Wstęp do python

Paweł Gliwny

Czym jest venv w Pythonie?

- Narzędzie do tworzenia izolowanych środowisk Pythona.
- Każde środowisko ma własny runtime Pythona i zainstalowane pakiety.
- Przydatne do zarządzania zależnościami w różnych projektach.

Tworzenie środowiska venv w systemie Windows

- Otwórz wiersz poleceń (Command Prompt).
- Przejdź do wybranego katalogu używając polecenia cd.
- Uruchom `python -m venv nazwa_środowiska`.
- Aktywuj środowisko poleceniem: `nazwa_środowiska\Scripts\activate`.
- Zainstaluj pakiety używając pip w środowisku.

pip install numpy

pip install psutil

`psutil` (Python system and process utilities)

Wieloplatformowa biblioteka w Pythonie do monitorowania systemu i procesów. Oferuje:

1. **Monitorowanie CPU**: Pozwala na sprawdzanie ogólnego użycia CPU oraz użycia poszczególnych rdzeni.
2. **Pamięć**: Dostarcza informacji o użyciu pamięci RAM i wirtualnej.
3. **Dyski**: Umożliwia sprawdzanie informacji o dyskach, w tym przestrzeni i użycia.
4. **Sieć**: Dostarcza danych o sieci, w tym statystyk używania i adresów IP.
5. **Zarządzanie Procesami**: Umożliwia uzyskanie informacji o procesach i ich zarządzanie, np. zabijanie procesów.
6. **Informacje o Systemie**: Obejmuje czas działania systemu, zalogowanych użytkowników itp.

Jest to narzędzie użyteczne w monitorowaniu wydajności, zarządzaniu systemem i automatyzacji zadań systemowych.

Zmienne w Python

Zmienne są używane do przechowywania danych, które mogą być manipulowane w trakcie wykonywania programu:

String: >> s = “Witaj, świecie”

Integer >> i = 42

Float >> f = 3.14

Boolean: True / False

Listy w Python

Lista to dynamiczna kolekcja, która może przechowywać elementy różnych typów danych.

```
>> moja_lista = [1, 2, 3, "Python", True]

>> moja_lista.append(42) # Dodawanie elementów

>> moja_lista.remove(2)

for element in moja_lista:
    print(element)
```

Biblioteka os

Biblioteka `os` w Pythonie jest standardowym modułem, który dostarcza funkcje do interakcji z systemem operacyjnym.

- **Zarządzanie ścieżkami plików**: Moduł `os` zawiera funkcje do pracy ze ścieżkami plików, takie jak `os.path.join`, `os.path.split` i `os.path.exists`, które pozwalają na łączenie, dzielenie i sprawdzanie istnienia ścieżek plików.
- **Zarządzanie plikami i katalogami**: Dzięki `os`, można tworzyć, usuwać, przenosić pliki i katalogi. Funkcje takie jak `os.mkdir`, `os.rmdir`, `os.rename` i `os.remove` są do tego wykorzystywane.
- **Wykonywanie poleceń systemowych**: Za pomocą `os.system` można wykonywać polecenia systemowe z poziomu skryptu Pythona.
- **Dostęp do zmiennych środowiskowych**: Moduł `os` umożliwia odczytywanie i ustawianie zmiennych środowiskowych przez funkcje takie jak `os.environ`, `os.getenv` i `os.putenv`.

Przykład

- Funkcja `listdir` w module `os` w Pythonie jest używana do uzyskania listy nazw plików i katalogów zawartych w określonym katalogu.
- Wynikiem jest lista zawierająca nazwy plików i katalogów w określonym katalogu.

```
>>> import os

>>> path = “C:\Program Files\Mozilla Firefox”

>>> files_mozilla_list = os.listdir(path)

>>> for f in files_mozilla_list:

        print(f)
```

Słownik

Słownik to nieuporządkowana kolekcja danych przechowywana w parach klucz-wartość.

Słowniki oferują wydajny i elastyczny sposób przechowywania danych w parach klucz-wartość, co ułatwia organizację i manipulację danymi.

```
>> moj_slownik = {"imie": "Anna", "wiek": 30, "zawod": "programista"}

>> moj_slownik[“jezyk”] = “Python” # Dodawanie elementów

>> del moj_slownik[“wiek”] # Usuwanie elementów

>> print(moj_slownik[“imie”])

for klucz, wartosc in moj_slownik.items():

        print(klucz, wartosc)
```

Otwieranie plików tekstowych

Python oferuje wbudowane funkcje do otwierania, czytania, zapisywania i manipulowania plikami tekstowymi.

Kroki do Otwarcia Pliku:

1. **Otwórz Plik:**
   - Użyj funkcji `open()`, aby otworzyć plik tekstowy.
   - Przykład: `file = open('nazwapliku.txt', 'r')`
2. **Czytaj Plik:**
   - Metody jak `read()`, `readline()` lub `readlines()` do czytania zawartości.
   - Przykład: `content = file.read()`
3. **Zamknij Plik:**
   - Zawsze pamiętaj, aby zamknąć plik po zakończeniu pracy.
   - Przykład: `file.close()`

Alternatywna Metoda - `with` Statement:

- Automatycznie zamyka plik po zakończeniu bloku kodu

```
with open('nazwapliku.txt', 'r') as file:
    content = file.read()
```

Zapisywanie do pliku w pythonie

Python umożliwia łatwe i intuicyjne zapisywanie danych w plikach tekstowych. Można to zrobić w kilku prostych krokach, używając wbudowanych funkcji i metod.

Otwórz lub Utwórz Plik:

- Użyj funkcji `open()` z trybem `'w'` do zapisu lub `'a'` do dodawania tekstu do istniejącego pliku.

  file = open(‘plik.txt’, ‘w’)

Zapisz Dane:

- Użyj metody `write()` do zapisania tekstu w pliku.

  file.write(“Witaj, świecie!”)

Zamknij Plik:

- Zawsze zamknij plik, używając metody `close()`, aby upewnić się, że wszystkie dane zostały zapisane.

  file.close()

Zapisywanie z ‘with’ statement

- Zaleca się użycie instrukcji `with`, która automatycznie zamyka plik po zakończeniu bloku kodu. Jest to bardziej efektywne i bezpieczne podejście.

```
with open('plik.txt', 'w') as file:
    file.write('Witaj, świecie!')
```

- Aby dodać tekst do istniejącego pliku, otwórz plik w trybie `'a'`.

```
with open('plik.txt', 'a') as file:
    file.write('\nDodatkowy tekst.')
```

Metoda `str.split()` and `str.strip()`

str.split() służy do dzielenia łańcucha znaków na listę podciągów na podstawie określonego separatora.

```
str.split([separator[, maxsplit]])
```

- separator: Opcjonalny. Znak, według którego dzielony jest łańcuch. Domyślnie jest to biała spacja.
- maxsplit: Opcjonalny. Maksymalna liczba podziałów. Domyślnie jest to -1, co oznacza "wszystkie wystąpienia".

str.strip() służy do usuwanie początkowych i końcowych białych znaków z łańcucha znaków.

```
str.strip([chars])
```

- Jeśli podano argument chars, metoda usuwa wszystkie kombinacje tych znaków z początku i końca łańcucha.
- W praktyce metoda `str.strip()` jest często używana do czyszczenia danych wejściowych, takich jak tekst wprowadzony przez użytkownika lub odczytany z pliku, z niepotrzebnych białych znaków lub innych niechcianych znaków na początku i końcu łańcucha.

```
text = "Hello, World!"


split_text = text.split()
print(split_text)  # ['Hello,', 'World!']


split_text = text.split(',')
print(split_text)  # ['Hello', ' World!']


strip_text = text.strip('!')
print(strip_text)  # ‘Hello, World’
```

Funkcje w python

Funkcje to zorganizowane bloki kodu, które można wielokrotnie wywoływać. Umożliwiają one modularność i ponowne użycie kodu, co sprawia, że programy są bardziej zorganizowane i łatwiejsze do zarządzania.

Zalety używania funkcji:

- Modularność: Podział kodu na mniejsze, zarządzalne części.
- Ponowne użycie: Możliwość wielokrotnego wywoływania tego samego bloku kodu.
- Łatwość w utrzymaniu: Ułatwiają zarządzanie i aktualizację kodu.

```
def powitanie():
    print("Witaj w świecie Pythona!")
```

Definiowanie i użycie funkcji

1. Użyj słowa kluczowego `def`, aby zadeklarować funkcję.
2. Dodaj nazwę funkcji, a następnie nawiasy `( )` zawierające parametry.
3. Umieść kod funkcji wewnątrz bloku wciętego.

```
# Definiowanie funkcji
def dodaj(a, b):
    suma = a + b
    print(f"Suma {a} i {b} wynosi {suma}")

# Wywoływanie funkcji z argumentami
dodaj(5, 7)
```

Moduły w python

Moduły w Pythonie to pliki zawierające zestaw funkcji, zmiennych i klas, które można zaimportować do innego programu. Umożliwiają organizację i modularność kodu, ułatwiając jego zarządzanie i rozwijanie.

Zalety używania modułów:

- **Organizacja:** Pomagają w organizacji i strukturyzacji kodu.
- **Ponowne użycie:** Umożliwiają wielokrotne użycie kodu w różnych projektach.
- **Przestrzeń nazw:** Pomagają w zarządzaniu przestrzenią nazw, izolując różne części kodu.

```
# Plik mymodule.py
def powitanie():
    print("Witaj w module Pythona!")
```

Funkcja lambda

Kiedy potrzeba bardzo prostej funkcji, możemy utworzyć funkcje bez nazwy (anonimową) za pomocą wyrażenia lambda:

lambda <PARAMETR>: <ZWRACANE WYRAZENIE>

```
>>> items = [[0, ‘a’, 2], [2, ‘c’, 3], [1, ‘b’, 4]]

>>> sorted(items)

def second(item):

    return item[1]

>>> sorted(items, key=second)

>>>sorted(items, key=lambda item: item[1])
```

Użycie modułów

Importowanie modułu:

1. Użyj słowa kluczowego `import`, aby zaimportować moduł.
2. Opcjonalnie, użyj `as` do nadania modułowi aliasu.

Wywoływanie funkcji z modułu:

- Użyj nazwy modułu, kropki i nazwy funkcji.

```python
# Plik mymodule.py
def powitanie():
    print("Witaj w module Pythona!")
```

```
from mymodule import powitanie

# Wywoływanie funkcji
powitanie()

lub

# Importowanie modułu
import mymodule

# Wywoływanie funkcji z modułu
mymodule.powitanie()
```

Biblioteka standardowa

- Zestaw modułów dostarczanych wraz z każdą instalacją Pythona, które dostarczają funkcji, klas i typów służących do realizacji różnorodnych zadań, bez konieczności instalowania dodatkowych pakietów lub bibliotek.
- Możemy szybko i łatwo tworzyć różne aplikacje bez konieczności wynajdywania koła na nowo.

- **math:** podstawowe funkcje matematyczne
- **datetime:** operacje na datach i czasie
- **os:** interakcje z systemem operacyjnym
- **re:** wyrażenia regularne
- **json:** obsługa formatu JSON
- **sqlite3:** obsługa bazy danych SQLite

Moduł os.path

os.path.join

- Łączy części ścieżek w jedną, zgodnie z konwencjami systemowymi.
- Automatycznie dostosowuje znaki rozdzielające ścieżki.
- Przykład: `os.path.join("/home/user", "dokument.txt")` → `/home/user/dokument.txt`

os.path.isfile

- Sprawdza, czy ścieżka jest plikiem.
- Zwraca `True` dla plików, `False` dla innych (np. katalogów).
- Przykład: `os.path.isfile("ścieżka/do/pliku")`

os.path.getsize

- Zwraca rozmiar pliku w bajtach.
- Wymaga istniejącej ścieżki do pliku.
- Przykład: `os.path.getsize("ścieżka/do/pliku")` → rozmiar w bajtach

Collections

Moduł z biblioteki standardowej, który dostarcza alternatywne wersje wbudowanych kontenerów (takich jak listy, krotki, słowniki itp.) oraz nowe typy kontenerów, które są bardziej specjalizowane i mogą być przydatne w różnych sytuacjach programistycznych.

Counter: Jest to słownik, który pomaga liczyć wystąpienia elementów.

```
from collections import Counter,


c = Counter("abracadabra")
print(c)
# wypisze Counter({'a': 5, 'b': 2,
'r': 2, 'c': 1, 'd': 1})
```

Moduł `datetime`

- Biblioteka do pracy z datami i czasami.
- Pozwala na manipulowanie datami, czasami i ich połączeniami, uwzględniając strefy czasowe.
- Umożliwia arytmetykę dat i czasów, formatowanie, parsowanie oraz obliczanie różnic czasowych.
- Intuicyjny i użyteczny w wielu zastosowaniach, jak logowanie, planowanie czy analiza danych czasowych.

```
from datetime import datetime

current_datetime = datetime.now()

formatted_datetime = current_datetime.strftime("%Y-%m-%d %H:%M:%S")

print(formatted_datetime)
```

Mierzenie czasu wykonywania kodu

W Pythonie, aby zmierzyć czas wykonania programu, możesz użyć modułu `time`.

```
import time
start_time = time.time()
… kod
end_time = time.time()
elapsed_time = end_time - start_time
print(f"Czas wykonania programu: {elapsed_time} sekund.")
```

pętla for vs operacje wektorowe w numpy

```
suma = 0
for i in range(0, 1000000):
        suma += 1


import numpy as np
>>> np.arange(1000000).sum()
```

Zadanie 1

Napisz kod do pomiaru czasu wykonania programu za pomocą pętli i numpy:

- sumowanie liczb od 0 do 1 000 000

Zadanie 2

Wczytaj plik distros.txt

a) policz liczbę linii
b) Stwórz słownik **linux_distribution_dict** którego kluczem będzie nazwa i wersja Linuksa, a wartością rok wydania

{'SUSE 10.2': '12/07/2006 ',
'Fedora 10': '11/25/2008 ',
'SUSE 11.0': '06/18/2008',
…}

Zadanie 3: Operacje na plikach i katalogach z użyciem biblioteki os

Jako administrator systemu, zautomatyzuj proces zarządzania plikami i katalogami. Użyj biblioteki os w Pythonie do:

1. Utworzenia katalogu "Raporty".
2. Wygeneruj 5 plików tekstowych w "Raporty" z bieżącą datą i godziną oraz użyciem cpu z ostatnich 5 sekund .
3. Stwórz plik "podsumowanie.txt" w którym będzie połączona informacje z plików **raport[1-5].txt:**

   w postaci:

   nazwa pliku, data i godzina, użycie cpu.

   aby uzyskać nazwę pliku wylistować zawartość katalogu “Raporty”

4. Stworzenie nowego katalogu w miejscu gdzie się pracuje o nazwie kosz i przeniesienie tam plików **raport[1-5].txt**

Wskazówki:

- Użyj funkcji jak `os.mkdir()`, `os.rename()`, `os.listdir()`, ….
- Wykorzystaj moduł `datetime` do zapisu bieżącej daty i godziny.

Zadanie 4

Wylistuj nazwy plików wraz z ich rozszerzeniami z folderu 'C:\Program Files\Mozilla Firefox'.

Następnie, używając klasy Counter z modułu collections, policz, ile jest unikalnych rozszerzeń plików w tym folderze.

Zadanie 5

Stwórz słownik, gdzie kluczem będzie nazwa pliku z danego katalogu, a wartością rozmiar tego pliku.

Następnie posortuj ten słownik od największego do najmniejszego