Seite 278, 235

Wiederholung Seite 229, 215

Aufgabe 163 wiederholen,

Dicotonary.get(k, default) + 1 hinterhergehen und üben.

Zuweisungsanweisung. Seite 125

spam.sort(key=str.lower)

isX-Stringmethoden Seite 172

Die Methoden rjust() und ljust() werden für einen String aufgerufen und geben eine Variante dieses Strings zurück, bei der der Text mithilfe von Leerzeichen ausgerichtet ist. .center()

Seite 176

Das Modul pyperclip enthält die Funktionen copy() und paste(), die Text an die Zwischenablage des Computers senden bzw. von dort empfangen können.

in Anhang B. Dort erfahren Sie nicht nur, wie Sie Python-Skripte auf komfortable Weise ausführen, sondern auch, wie Sie Befehlszeilenargumente übergeben können (was nicht möglich ist, wenn Sie die Programme in IDLE ausführen).

Sys.argv beinhaltet als erstes den Dateinamen mit Pfad, und dann Befehlszeilenargumente.

Aufgabe Projekt TablePrint Seite 187 wiederholen indexing mit verschachtelten Schleifen und enumerate üben.

Zu wiederholde Projekte, Aufgaben, Seite Dateien handhaben 238

Widerholungsfragen Regex Seite 218

Regex Projekt Seite 220 wiederholen. Regex Projekt 248 wiederholen.

Ppritnt() „Saubere Ausgabe“ Kapitel 5 und Seite 236,

Zb. dictionarys die im Python-Format bleiben sollen mit pprint.pformat() z.B. in als eigenständige Datei, als Modul speichern wird als liste die das dictionary enthält gespeichert.

Der bist jetzt beste Weg ein **Skript aufzurufen, Bat datei** in einen Path-Ordner:   
@py.exe C:\Pfad\zu\IhremSkript.py **%\***

@pause

Geben Sie statt des Platzhalters den absoluten Pfad zu Ihrem Programm an und   
speichern Sie die Datei mit der Endung .bat. Das Zeichen @ zu Beginn der einzelnen   
Befehle verhindert, dass sie im Terminalfenster angezeigt werden, und die Zeichenfolge **%\*** sorgt dafür, dass alle Befehlszeilenargumente, die Sie hinter dem Namen   
der Batchdatei angeben, an das Python-Skript weitergeleitet werden. Das Skript   
wiederum liest diese Argumente in die Liste sys.argv ein.

Beispiel Raw-String über mehre Zeilen, „…\“darf nicht am ende der Zeile stehen

zipdatei = (r"F:\Backup-23-01-25-a\D-backup-2023-01-25\desk-ab-21\office-training"

r"\Python-ordner-a\python-scripts\_vs-code\sonstige\_projekte\python\_Ai\_Sweigert"

r"\Routine\_Aufgaben\_Automatisieren\neu.zip")

Projekt, Seite 212 **wiederholen RegEx**

## Pathlib, OS, Pfade usw.

os.path.**relpath**('C:\\Windows', 'C:\\spam\\eggs')

'..\\..\\Windows'

C:\Windows\System32\calc.exe  
Verzeichnisname \Grundname

Basisname eines Path-Objektes in pathlib = .name  
in OS: os.path.basename(path)

Verzeichnisname in pathlib:= path\_objekt.parent  
in OS: os.path.dirname(path)

Tupel(vezeinsname, basename) = os.path.split(path)  
Allerdings ist os.path.split() eine praktische Abkürzung.  
Beachten Sie aber, dass os.path.split() keinen Dateipfad entgegennimmt und   
daraus eine Liste der Strings für die einzelnen Ordner zurückgibt. Für diesen   
Zweck müssen Sie den Pfad mit der Stringmethode split() zerlegen und ihr die   
Variable **os.path.sep** übergeben, in der das Ordnertrennzeichen für das vorliegende   
Betriebssystem gespeichert ist.

Die Methode `Path.chmod()` in der `pathlib`-Bibliothek wird verwendet, um die Berechtigungen (Zugriffsrechte) einer Datei oder eines Verzeichnisses zu ändern. Diese Methode ist eine Entsprechung zu `os.chmod()` und ermöglicht es dir, die Zugriffsrechte auf Dateien und Verzeichnisse zu setzen, indem du den entsprechenden Berechtigungsmodus angibst.  
  
### Verwendung von `Path.chmod()`  
  
```python  
from pathlib import Path  
  
# Definiere den Pfad zur Datei  
path = Path('example.txt')  
  
# Ändere die Berechtigungen auf 0o644 (rw-r--r--)  
path.chmod(0o644)  
```  
  
### Parameter:  
  
- \*\*`mode`\*\*: Der Modus ist eine Ganzzahl, die die Berechtigungen in Oktalnotation angibt. Zum Beispiel:  
 - `0o644`: Lese- und Schreibrechte für den Eigentümer, nur Leserechte für andere.  
 - `0o755`: Lese-, Schreib- und Ausführungsrechte für den Eigentümer, Lese- und Ausführungsrechte für andere.  
 - `0o700`: Lese-, Schreib- und Ausführungsrechte nur für den Eigentümer.  
  
### Beispiel  
  
Hier ist ein vollständiges Beispiel zur Verwendung von `Path.chmod()`:  
  
```python  
from pathlib import Path  
  
# Erstelle eine Testdatei  
path = Path('example.txt')  
path.write\_text('Dies ist ein Test.')  
  
# Ändere die Berechtigungen der Datei  
print(f"Vorherige Berechtigungen: {oct(path.stat().st\_mode & 0o777)}")  
path.chmod(0o600) # Nur Lese- und Schreibrechte für den Eigentümer  
print(f"Nachherige Berechtigungen: {oct(path.stat().st\_mode & 0o777)}")  
```  
  
In diesem Beispiel:  
  
1. Eine Datei `example.txt` wird erstellt.  
2. Die Berechtigungen der Datei werden vor und nach der Verwendung von `chmod()` ausgedruckt.  
3. `0o600` setzt die Berechtigungen so, dass nur der Eigentümer Lese- und Schreibrechte hat.  
  
### Wichtige Punkte:  
  
- \*\*Zugriffsrechte\*\*: Die Berechtigungen werden in Oktalnotation angegeben. Zum Beispiel steht `0o644` für Lese- und Schreibrechte des Eigentümers und nur Leserechte für andere.  
- \*\*Berechtigungsänderungen\*\*: `chmod()` funktioniert nur bei Dateien und Verzeichnissen, die im Dateisystem existieren.  
- \*\*Plattformabhängigkeit\*\*: `chmod()` funktioniert in der Regel auf Unix-ähnlichen Systemen (wie Linux und macOS). Auf Windows-Systemen werden Berechtigungen anders gehandhabt, und `chmod()` hat möglicherweise eine eingeschränkte Wirkung.  
  
Die Methode `Path.chmod()` bietet eine objektorientierte Möglichkeit, Dateiberechtigungen zu ändern, und integriert sich nahtlos in die `pathlib`-Bibliothek, die für die Arbeit mit Pfaden und Dateien in Python entwickelt wurde.

pathlib bietet eine Methode zur Bestimmung der Größe einer Datei, die der Funktion os.path.getsize() entspricht.

In pathlib kannst du die Methode **.stat()** verwenden, um verschiedene Dateieigenschaften abzurufen, einschließlich der Größe der Datei. Die Größe wird im Attribut **st\_size** des os.stat\_result-Objekts gespeichert, das von .stat() zurückgegeben wird.

Seite 290 Debugging

* Das Modul **traceback**:

import traceback   
>>> try:   
 raise Exception('This is the error message.')   
except:   
 errorFile = open('errorInfo.txt', 'w')   
 errorFile.write(traceback.format\_exc())   
 errorFile.close()   
 print('The traceback info was written to errorInfo.txt.')   
# schreibt den traceback in die Datei “errorInfo.txt”

assert prüfung\_ob\_True, „Message“  
Zusicherungen(assert) lassen sich deaktivieren, indem Sie bei der Ausführung von Python die Option -0 übergeben

logging:

Marker Seite 278

Seite 294 Web Scraping

Wiederholen RegEx Seite 198