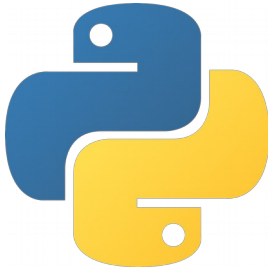
Python Strings

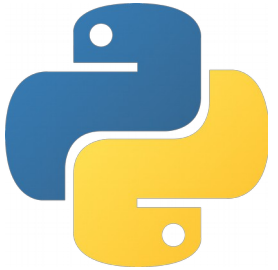


- Lokalisierung von Programmen – gettext
  - Beispiel für die Verwendung von gettext

```
import gettext, random
trans = gettext.translation("meinprogramm", "locale", ["de"])
trans.install()

werte = []
while True:
        w = input(_("Please enter a value: "))
        if not w:
                break
        werte.append(w)
print(_("The random choice is {0}").format(random.choice(werte)))
```

Python Strings



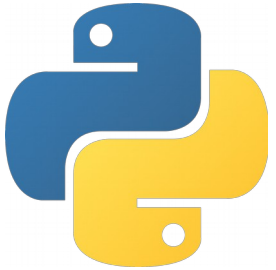
- Lokalisierung von Programmen – gettext
  - Beispiel für die Verwendung von gettext
  - Der eigentlich interessante Teil des Programms sind diese beiden Zeilen

    ```
    trans = gettext.translation("meinprogramm", "locale", ["de"])
    trans.install()
    ```

  - Hier wird ein sogenanntes Translation-Objekt erstellt
  - Das ist eine Instanz, die die Übersetzung aller Strings in eine bestimmte Sprache gewährleistet

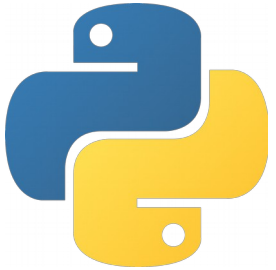
  - Einen frei wählbaren Namen, die sogenannte Domain, als ersten Parameter
  - Der zweite Parameter ist das Unterverzeichnis, in dem sich die Übersetzungen befinden
  - Der dritte Parameter ist schließlich eine Liste von Sprachen

Python Strings



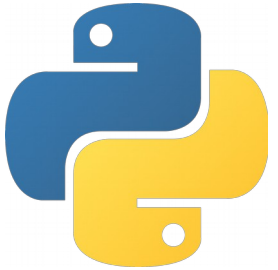
- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats
  - Zum Erstellen des Sprachkompilats müssen Sie zunächst eine Liste aller zu übersetzenden Strings erstellen
  - Das sind all jene, die vor der Ausgabe durch die Funktion _ geschickt werden
  - Da es eine unzumutbare Arbeit wäre, diese Liste von Hand anzufertigen, ist in Python ein Programm namens pygettext.py im Lieferumfang enthalten, das genau dies erledigt
  - Das Programm erstellt eine sogenannte .po-Datei

Python Strings



- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats
  - Das ist eine für Menschen lesbare Variante des .mo-Dateiformats
  - Diese .po-Datei wird dann von den Übersetzern in verschiedene Sprachen übersetzt
  - Dies kann von Hand geschehen oder durch Einsatz diverser Tools, die für diesen Zweck existieren

Python Strings



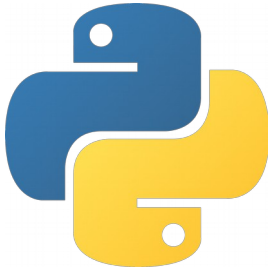
- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats
  - Die für unser Beispielprogramm erstellte .po-Datei sieht folgendermaßen aus

    [...]

    #: main.py:9
    msgid "Please enter a value: "
    msgstr "Bitte geben Sie einen Wert ein: "

    #: main.py:13
    msgid "The random choice is {0}"
    msgstr "Die Zufallswahl ist {0}"

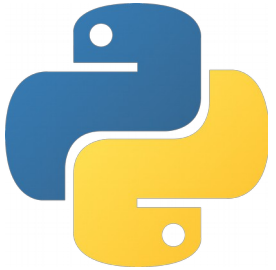
Python Strings



- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats
  - Anstelle der [...] enthält die Datei Informationen wie etwa den Autor, die verwendete Software oder das Encoding der Datei
  - Eine übersetzte .po-Datei wird durch das Programm msgfmt.py, das ebenfalls zum Lieferumfang von Python gehört, in das binäre .mo-Format kompiliert
  - Ein fertiges Sprachkompilat muss sich in folgendem Ordner befinden, damit es von gettext als solches gefunden wird

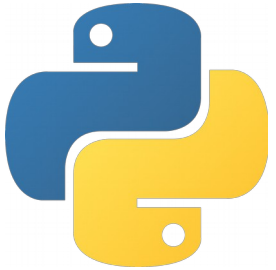
    Programmverzeichnis/Unterordner/Sprache/LC_MESSAGES/Domain.mo

Python
Strings



- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats
  - Beim Aufruf der Funktion gettext.translate wird der Name des Verzeichnisses Unterordner angegeben
  - Er war in unserem Beispiel locale
  - Dieses Verzeichnis muss für jede vorhandene Sprache ein weiteres Verzeichnis enthalten, das seinerseits über ein Unterverzeichnis LC_MESSAGES verfügen muss
  - Das Sprachkompilat selbst muss die im Programm angegebene Domain als Namen haben

Python Strings

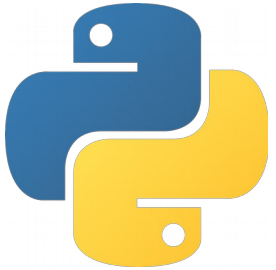


- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats
  - In unserem Beispiel muss das Sprachkompilat also in folgendem Verzeichnis liegen

    Programmverzeichnis/locale/de/LC_MESSAGES/meinprogramm.mo
  - Wenn das Sprachkompilat nicht vorhanden ist, wird beim Aufruf der Funktion gettext.translate eine entsprechende Exception geworfen

    Traceback (most recent call last):
    [...]
    IOError: [Errno 2] No translation file found for domain: 'meinprogramm'

Python
Strings

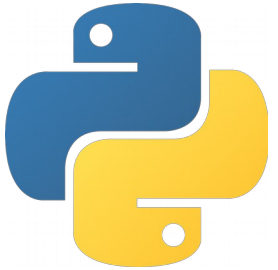


- Lokalisierung von Programmen – gettext
  - Erstellen des Sprachkompilats

  - Wenn das Sprachkompilat an seinem Platz ist, werden Sie beim Ausführen des Programms feststellen, dass alle Strings ins Deutsche übersetzt wurden

    Bitte geben Sie einen Wert ein: Donald Duck
    Bitte geben Sie einen Wert ein: Daisy Duck
    Bitte geben Sie einen Wert ein: Onkel Dagobert
    Bitte geben Sie einen Wert ein:
    Die Zufallswahl ist Donald Duck

Python
Strings

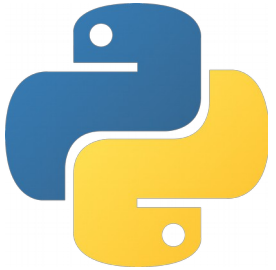


- Lokalisierung von Programmen – gettext
  - Hash-Funktionen – hashlib
  - Das folgende kleine Beispiel verwendet das Modul hashlib, um einen Passwortschutz zu realisieren

```
import hashlib
pwhash = "578127b714de227824ab105689da0ed2"
m = hashlib.md5(bytes(input("Ihr Passwort bitte: "), "utf-8"))
if pwhash == m.hexdigest():
        print("Zugriff erlaubt")
else:
        print("Zugriff verweigert")
```

  - Das Passwort lautet »Mein Passwort«

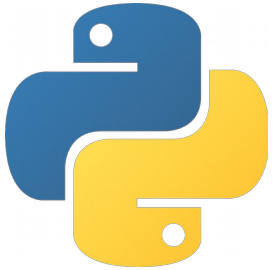
Python
Strings



- Lokalisierung von Programmen – gettext
  - Hash-Funktionen – hashlib

Klasse	Algorithmus	Beschreibung
md5	MD5	*Message-Digest Algorithm 5* Erzeugt aus einem beliebigen String einen 128-Bit-Hash-Wert. Beachten Sie, dass der MD5-Algorithmus bereits ansatzweise geknackt wurde.
sha1	SHA-1	*Secure Hash Algorithm 1* Erzeugt aus einem beliebigen String einen 160-Bit-Hash-Wert. Beachten Sie, dass der SHA-1-Algorithmus bereits ansatzweise geknackt wurde.

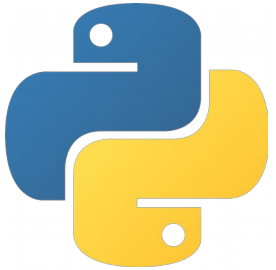
Python Strings



- Lokalisierung von Programmen – gettext
  - Hash-Funktionen – hashlib

Klasse	Algorithmus	Beschreibung
sha224	SHA-224	*Secure Hash Algorithm 224* Erzeugt aus einem beliebigen String einen 224-Bit-Hash-Wert.
sha256	SHA-256	*Secure Hash Algorithm 256* Erzeugt aus einem beliebigen String einen 256-Bit-Hash-Wert.
sha384	SHA-384	*Secure Hash Algorithm 384* Erzeugt aus einem beliebigen String einen 384-Bit-Hash-Wert.
sha512	SHA-512	*Secure Hash Algorithm 512* Erzeugt aus einem beliebigen String einen 512-Bit-Hash-Wert.

Python
Strings

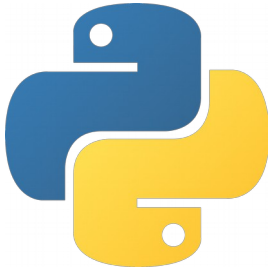


- Lokalisierung von Programmen – gettext
  - Hash-Funktionen – Die wichtigsten

```
>>> import hashlib
>>> m = hashlib.md5(b"Hallo Welt")
>>> print(m.hexdigest())
55243ecf175013cfe9890023f9fd9037
```

  - Durch Aufruf der Methode hexdigest wird der berechnete Hash-Wert als String zurückgegeben, der eine Folge von zweistelligen Hexadezimalzahlen enthält
  - Diese Hexadezimalzahlen repräsentieren jeweils ein Byte des Hash-Wertes
  - Der zurückgegebene String enthält ausschließlich druckbare Zeichen

Python
Strings



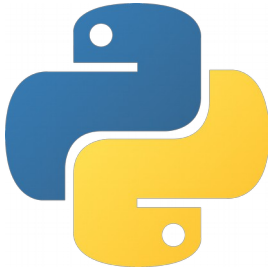
- Lokalisierung von Programmen – gettext
  - Hash-Funktionen – Die wichtigsten

```
>>> import hashlib
>>> m = hashlib.md5(b"Hallo Welt")
>>> print(m.digest())
b'U$>\xcf\x17P\x13\xcf\xe9\x89\x00#\xf9\xfd\x907'
```

  - Durch Aufruf der Methode digest wird der berechnete Hash-Wert als Bytefolge zurückgegeben

  - Beachte
    - Die zurückgegebene bytes-Instanz kann durchaus nicht-druckbare Zeichen enthalten

Python
Strings



- Lokalisierung von Programmen – gettext
  - Hash-Funktionen – Die wichtigsten

    Beachte
    - Beim Instanziieren der Klasse md5 wird eine bytes-Instanz übergeben, deren Hash-Wert berechnet werden soll

    - Das Berechnen eines Hash-Wertes aus einem String ist seit Python 3.0 nicht mehr möglich