Zaawansowane programowanie w Pythonie

Wykład 5

Agnieszka Zbrzezny

KMMI WMII UWM

14 listopada 2024

Moduł sys

Wybrane zmienne

- `sys.argv`
- `sys.byteorder`
- `sys.copyright`
- `sys.float_info`
- `sys.int_info`
- `sys.modules`
- `sys.path`
- `sys.platform`
- `sys.prefix`
- `sys.ps1` – tylko w interpreterze
- `sys.ps2` – tylko w interpreterze

Moduł sys

Wybrane funkcje

- `sys.exit([status])`
- `sys.getdefaultencoding()`
- `sys.getfilesystemencoding()`
- `sys.getrecursionlimit()`
- `sys.getrefcount(object)`
- `sys.getsizeof(object[, default])`

Przykłady

- Program `sys-p00.py`
- Programy `sys-p01.py`, `sys-p02.py`, `sys-p03.py`

Moduł os

Uwagi ogólne

- Moduł `os` stanowi opakowanie dla modułów specyficznych dla danej platformy, takich jak `posix`, `nt` i `mac`.
- Interfejs API funkcji dostępnych na wszystkich platformach powinien być taki sam, więc używanie modułu `os` zapewnia pewną przenośność.
- Nie wszystkie funkcje są jednak dostępne na każdej platformie. Wiele z funkcji zarządzania procesami nie jest dostępnych dla systemu Windows.
- Dokumentacja modułu `os` w języku Python nosi podtytuł „Różne interfejsy systemu operacyjnego”.
- Moduł składa się głównie z funkcji do tworzenia i zarządzania uruchomionymi procesami oraz zawartością systemu plików (plikami i katalogami), a także z kilku innych funkcji.

Moduł os

Przykłady (Wybrane funkcje modułu os)

- Program os-p01.py – os.getcwd, os.chdir
- Program os-p02.py – os.access
- Program os-p03.py – os.symlink, os.readlink, os.unlink
- Program os-p04.py – os.stat
- Program os-p05.py – os.makedirs, os.listdir, os.rmdir
- Program os-p06.py – os.chmod
- Program os-p07.py – os.walk

Praca z systemem plików

Wprowadzenie

- Praca z systemem plików obejmuje między innymi:
  - obsługę ścieżek i nazw,
  - uzyskiwanie informacji o plikach,
  - dokonywanie operacji na systemie plików,
  - przetwarzanie wszystkich plików w podkatalogach.
- Tradycyjnym sposobem obsługi ścieżek plików i operacji na systemie plików w Pythonie jest korzystanie z funkcji zawartych w modułach `os` oraz `os.path`.
- Funkcje te sprawdzają się świetnie, ale często wymagały pisania większej ilości kodu, niż można by przypuszczać.
- Począwszy od Pythona 3.5, mamy do dyspozycji nową bibliotekę, `pathlib`, która udostępnia te same operacje, ale w sposób bardziej jednolity i zorientowany obiektowo niż moduły wspomniane wcześniej.

Praca z systemem plików

Wprowadzenie

- Nadal jednak można spotkać wiele kodu korzystającego z `os` oraz `os.path`, więc zajmiemy się przede wszystkim tymi modułami.
- Z drugiej jednak strony, biblioteka `pathlib` ma wiele do zaoferowania i jest wysoce prawdopodobne, że stanie się nowym standardem.
- Z tego powodu, zaprezentujemy także wybrane przykłady z wykorzystaniem `pathlib`.

Praca z systemem plików

Ścieżki i nazwy ścieżek

- Wszystkie systemy operacyjne odnoszą się do plików i katalogów za pomocą łańcucha znaków nazywającego dany plik czy katalog.
- Łańcuchy znaków używane w tym celu znane są jako nazwy ścieżek (lub krócej ścieżki) – i tego właśnie terminu będziemy używać.
- Fakt, że nazwy ścieżek są łańcuchami znaków, rodzi pewne trudności w pracy z nimi.
- Python dobrze radzi sobie jednak z unikaniem tych trudności, ale by sprawnie korzystać z jego funkcji, powinieneś zrozumieć, skąd mogą się one brać.

Praca z systemem plików

Ścieżki i nazwy ścieżek

- Semantyka nazw ścieżek jest bardzo podobna we wszystkich systemach operacyjnych, ponieważ systemy plików na niemal wszystkich platformach modelowane są jako struktura drzewiasta, z dyskiem u korzenia, katalogami i podkatalogami w charakterze gałęzi czy podgałęzi itd.
- Oznacza to, że większość systemów operacyjnych odnosi się do plików w zasadniczo taki sam sposób.
- Nazwa ścieżki określa drogę od korzenia drzewa systemu plików (dysku) aż po konkretny plik (to przyrównanie korzenia do dysku twardego stanowi uproszczenie, ale jest wystarczająco bliskie prawdy na potrzeby tego omówienia).
- Nazwa ścieżki zawiera więc serię katalogów, do których należy wejść, by wreszcie dostać się do pożądanego pliku.

Praca z systemem plików

Ścieżki i nazwy ścieżek

- Różne systemy operacyjne mają jednak różne konwencje odnośnie dokładnej składni nazw ścieżek.
- Znakiem oddzielającym nazwy katalogów czy katalogu i pliku w systemach `UNIX/Linux/macOS` jest `/`, podczas gdy nazwy ścieżek w systemach Windows używają znaku `\` jako separatora.
- Co więcej, system plików UNIX-a ma jeden korzeń, do którego odnosimy się poprzez pojedynczy znak `/` na początku nazwy ścieżki, a system plików w Windowsie ma osobny korzeń dla każdego dysku, oznaczonego `A:\`, `B:\`, `C:\` (który stanowi zwykle główny dysk) itd.
- Z powodu tych różnic nazwy ścieżek mają różne reprezentacje w zależności od systemu operacyjnego.

Praca z systemem plików

Ścieżki i nazwy ścieżek

- Plik o nazwie `C:\data\mójplik` w MS Windows może być nazwany `/data/mójplik` w systemach UNIX czy macOS.
- Python udostępnia funkcje i stałe, które pozwalają wykonywać powszechne operacje na nazwach ścieżek bez konieczności martwienia się o składniowe szczegóły.
- Przy odrobinie uwagi można pisać swoje programy tak, że będą poprawnie wykonywały się niezależnie od systemu plików, z jakim pracują.

Praca z systemem plików

Ścieżki bezwzględne i względne

- Bezwzględne (absolutne) nazwy ścieżek określają dokładną lokalizację pliku w systemie plików bez jakichkolwiek niedomówień. Osiągają to poprzez prezentowanie całej ścieżki do pliku, począwszy od korzenia systemu plików.
- Względne (relatywne) nazwy ścieżek określają położenie pliku w odniesieniu do jakiegoś innego miejsca w systemie plików, a to inne miejsce nie jest wskazane przez samą ścieżkę względną. Bezwzględny punkt początkowy względnej nazwy ścieżki dostarczany jest poprzez kontekst swojego użycia.

Moduł `os.path`

Funkcja `join`

- Funkcja `os.path.join` interpretuje swoje argumenty jako serię nazw katalogów lub plików, które ma połączyć w jeden łańcuch znaków zrozumiały dla systemu operacyjnego jako ścieżka względna.
- Na komputerach z Windows oznacza to ścieżkę oddzieloną odwróconymi ukośnikami.
- Na platformie UNIX wynikiem jest taka sama ścieżka, ale używająca konwencji Linux/UNIX odnośnie separatora w postaci ukośnika zamiast konwencji znanej z Windowsa.
- Innymi słowy, funkcja `os.path.join` pozwala tworzyć ścieżki plików na podstawie sekwencji nazw katalogów i plików bez konieczności martwienia się o wymagania systemu operacyjnego.

Moduł `os.path`

Funkcja `join`

- Jest to funkcja o fundamentalnym znaczeniu dla możliwości tworzenia nazw ścieżek w sposób pozbawiony ograniczeń bieżącego systemu operacyjnego.
- Argumentami funkcji `os.path.join` nie muszą być pojedyncze nazwy katalogów czy plików. Można korzystać z fragmentów ścieżek łączonych w dłuższe nazwy.
- Polecenie `os.path.join` rozróżnia również ścieżki względne i bezwzględne.
- W systemach Linux/UNIX ścieżka bezwzględna zawsze zaczyna się od znaku `/`, ponieważ pojedynczy ukośnik oznacza podstawowy katalog całego drzewa katalogów w systemie.
- Ścieżką względną w tych systemach jest każda poprawna ścieżka nierozpoczynająca się od ukośnika.

Moduł `os.path`

Funkcja `join`

- W każdym z systemów Windows sytuacja jest bardziej złożona, ponieważ sposób, w jaki Windows obsługuje ścieżki względne i bezwzględne, jest tam bardziej zagmatwany.
- Niezależnie od używanego systemu operacyjnego, funkcja `os.path.join` nie będzie sprawdzać poprawności ścieżki.
- Przy jej użyciu możliwe jest stworzenie ścieżek, które będą zawierały w sobie znaki niedopuszczalne w nazwach ścieżek danego systemu operacyjnego.
- Jeżeli chcemy tego uniknąć, najprawdopodobniej najlepszym rozwiązaniem będzie napisanie samemu niewielkiej funkcji walidującej.

Moduł os.path

Przykłady (Wybrane zmienne i funkcje modułu os.path)

- Program os-p08.py – zmienne
- Program os-p09.py – funkcje split, splitext, abspath
- Program os-p10.py – funkcje basename, commonprefix, realpath
- Program os-p11.py – funkcje isdir, getsize, exists, dirname, normpath, expandvars, expanduser
- Program os-p12.py – funkcje ismount, isabs, relpath, islink, samefile