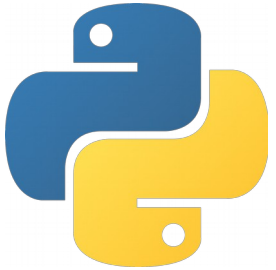
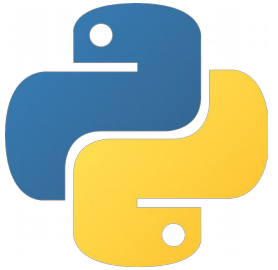
Python Betriebssystem



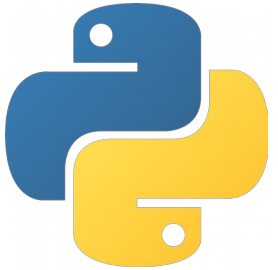
- Funktionen des Betriebssystems – os
  - Mit dem os-Modul können Sie auf mehrere Klassen von Operationen zugreifen
  - Da die gebotenen Funktionen sehr umfangreich sind und zu einem großen Teil nur selten gebraucht werden, beschränken wir uns hier auf ein paar wichtige
  - Die paar wichtigen lassen sich in drei Kategorien einteilen
    1. Zugriff auf den Prozess, in dem unser Python-Programm läuft, und auf andere Prozesse
    2. Zugriff auf das Dateisystem
    3. Informationen über das Betriebssystem

Python Betriebssystem



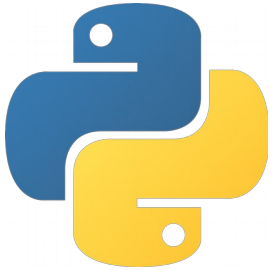
- Funktionen des Betriebssystems – os
  - Außerdem stellt das Submodul os.path nützliche Operationen für die Manipulation und Verarbeitung von Pfadnamen bereit
  - Das Modul os hat eine eigene Exception-Klasse namens os.error
  - Ein alternativer Name für die Fehlerklasse ist OSError
  - Hinweis
  - Seit Python 3.0 wird streng zwischen Text und Daten durch die Datentypen str und bytes unterschieden
  - Alle Methoden und Funktionen, die von os bereitgestellt werden und str -Objekte als Parameter akzeptieren, können stattdessen auch mit bytes-Objekten gerufen werden
  - Kurz: str rein – str raus; bytes rein – bytes raus

Python Betriebssystem



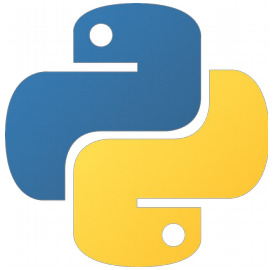
- Funktionen des Betriebssystems – os
  - Zugriff auf den eigenen Prozess und andere Prozesse
  - environ
    - Diese Konstante enthält ein Dictionary, das die Umgebungsvariablen speichert, die für unser Programm vom Betriebssystem bereitgestellt wurden
    - Beispielsweise lässt sich auf vielen Plattformen mit os.environ['HOME'] der Pfad des Ordners für die Dateien des aktiven Benutzers ermitteln

Python Betriebssystem



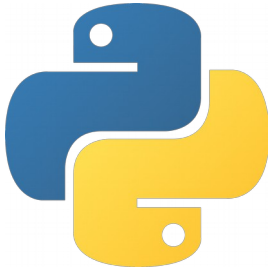
- Funktionen des Betriebssystems – os
  - Zugriff auf den eigenen Prozess und andere Prozesse
  - environ
  - Beispiel (Windows)
    >>> print(os.environ['HOME'])
    C:\Dokumente und Einstellungen\username
  - Beispiel(Linux/UNIX)
    >>> print(os.environ['HOME'])
    /home/username
  - Sie können die Werte des os.environ-Dictionarys auch verändern, was allerdings auf bestimmten Plattformen zu Problemen führen kann und deshalb mit Vorsicht zu genießen ist

Python Betriebssystem



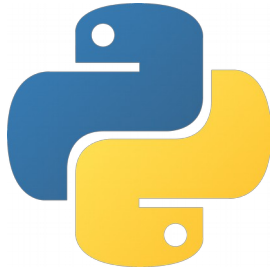
- Funktionen des Betriebssystems – os
  - Zugriff auf den eigenen Prozess und andere Prozesse
  - getpid()
    - Der Python-Prozess, der das aktuell laufende Programm ausführt, hat eine eindeutige Identifikationsnummer. Sie lässt sich mit os.getpid() ermitteln
    - Beispiel
      >>> os.getpid()
      1360
    - Diese Funktion ist nur unter Windows- und Unix-Systemen verfügbar

Python Betriebssystem



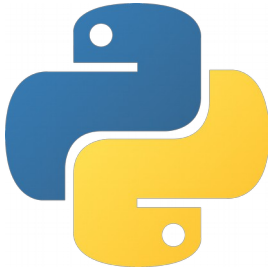
- Funktionen des Betriebssystems – os
  - Zugriff auf den eigenen Prozess und andere Prozesse
  - system(cmd)
    - Mit os.system können Sie beliebige Kommandos des Betriebssystems ausführen, so als ob Sie es in einer echten Konsole tun würden
    - Beispielsweise lassen wir uns mit folgendem Beispiel einen neuen Ordner mit dem Namen test_ordner über das mkdir-Kommando anlegen
    - Beispiel
      >>> os.system("mkdir test_ordner")
      0

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf den eigenen Prozess und andere Prozesse
  - popen(command[, mode[, bufsize]])
    - Mit der Funktion os.popen werden beliebige Befehle wie auf einer Kommandozeile des Betriebssystems ausgeführt
    - Die Funktion gibt ein dateiähnliches Objekt zurück, mit dem Sie auf die Ausgabe des ausgeführten Programms zurückgreifen können
    - Der Parameter mode gibt wie bei der Built-in Function open an, ob das Dateiobjekt lesend ("r") oder schreibend ("w") geöffnet werden soll

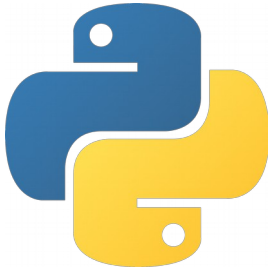
Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf den eigenen Prozess und andere Prozesse
  - popen(command[, mode[, bufsize]])
    - Bei schreibendem Zugriff können auch Daten an das laufende Programm übergeben werden
    - Beispiel

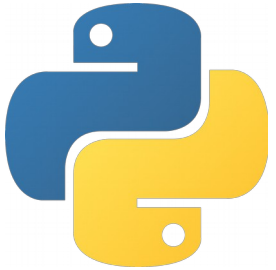
```
>>> ausgabe = os.popen("ls -la /opt")
>>> dateien = [zeile.strip() for zeile in ausgabe]
>>> dateien
['insgesamt 36', 'drwxr-xr-x  9 root  root  4096 Nov 28
14:13 .', 'drwxr-xr-x 24 root  root  4096 Nov 27 20:40 ..',
< .. .>', 'drwxr-xr-x  6 root  root  4096 Nov 28 13:05
vivaldi-beta']
```

Python Betriebssystem



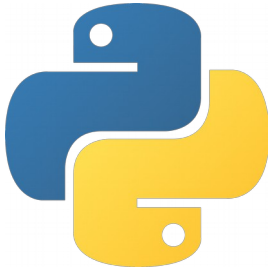
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - Mit den nachfolgend beschriebenen Funktionen können Sie sich wie mit einer Shell oder einem Dateimanager durch das Dateisystem bewegen
  - Informationen zu Dateien und Ordnern ermitteln, diese umbenennen, löschen oder erstellen
  - Sie werden oft einen sogenannten Pfad als Parameter an die beschriebenen Funktionen übergeben können
  - Dabei unterscheiden wir zwischen absoluten und relativen Pfaden, wobei letztere sich auf das aktuelle Arbeitsverzeichnis beziehen

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - access(path, mode)
    - Mit access überprüfen Sie, welche Rechte das laufende Python-Programm für den Pfad path hat
    - Der Parameter mode gibt dabei eine Bitmaske an, die die zu überprüfenden Rechte enthält
    - Die Werte auf der nächsten Seite können einzeln oder mithilfe des bitweisen ODER zusammengefasst übergeben werden

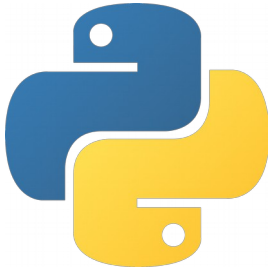
Python
Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - access(path, mode)

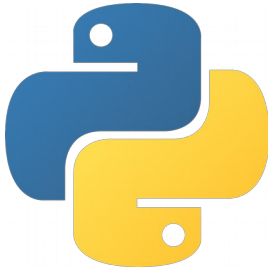
Konstante	Bedeutung
F_OK	Prüft, ob der Pfad überhaupt existiert.
R_OK	Prüft, ob der Pfad gelesen werden darf.
W_OK	Prüft, ob der Pfad geschrieben werden darf.
X_OK	Prüft, ob der Pfad ausführbar ist.

Python Betriebssystem



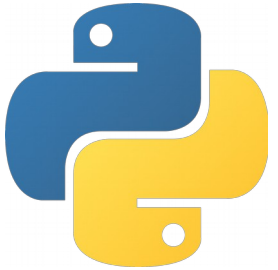
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - access(path, mode)
    - Der Rückgabewert von access ist True, wenn alle für mode übergebenen Werte auf den Pfad zutreffen
    - False, wenn mindestens ein Zugriffsrecht für das Programm nicht gilt
    - Beispiel
      >>> os.access("C:\\Python32\\python.exe", os.F_OK | os.X_OK)
      True

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - chmod(path, mode)
    - Diese Funktion setzt die Zugriffsrechte der Datei oder des Ordners unter dem übergebenen Pfad
    - mode ist dabei eine dreistellige Oktalzahl, bei der jede Ziffer die Zugriffsrechte für eine Benutzerklasse angibt
    - Die erste Ziffer steht für den Besitzer der Datei, die zweite für seine Gruppe und die dritte für alle anderen Benutzer

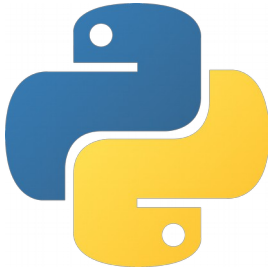
Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - chmod(path, mode)
    - Dabei sind die einzelnen Ziffern Summen aus den folgenden drei Werten

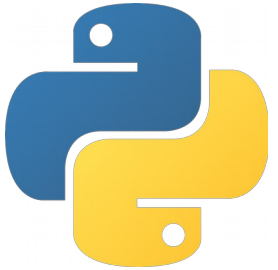
Wert	Beschreibung
1	ausführen
2	schreiben
4	lesen

Python Betriebssystem



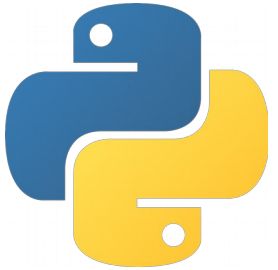
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - chmod(path, mode) → Nur Windows- UNIX-Systeme
    - Beispiel
      (erteilen von vollen Lese- und Schreibzugriff für Besitzer)
      >>> os.chmod("eine_datei", 0o640)
    - Ausführen kann er die Datei aber trotzdem nicht
    - Die restlichen Benutzer seiner Gruppe dürfen die Datei auslesen, aber nicht verändern, und für alle anderen bleibt aufgrund der fehlenden Leseberechtigung auch der Inhalt der Datei verborgen
    - Beachten Sie das führende 0o bei den Zugriffsrechten, das das Literal einer Oktalzahl einleitet

Python Betriebssystem



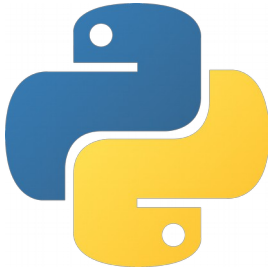
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - listdir(path)
    - Diese Funktion gibt eine Liste zurück, die alle Dateien und Unterordner des Ordners angibt, der mit path übergeben wurde
    - Diese Liste enthält nicht die speziellen Einträge für das Verzeichnis selbst (".") und für das nächsthöhere Verzeichnis ("..")
    - Die Elemente der Liste haben den gleichen Typ wie der übergebene path-Parameter, also entweder str oder bytes
    - Dabei werden auftretende Sonderzeichen mithilfe von UTF-8 kodiert

Python Betriebssystem



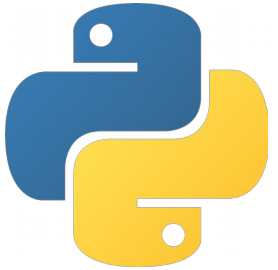
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - mkdir(path[, mode])
    - Diese Funktion legt einen neuen Ordner in dem mit path übergebenen Pfad an
    - Der optionale Parameter mode gibt dabei eine Bitmaske an, die die Zugriffsrechte für den neuen Ordner festlegt
    - Standardmäßig wird für mode die Oktalzahl 0o777 verwendet

Python Betriebssystem



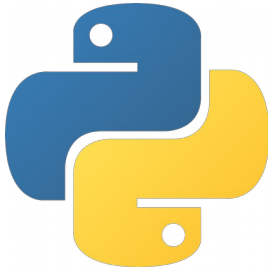
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - mkdir(path[, mode])
    - Ist der angegebene Ordner bereits vorhanden, wird eine os.error-Exception geworfen
    - Beachten Sie, dass mkdir nur dann den neuen Ordner erstellen kann, wenn alle übergeordneten Verzeichnisse bereits existieren
    - Beispiel
      >>> os.mkdir(r"C:\Diesen\Pfad\gibt\es\so\noch\nicht")
      [...]
      WindowsError: [Error 3] Das System kann den angegebenen Pfad nicht finden:
      'C:\\Diesen\\Pfad\\gibt\\es\\so\\noch\\nicht'

Python Betriebssystem



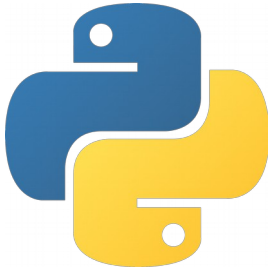
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - makedirs(path[, mode])
    - Wie mkdir; aber diese Funktion erzeugt im Gegensatz dazu die komplette Verzeichnisstruktur inklusive aller übergeordneten Verzeichnisse
    - Beispiel
      >>> os.makedirs(r"C:\Diesen\Pfad\gibt\es\so\noch\nicht")
      >>>
    - Wenn der übergebene Ordner schon existiert, wird eine os.error-Exception geworfen

Python Betriebssystem



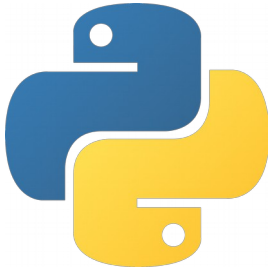
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - remove(path)
    - Diese Funktion entfernt die mit path angegebene Datei aus dem Dateisystem
    - Übergeben Sie statt eines Pfads zu einer Datei einen Pfad zu einem Ordner, wirft remove eine os.error-Exception
    - Beachten Sie bitte, dass es unter Windows-Systemen nicht möglich ist, eine Datei zu löschen, die gerade benutzt wird
      - In diesem Fall wird ebenfalls eine Exception geworfen

Python Betriebssystem



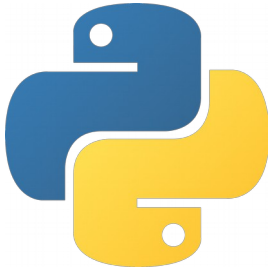
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - removedirs(path)
    - Diese Funktion löscht eine ganze Ordnerstruktur, wobei von der tiefsten bis zur höchsten Ebene nacheinander alle Ordner gelöscht werden, sofern diese leer sind
    - Kann der tiefste Ordner nicht gelöscht werden, wird eine os.error-Exception geworfen
    - Fehler, die beim Entfernen der Elternverzeichnisse auftreten, werden ignoriert

Python Betriebssystem



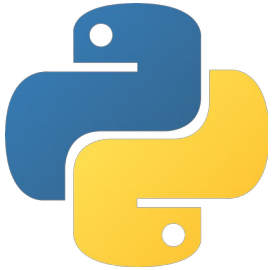
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - removedirs(path)
    - Beispiel
      >>> os.removedirs(r"C:\Irgend\ein\Beispielpfad")
    - Zuerst wird versucht, den Ordner C:\Irgend\ein\Beispielpfad zu löschen
    - Wenn dies erfolgreich war, wird C:\Irgend\ein entfernt und bei Erfolg anschließend C:\Irgend

Python Betriebssystem



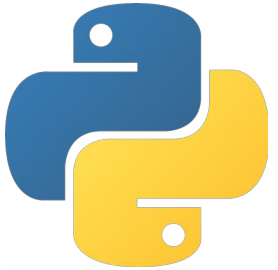
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - rename(src, dst)
    - Diese Funktion benennt die mit src angegebene Datei oder den Ordner in dst um
    - Wenn unter dem Pfad dst bereits eine Datei oder ein Ordner existieren, wird os.error geworfen
    - Hinweis
      - Auf Unix-Systemen wird eine bereits unter dem Pfad dst erreichbare Datei ohne Meldung überschrieben, wenn Sie rename aufrufen
      - Bei bereits existierenden Ordnern wird aber weiterhin eine Exception erzeugt

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - rename(src, dst)
    - Die Methode os.rename funktioniert nur dann, wenn bereits alle übergeordneten Verzeichnisse von dst existieren
    - Wenn Sie die Erzeugung der nötigen Verzeichnisstruktur wünschen, benutzen Sie stattdessen os.renames

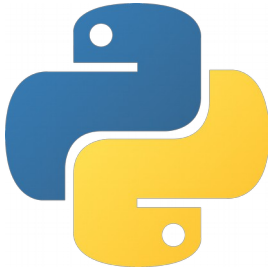
Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - walk(top[, topdown=True[, onerror=None]])
    - Eine komfortable Möglichkeit, einen Verzeichnisbaum komplett zu durchlaufen
    - Der Parameter top gibt die Wurzel des zu durchlaufenden Teilbaums an
    - Die Iteration geht dabei so vonstatten, dass walk für den Ordner top und für jeden seiner Unterordner ein Tupel mit drei Elementen zurückgibt

('ein\\pfad', ['ordner1'], ['datei1', 'datei2'])

Python Betriebssystem

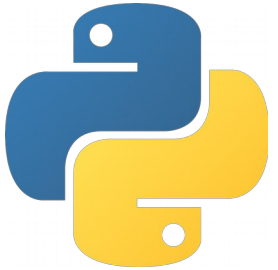


- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - walk(top[, topdown=True[, onerror=None]])

    ('ein\\pfad', ['ordner1'], ['datei1', 'datei2'])

    - Das erste Element ist der relative Pfad von top zu dem Unterordner
    - Das zweite Element enthält eine Liste mit allen Ordnern, die der aktuelle Unterordner selbst enthält
    - Das letzte Element speichert alle Dateien des Unterordners

Python Betriebssystem



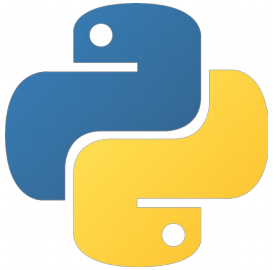
- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem

  - walk(top[, topdown=True[, onerror=None]])

Vorgabe:

```
ich
    eltern
        mutter
        vater
    freunde
        entfernte_freunde
                erwin
                heinz
        christian
        lucas
        peter
```

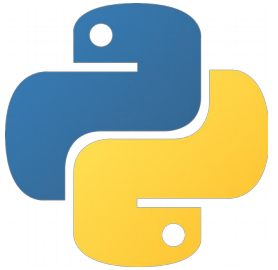
Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - walk(top[, topdown=True[, onerror=None]])
    - Beispiel (top → down)

```
>>> for t in os.walk("ich"):
        print(t)

('ich', ['freunde', 'eltern'], [])
('ich\\freunde', ['entfernte_freunde'],
['peter', 'christian', 'lucas'])
('ich\\freunde\\entfernte_freunde',[], ['heinz', 'erwin'])
('ich\\eltern', [], ['vater', 'mutter'])
```

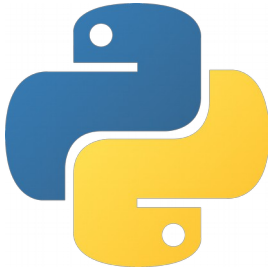
Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Zugriff auf das Dateisystem
  - walk(top[, topdown=True[, onerror=None]])
    - Beispiel (bottom → up)

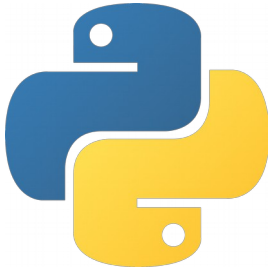
```
>>> for t in os.walk("ich", False):
        print(t)

('ich\\freunde\\entfernte_freunde', [], ['heinz', 'erwin'])
('ich\\freunde', ['entfernte_freunde'],
['peter', 'christian', 'lucas'])
('ich\\eltern', [], ['vater', 'mutter'])
('ich', ['freunde', 'eltern'], [])
```

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
    - Verschiedene Plattformen – verschiedene Pfadnamenskonventionen
    - Während beispielsweise Windows-Betriebssysteme bei absoluten Pfadnamen das Laufwerk erwarten, auf das sich der Pfad bezieht, wird unter Unix ein einfacher Slash vorangestellt
    - Außerdem unterscheiden sich auch die Trennzeichen für einzelne Ordner innerhalb des Pfadnamens, denn Microsoft hat sich im Gegensatz zur Unix-Welt, in der der Slash üblich ist, für den Backslash entschieden

Python Betriebssystem



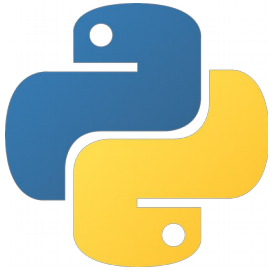
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path

    - Als Programmierer für plattformübergreifende Software stehen Sie nun vor dem Problem, dass Ihre Programme mit diesen verschiedenen Konventionen und auch denen dritter Betriebssysteme zurechtkommen müssen

    - Damit dafür keine programmtechnischen Verrenkungen notwendig werden, wurde das Modul os.path entwickelt, mit dem Sie Pfadnamen komfortabel verwenden können

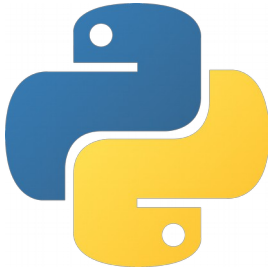
    - Sie können das Modul auf zwei Arten einbinden
      - Sie importieren erst os und greifen dann über os.path darauf zu
      - Sie importieren os.path direkt

Python Betriebssystem



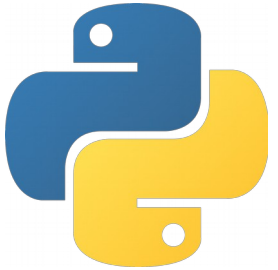
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - abspath(path)
    - Diese Funktion gibt zu einem relativen Pfad den dazugehörigen absoluten und normalisierten Pfad (siehe dazu os.normpath) zurück
    - Beispiel
      >>> os.path.abspath(".")
      'Z:\\beispiele\\os'
    - In diesem Fall haben wir mithilfe des relativen Pfads "." auf das aktuelle Verzeichnis herausgefunden, dass unser Script unter 'Z:\\beispiele\\os' gespeichert ist

Python Betriebssystem



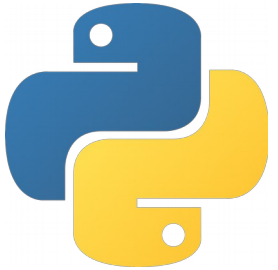
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - basename(path)
    - Diese Funktion gibt den sogenannten Basisnamen des Pfads zurück
    - Der Basisname eines Pfads ist der Teil hinter dem letzten Ordnertrennzeichen, wie zum Beispiel \ oder /
    - Diese Funktion eignet sich sehr gut, um den Dateinamen aus einem vollständigen Pfad zu extrahieren

Python Betriebssystem



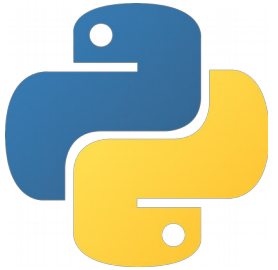
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - basename(path)
    - Beispiel
      >>>os.path.basename(r"C:\Windows\System32\ntoskrnl.exe")
      'ntoskrnl.exe'
    - Hinweis
      - Diese Funktion unterscheidet sich von dem Unix-Kommando basename dadurch, dass sie einen leeren String zurückgibt, wenn der String mit einem Ordnertrennzeichen endet
        >>> os.path.basename(r"/usr/lib/compiz/") ← Python
        ' '
        $ basename /usr/lib/compiz/ ← UNIX-Kommando
        compiz

Python Betriebssystem



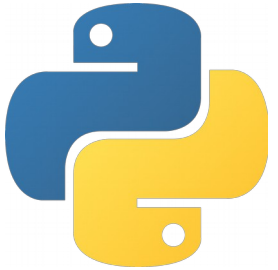
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - commonprefix(list)
    - Diese Funktion gibt einen möglichst langen String zurück, mit dem alle Elemente der als Parameter übergebenen Pfadliste list beginnen
    - Beispiel
      >>> os.path.commonprefix([r"C:\Windows\System32\ntoskrnl.exe",
      r"C:\Windows\System\TAPI.dll",
      r"C:\Windows\system32\drivers"])
      - 'C:\\Windows\\'
    - Es ist aber nicht garantiert, dass der resultierende String auch ein gültiger und existierender Pfad ist, da die Pfade als einfache Strings betrachtet werden

Python Betriebssystem



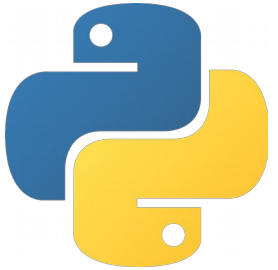
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - dirname(path)
    - Diese Funktion gibt den Ordnerpfad zurück, den path enthält
    - Beispiel
      >>> os.path.dirname(r"C:\Windows\System\TAPI.dll")
      'C:\\Windows\\System'

Python Betriebssystem



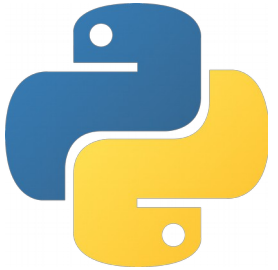
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - dirname(path)
    - Genau wie bei os.path.basename müssen Sie auch hier das abweichende Verhalten bei Pfaden beachten, die mit einem Ordnertrennzeichen enden
    - Beispiel
      >>> os.path.dirname(r"/usr/lib/compiz")
      '/usr/lib'
      >>> os.path.dirname(r"/usr/lib/compiz/")
      '/usr/lib/compiz'

Python Betriebssystem



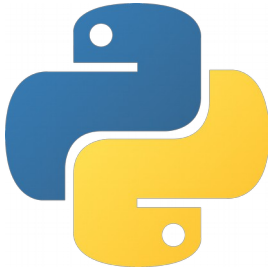
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - exists(path)
    - Diese Funktion gibt True zurück, wenn der angegebene Pfad auf eine existierende Datei oder ein vorhandenes Verzeichnis verweist, ansonsten False
  - getatime(path)
    - Diese Funktion gibt den Unix-Zeitstempel des letzten Zugriffs auf den übergebenen Pfad zurück
    - Kann auf die übergebene Datei oder den Ordner nicht zugegriffen werden oder ist sie bzw. er nicht vorhanden, führt dies zu einem os.error

Python Betriebssystem



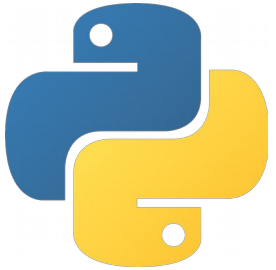
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - getmtime(path)
    - Diese Funktion gibt einen Unix-Zeitstempel zurück, der angibt, wann die Datei oder der Ordner unter path zum letzten Mal verändert wurden
    - Existiert der übergebene Pfad nicht im Dateisystem, wird os.error geworfen

Python Betriebssystem



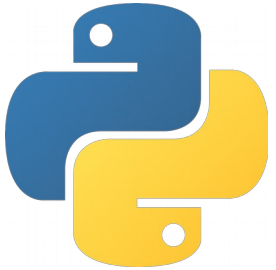
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - join(path1[, path2[, ...]])
    - Diese Funktion fügt die übergebenen Pfadangaben zu einem einzigen Pfad zusammen, indem sie verkettet werden
    - Dabei wird das übliche Trennzeichen des Betriebssystems verwendet
    - Beispiel
      >>> os.path.join(r"C:\Windows", r"System\ntoskrnl.exe")
      'C:\\Windows\\System\\ntoskrnl.exe'

Python Betriebssystem



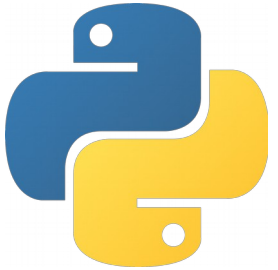
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - join(path1[, path2[, ...]])
    - Wird ein absoluter Pfad als zweites oder späteres Argument übergeben, ignoriert os.path.join alle übergebenen Pfade vor dem absoluten
    - Beispiel
      ```
      >>> os.path.join(r"Das\wird\ignoriert", r"C:\Windows",
                                                  r"System\ntoskrnl.exe")
      'C:\\Windows\\System\\ntoskrnl.exe'
      ```

Python Betriebssystem



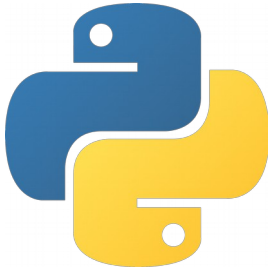
- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - normcase(path)
    - Auf Betriebssystemen, die bei Pfaden nicht hinsichtlich Groß- und Kleinschreibung unterscheiden (z.B. Windows), werden alle Großbuchstaben durch ihre kleinen Entsprechungen ersetzt
    - Außerdem werden unter Windows alle Slashs durch Backslashs ausgetauscht
    - Beispiel
      >>> os.path.normcase(r"C:\Windows/System32/ntoskrnl.exe")
      'c:\\windows\\system32\\ntoskrnl.exe'
    - Unter Unix wird der übergebene Pfad ohne Änderung zurückgegeben

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - split(path)
    - Diese Funktion teilt den übergebenen Pfad in den Namen des Ordners oder der Datei, die er beschreibt, und den Pfad zu dem direkt übergeordneten Verzeichnis und gibt ein Tupel zurück, das die beiden Teile enthält
    - Beispiel
      >>> os.path.split(r"C:\Windows\System32\ntoskrnl.exe")
      ('C:\\Windows\System32', 'ntoskrnl.exe')
    - Hinweis
    - Wenn der Pfad mit einem Slash oder Backslash endet, ist das zweite Element des Tupels ein leerer String
      >>> os.path.split("/home/revelation/")
      ('/home/revelation', '')

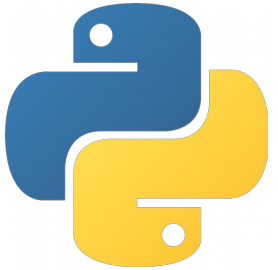
Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - splitdrive(path)
    - Diese Funktion teilt den übergebenen Pfad in die Laufwerksangabe und den Rest, sofern die Plattform Laufwerksangaben unterstützt
    - Beispiel
      >>> os.path.splitdrive(r"C:\Windows/System32/ntoskrnl.exe")
      ('C:', '\\Windows/System32/ntoskrnl.exe')
    - Unter Betriebssystemen, die keine Laufwerksbuchstaben unterstützen, ist der erste String des Tupels ein leerer String

      >>> os.path.splitdrive("/usr/share/bin")
      ('', '/usr/share/bin')

Python Betriebssystem



- Funktionen des Betriebssystems – os
  - Umgang mit Pfaden – os.path
  - splitext(path)
    - Diese Funktion teilt den path in den Pfad zu der Datei und die Dateiendung
    - Beide Elemente werden in einem Tupel zurückgegeben
    - Beispiel
      >>> os.path.splitext(r"C:\Windows\System32\Notepad.exe")
      ('C:\\Windows\\System32\\Notepad', '.exe')