben, werden die Dateien des aktuellen Arbeitsverzeichnisses gepackt
    - Mit dem Parameter base_name werden der Speicherort und der Name der Archivdatei festgelegt, wobei die Dateiendung nicht mitangegeben werden sollte

Python Betriebssystem



- Zugriff auf das Dateisystem – shutil
  - Archivoperationen
  - make_archive(base_name, format[, root_dir[, base_dir[, verbose[, dry_run[, owner[, group[, logger]]]]]]])
    - Über den Parameter format wird das gewünschte Format der Archivdatei angegeben
    - Die verfügbaren Archivformate können Sie mit der Funktion get_archive_formats ermitteln
    - Typischer Aufruf
      >>> shutil.make_archive("test", "zip", "daten")
      '/home/user/test.zip'

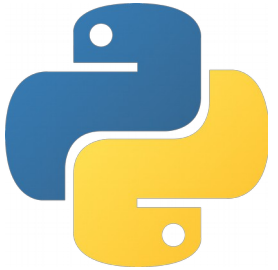
Python Betriebssystem



- Zugriff auf das Dateisystem – shutil
  - Archivoperationen
  - make_archive(base_name, format[, root_dir[, base_dir[, verbose[, dry_run[, owner[, group[, logger]]]]]]])
    - Möchte man nur die Dateien eines Unterverzeichnisses von root_dir inklusive des zugehörigen relativen Pfads packen, kann man dies mit dem Parameter base_dir erreichen
    - Beispiel nur die Dateien im Unterverzeichnis unterordner2 packen, wobei jedoch der relative Pfad innerhalb vom Verzeichnis daten erhalten bleibt

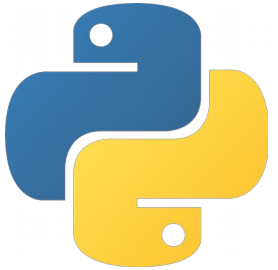
      >>> shutil.make_archive("test2", "zip", "daten", "unterordner2")
      '/home/user/test2.zip'

Python Betriebssystem



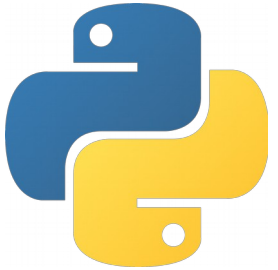
- Zugriff auf das Dateisystem – shutil
  - Archivoperationen
  - get_archive_formats()
    - Diese Funktion gibt eine Liste von zweielementigen Tupeln zurück, in denen die verfügbaren Formate zum Erstellen von Archiven beschrieben werden
    - Beispiel
      >>> shutil.get_archive_formats()
      [('bztar', "bzip2'ed tar-file"), ('gztar', "gzip'ed tar-file"), ('tar', 'uncompressed tar file'), ('zip', 'ZIP file')]
    - Jedes Tupel in dieser Liste enthält zwei Strings: den Namen des Formats und eine kurze Beschreibung

Python Betriebssystem



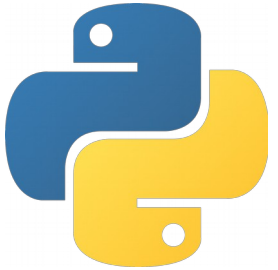
- Zugriff auf das Dateisystem – shutil
  - Archivoperationen
  - get_unpack_formats()
    - Diese Funktion arbeitet wie get_archive_formats , nur werden hier die verfügbaren Formate zum Entpacken von Archiven aufgelistet
    - Beispiel
      >>> shutil.get_unpack_formats()
      [('bztar', ['.bz2'], "bzip2'ed tar-file"), ('gztar', ['.tar.gz', '.tgz'], "gzip'ed tar-file"), ('tar', ['.tar'], 'uncompressed tar file'), ('zip', ['.zip'], 'ZIP file')]

Python Betriebssystem



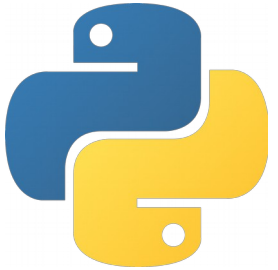
- Zugriff auf das Dateisystem – shutil
  - Archivoperationen
  - unpack_archive(filename[, extract_dir[, format]])
    - Diese Funktion entpackt das Archiv unter dem Pfad filename in das Zielverzeichnis extract_dir
    - Wird extract_dir nicht angegeben, werden die Daten in das aktuelle Arbeitsverzeichnis entpackt
    - Mit dem Parameter format kann das Format des Archivs angegeben werden
    - Falls kein Wert für format übergeben wurde, versucht unpack_archive, das Format des Archivs anhand der Dateiendung zu ermitteln

Python Betriebssystem



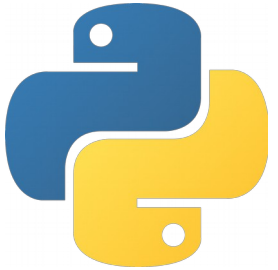
- Zugriff auf die Laufzeitumgebung – sys
  - Das Modul sys der Standardbibliothek stellt vordefinierte Variablen und Funktionen zur Verfügung, die sich auf den Python-Interpreter selbst beziehen oder eng mit diesem zusammenhängen
  - So können Sie über das Modul sys beispielsweise die Versionsnummer des Interpreters oder des Betriebssystems abfragen
  - Das Modul stellt dem Programmierer eine Reihe von Informationen zur Verfügung, die mitunter sehr nützlich sein können
  - Es lohnt sich also, sich einen Überblick über die Funktionalität von sys zu verschaffen, allein schon, um einen Begriff davon zu bekommen, an welche Informationen Sie durch dieses Modul gelangen können

Python Betriebssystem



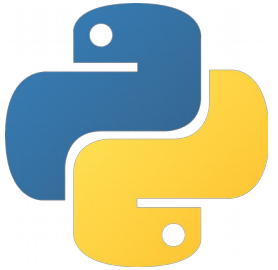
- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - argv
    - Die Liste argv enthält die Kommandozeilenparameter, mit denen das Python-Programm aufgerufen wurde
    - argv[0] ist der Name des Programms selbst
      - Im interaktiven Modus ist argv leer
    - Bei dem Programmaufruf
      programm.py -bla 0 -blubb abc
    - referenziert argv die folgende Liste
      ['programm.py', '-bla', '0', '-blubb', 'abc']

Python Betriebssystem



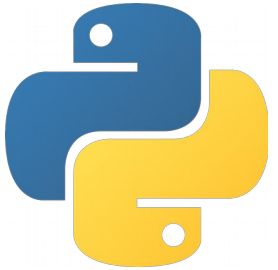
- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - executable
    - Dies ist ein String, der den vollen Pfad zur ausführbaren Datei des Python-Interpreters angibt
    - Beispiel
      >>> sys.executable
      '/usr/bin/python3'

Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - hexversion
    - Diese Konstante enthält die Versionsnummer des Python-Interpreters als ganze Zahl
    - Wenn sie durch Aufruf der Built-in Function hex als Hexadezimalzahl geschrieben wird, wird der Aufbau der Zahl deutlich
    - mit den Operatoren < und > können Sie testen, verwendete Version des Interpreters aktueller ist als eine bestimmte

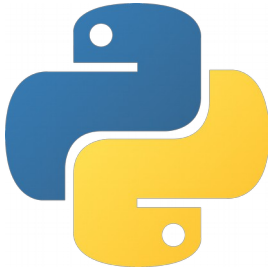
Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - hexversion
    - Beispiel
      >>> hex(sys.hexversion) ← Python Version 3.2.2
      '0x30202f0'

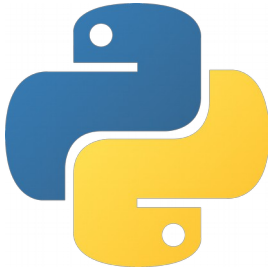
      >>> hex(sys.hexversion) ← Python Version 3.4.2
      '0x30402f0'

Python Betriebssystem



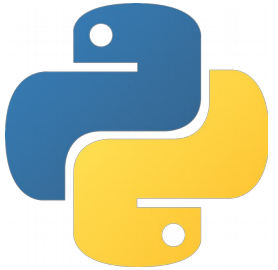
- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - path
    - Die Liste path enthält eine Reihe von Pfadangaben, die beim Einbinden eines Moduls der Reihe nach vom Interpreter durchsucht werden
    - Das zuerst gefundene Modul mit dem gesuchten Namen wird eingebunden
    - Es steht dem Programmierer frei, die Liste so zu modifizieren, dass das Einbinden eines Moduls nach seinen Wünschen erfolgt

Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - path
    - Beispiel
      >>> sys.path
      ['', 'C:\\WINDOWS\\system32\\python32.zip',
      'C:\\Python32\\DLLs',
      'C:\\Python32\\lib',
      'C:\\Python32\\lib\\plat-win',
      'C:\\Python32',
      'C:\\Python32\\lib\\site-packages']

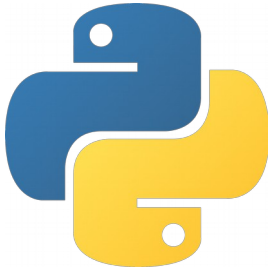
Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - platform
    - Dieser String enthält eine Kennung des zugrundeliegenden Betriebssystems
    - Die Kennungen der drei gängigsten Betriebssysteme

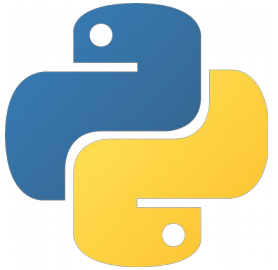
System	Kennung
Linux	`"linux2"` bzw. `"linux"`
Mac OS X	`"darwin"`
Windows	`"win32"`

Python Betriebssystem



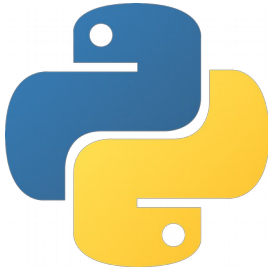
- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - platform
    - Hinweis
      - Bis Python 3.2 ist der Wert von platform auch unter Linux 3.x "linux2"
      - Ab Python 3.3 ist der Wert unter Linux "linux" unabhängig von der Linux-Version
      - Aus diesem Grund ist es ratsam, mittels sys.platform.startswith("linux") zu prüfen, ob das Programm auf einem Linux-System ausgeführt wird

Python Betriebssystem



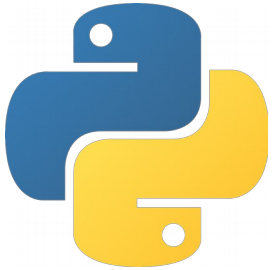
- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - version
    - Dies ist ein String, der die Versionsnummer des Python-Interpreters und einige weitere Informationen, wie beispielsweise das Datum seiner Kompilierung und den verwendeten Compiler, enthält
    - Beispiel
      >>> print(sys.version)
      3.2.2 (default, Sep 5 2011, 04:52:19)
      [GCC 4.6.1 20110819 (prerelease)]
    - Im Gegensatz zu hexversion nicht garantiert ist, dass die Versionsnummern mit den Operatoren > und < sinnvoll miteinander verglichen werden können

Python Betriebssystem



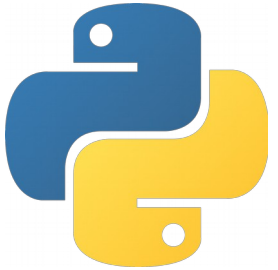
- Zugriff auf die Laufzeitumgebung – sys
  - Konstanten
  - version_info
    - Dies ist ein benanntes Tupel, das die einzelnen Komponenten der Versionsnummer des Interpreters enthält
    - Beispiel
      >>> sys.version_info
      sys.version_info(major=3, minor=2, micro=2, releaselevel='final', serial=0)
    - Auf die einzelnen Elemente der Versionsnummer lässt sich über einen Index oder über den Komponentennamen zugreifen

Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Exceptions
  - Das Modul sys enthält einige Funktionen, die speziell dazu gedacht sind, Zugriff auf geworfene Exceptions zu erhalten oder anderweitig mit Exceptions zu arbeiten
  - exc_info()
    - Diese Funktion ermöglicht es, Zugriff auf eine momentan abgefangene Exception zu erlangen
    - Momentan abgefangen bedeutet, dass sich der Kontrollfluss innerhalb eines except-Zweiges einer try/except-Anweisung befinden muss, damit diese Funktion einen sinnvollen Wert zurückgibt

Python Betriebssystem



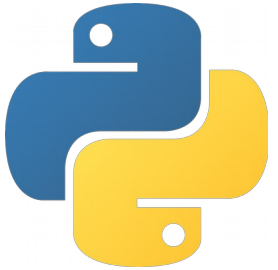
- Zugriff auf die Laufzeitumgebung – sys
  - Exceptions
  - exc_info()
    - Die Funktion exc_info gibt ein Tupel zurück, das drei Werte enthält

1. den Exception-Typ
2. die geworfene Instanz des Exception-Typs
3. das entsprechende Traceback-Objekt ← Später mehr

- Beispiel

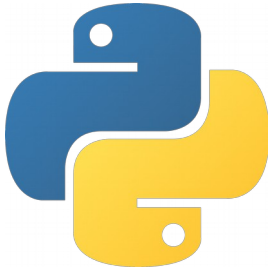
```
>>> try:
…         raise ValueError("Test")
... except ValueError:
…         print(sys.exc_info())
(<class 'ValueError'>, ValueError('Test',), <traceback object at
0x7fea775d3998>)
```

Python Betriebssystem



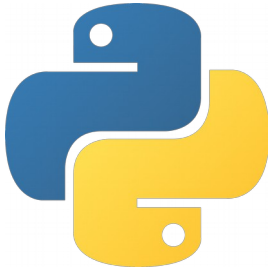
- Zugriff auf die Laufzeitumgebung – sys
  - Exceptions
  - exc_info()
    - Die Informationen über die aktuell abgefangene Exception bleiben nicht erhalten, sondern sind nur innerhalb des except-Zweiges verwendbar
    - Falls Sie Informationen über die zuletzt geworfene Exception außerhalb eines except-Zweiges benötigen, sollten Sie last_type, last_value oder last_traceback verwenden

Python Betriebssystem



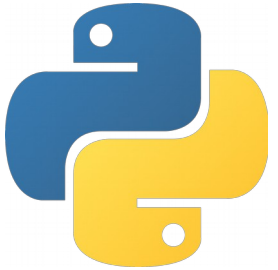
- Zugriff auf die Laufzeitumgebung – sys
  - Exceptions
  - last_type, last_value, last_traceback
    - Diese Referenzen erlauben es, Zugriff auf die zuletzt geworfene Exception zu erlangen
    - Die drei Informationen entsprechen denen, die von exc_info zurückgegeben werden
    - Im Gegensatz zu den von exc_info zurückgegebenen Werten sind diese Referenzen auch außerhalb eines except-Zweiges gültig, da sie stets Informationen über die zuletzt geworfene Exception enthalten

Python Betriebssystem



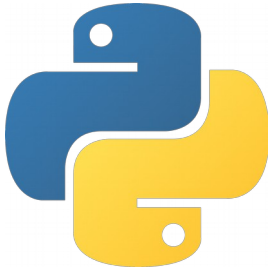
- Zugriff auf die Laufzeitumgebung – sys
  - Exceptions
  - tracebacklimit
    - Diese ganze Zahl kennzeichnet die maximale Tiefe, bis zu der ein Traceback Informationen über die Funktionshierarchie liefern soll
    - Initial ist dieser Wert auf 1000 gesetzt
    - Ein Wert von 0 veranlasst, dass ein Traceback nur aus dem Exception-Typ und der Fehlermeldung besteht

Python Betriebssystem



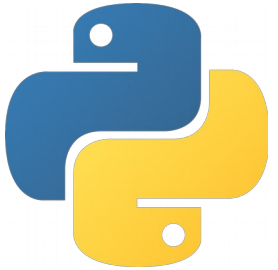
- Zugriff auf die Laufzeitumgebung – sys
  - Hooks
  - Das Modul sys erlaubt den Zugriff auf sogenannte Hooks
  - Das sind Funktionen, die bei gewissen Aktionen des Python-Interpreters aufgerufen werden
  - Durch Überschreiben dieser Funktionen kann sich der Programmierer in den Interpreter »einhaken« und so die Funktionsweise des Interpreters verändern

Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Hooks
  - displayhook(value)
    - Diese Funktion wird immer dann aufgerufen, wenn das Ergebnis eines Ausdrucks im interaktiven Modus ausgegeben werden soll
    - Beispiel
      >>> 42
      42
    - Durch Überschreiben von displayhook mit einer eigenen Funktion lässt sich dieses Verhalten ändern

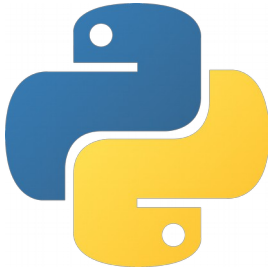
Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Hooks
  - displayhook(value)
    - Beispiel (Ausgabe der Identität, statt dem Wert)

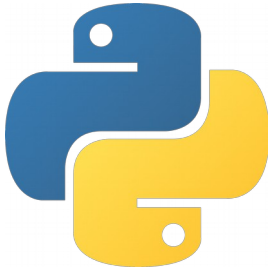
```
>>> def f(value):
…                 print(id(value))
...
>>> sys.displayhook = f
>>> 42
134536524
>>> 97 + 32
134537456
>>> "Hallo Welt"
3083420560
```

Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Hooks
  - excepthook(type, value, traceback)
    - Diese Funktion wird immer dann aufgerufen, wenn eine nicht abgefangene Exception auftritt
    - Sie ist dafür verantwortlich, den Traceback auszugeben
    - Durch Überschreiben dieser Funktion mit einem eigenen Funktionsobjekt lassen sich zum Beispiel Fehler protokollieren oder die Ausgabe eines Tracebacks verändern
    - Die drei Parameter der Funktion entsprechen denen, die von exc_info zurückgegeben werden, und enthalten Informationen über die Exception

Python Betriebssystem

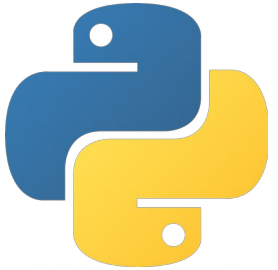


- Zugriff auf die Laufzeitumgebung – sys
  - Hooks
  - excepthook(type, value, traceback)
    - Wir möchten einen Hook einrichten, damit bei einer nicht abgefangenen Exception kein dröger Traceback mehr ausgegeben wird, sondern ein hämischer Kommentar

```
>>> def f(type, value, traceback):
...
 print('gnahahaha: "{}"'.format(value))
...
>>> sys.excepthook = f
>>> abc
gnahahaha: "name 'abc' is not defined"
```

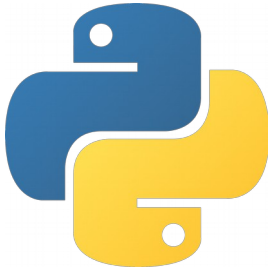
    - Das ursprüngliche Funktionsobjekt von excepthook können Sie über sys.__excepthook__ erreichen und somit die ursprüngliche Funktionsweise wiederherstellen

Python Betriebssystem



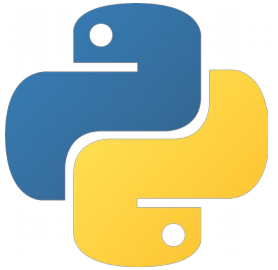
- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - exit([arg])
    - Diese Funktion wirft eine SystemExit-Exception
    - Diese hat, sofern sie nicht abgefangen wird, zur Folge, dass das Programm ohne Traceback-Ausgabe beendet wird
    - Als optionalen Parameter arg können Sie, wenn es sich um eine ganze Zahl handelt, einen Exit Code ans Betriebssystem übergeben
    - Ein Exit Code von 0 steht im Allgemeinen für erfolgreiches Beenden des Programms, und ein Exit Code ungleich 0 repräsentiert Programmabbruch aufgrund eines Fehlers

Python Betriebssystem



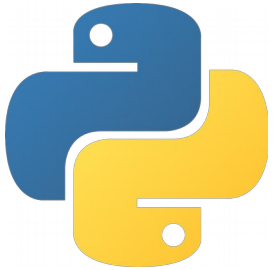
- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - exit([arg])
    - Wenn Sie eine andere Instanz für arg übergeben haben, beispielsweise einen String, wird diese nach stderr ausgegeben, bevor das Programm mit dem Exit Code 0 beendet wird

Python Betriebssystem



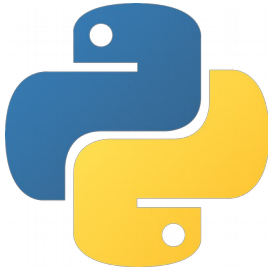
- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - getrefcount(object)
    - Diese Funktion gibt den aktuellen Reference Count für die übergebene Instanz object zurück
    - Der Reference Count ist eine ganze Zahl und entspricht der Anzahl von Referenzen, die auf eine Instanz bestehen
    - Wenn eine Instanz einen Reference Count von 0 hat, kann sie vom Garbage Collector entsorgt werden
    - Beachten Sie, dass es dem Interpreter bei Instanzen unveränderlicher Datentypen freisteht, eine neue Instanz zu erzeugen oder eine bereits bestehende neu zu referenzieren

Python Betriebssystem



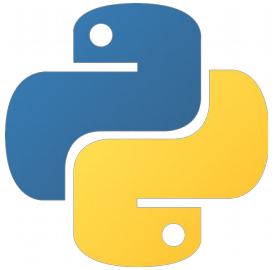
- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - getrecursionlimit(), setrecursionlimit(limit)
    - Mit diesen Funktionen wird die maximale Rekursionstiefe ausgelesen oder verändert
    - Die maximale Rekursionstiefe ist mit 1000 vorbelegt und bricht beispielsweise endlos rekursive Funktionsaufrufe ab, bevor diese zu einem Speicherüberlauf führen können

Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - getwindowsversion() ← Nur Windows
    - Diese Funktion erlaubt es, die Details über die Version des aktuell verwendeten Windows-Betriebssystems auszulesen
    - Die Funktion gibt ein benanntes Tupel zurück, dessen erste drei Elemente ganze Zahlen sind und die Versionsnummer beschreiben
    - Das vierte Element ist ebenfalls eine ganze Zahl und steht für die verwendete Plattform

Python Betriebssystem

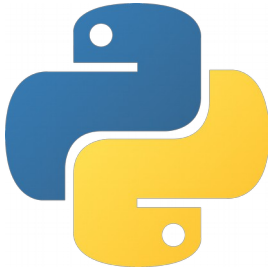


- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen

  - getwindowsversion() ← Nur Windows

Plattform	Bedeutung
0	Windows 3.1 (32-Bit)
1	Windows 95/98/ME
2	Windows NT / 2000 / XP / Server 2003 / Vista / Server 2008 / 7
3	Windows CE

Python Betriebssystem

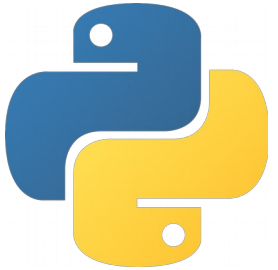


- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - getwindowsversion() ← Nur Windows
    - Das letzte Element des Tupels ist ein String, der weiterführende Informationen enthält

      ```
      >>> sys.getwindowsversion()
      sys.getwindowsversion(major=5, minor=1, build=2600,
      platform=2,
      service_pack='Service Pack 3')
      ```

    - Auf das benannte Tupel kann wie bei version_info sowohl über Indizes als auch über die Komponentennamen zugegriffen werden

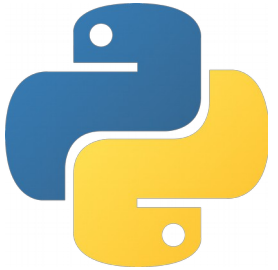
Python Betriebssystem



- Zugriff auf die Laufzeitumgebung – sys
  - Sonstige Funktionen
  - getwindowsversion() ← Nur Windows
    - Um zwischen den einzelnen Windows-Versionen zu unterscheiden, die unter Plattform 2 zusammengefasst sind, kann die Versionsnummer (bestehend aus dem major- und dem minor-Feld des Tupels) herangezogen werden

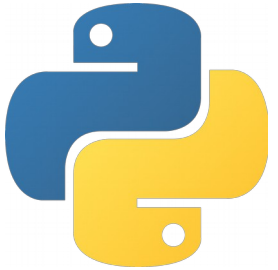
Name	Major	Minor
Windows 2000	5	0
Windows XP	5	1
Windows Server 2003	5	2
Windows Vista bzw. Windows Server 2008	6	0
Windows 7	6	1

Python Betriebssystem



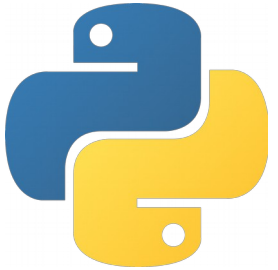
- Informationen über das System – platform
  - Das Modul platform der Standardbibliothek stellt Informationen über das Betriebssystem bzw. die zugrundeliegende Hardware bereit
  - Diese Informationen sind teilweise deckungsgleich mit denen, auf die Sie über das Modul sys zugreifen
  - Aus diesem Grund werden wir hier nur die wichtigsten Funktionen erläutern

Python Betriebssystem



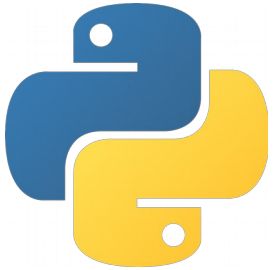
- Informationen über das System – platform
  - Die Wichtigsten
  - platform.machine()
    - Diese Funktion gibt die Prozessorarchitektur des PCs als String zurück
    - Bei aktuellen PCs ist dies i686 (32-Bit Systeme) oder x86_64 (64-Bit Systeme)

Python Betriebssystem



- Informationen über das System – platform
  - Die Wichtigsten
  - platform.node()
    - Diese Funktion gibt den Netzwerknamen des PCs als String zurück
  - platform.processor()
    - Diese Funktion gibt einen String zurück, der den Typ und den Hersteller des Prozessors enthält
  - platform.system()
    - Diese Funktion gibt einen String zurück, der Name des OS, beispielsweise also »Linux« oder »Windows«

Python Betriebssystem



- Informationen über das System – platform
  - Weiteres in Eigenverantwortung ermitteln

    - Kommandozeilenparameter – argparse
    - Kopieren von Instanzen – copy
    - Das Programmende – atexit