Wyjątki, venv, operacje na plikach

Wyjątki

Wyjątki:

- jeżeli w czasie pracy programu wystąpi błąd, interpreter ‘wyrzuca’ wyjątek,
- jeżeli wyjątek nie będzie obsłużony, działanie programu zostanie przerwane.

Przykład 1:

```
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> slownik={}
>>> slownik['test']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'test'
>>> lista=[]
>>> lista[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
```

Wyjątki

Obsługa wyjątków:

- w Pythonie wyjątki obsługuje się przy pomocy instrukcji ‘try’ i ‘except’.

Przykład 2 - obsługa jednego wyjątku:

```
def divide(x,y):
    try:
        result=x/y
    except ZeroDivisionError:
        print(f'Wyjatek, dzielenie przez zero: {x}/{y}!')
    else:
        print(f'Brak wyjatku ZeroDivisionError, wynik: {result:.3}')
    finally:
        print('To zawsze się wykona')
```

```
#output:

Brak wyjatku ZeroDivisionError, wynik: 0.667
To zawsze się wykona
Wyjatek, dzielenie przez zero: 2/0!
To zawsze się wykona
```

Wyjątki

Przykład 3 - obsługa kilku wyjątków:

```
def divide(x,y):
    try:
        result=x/y
    except (ZeroDivisionError, TypeError) as ex:
        print(f'Wyjatek {ex}')
    else:
        print(f'Brak wyjatku, wynik: {result:.3}')
    finally:
        print('To zawsze się wykona')


divide(2,3)
divide(2,0)
divide(2,'2')
```

```
#output:
Brak wyjatku, wynik: 0.667
To zawsze się wykona
Wyjatek division by zero
To zawsze się wykona
Wyjatek unsupported operand type(s) for /: 'int' and 'str'
To zawsze się wykona
```

Wyjątki

Przykład 3a - obsługa kilku wyjątków:

```
def divide(x,y):
    try:
        result=x/y
    except ZeroDivisionError as ex:
        print(f'Wyjatek {ex}')
    except TypeError as ex:
        print(f'Wyjatek {ex}')
    else:
        print(f'Brak wyjatku, wynik: {result:.3}')
    finally:
        print('To zawsze się wykona')

divide(2,3)
divide(2,0)
divide(2,'2')
```

```
#output:
Brak wyjatku, wynik: 0.667
To zawsze się wykona
Wyjatek division by zero
To zawsze się wykona
Wyjatek unsupported operand type(s) for /: 'int' and 'str'
To zawsze się wykona
```

Tworzenie i generowanie własnego wyjątku

Przykład 4:

```
class MojWyjatek(Exception):   #definicja – musi dziedziczyć po wbd. klasie Exception
    pass

try:
    raise MojWyjatek('MojError')  #w nawiasie opcjonalny parametr wyjatku
except MojWyjatek as ex:
    print(ex)
```

```
#output:
MojError
```

venv

venv (Virtual Environment):

- jest modułem/programem, który umożliwia utworzenie środowiska dla języka Python z wybranymi wersjami bibliotek, obok ich standardowej instalacji w systemie operacyjnym,
- każdy użytkownik systemu operacyjnego może mieć wiele różnych środowisk – zalecane rozwiązanie to tworzenie oddzielnego środowiska dla każdego projektu,
- jest standardowo dostępny w Python'ie 3.3 i nowszych,
- w zależności od wersji tworzone środowisko może mieć nieco inną konfigurację czy inny zestaw pakietów startowych, np. od wersji 3.4 do środowiska domyślnie dokładany jest manager pakietów 'pip'.

venv

Przykład 5a – tworzenie środowiska :

```
#tworzenie środowiska
D:\tmp>py -m venv myenv

# katalog  środowiska:
D:\tmp\myenv>

# aktywacja środowiska:
D:\tmp\myenv>Scripts\activate

(myenv) D:\tmp\myenv>
```

venv

Przykład 5b – tworzenie środowiska:

```
# teraz można doinstalować pakiet potrzebny
# dla konkretnego projektu, np.:

(myenv) D:\tmp\myenv>pip install numpy
Collecting numpy
  Downloading
https://files.pythonhosted.org/packages/ed/29/d97b6252591da5f8add0
d25eecda296ea72729a0aad7998edba1981b47c8/numpy-1.16.2-cp36-cp36m-
win_amd64.whl (
11.9MB)
    100% |████████████████████████████████████████| 11.9MB 140kB/s
Installing collected packages: numpy
Successfully installed numpy-1.16.2

(myenv) D:\tmp\myenv>
 ...
```

venv

Przykład 5c: numpy_example.py

```
# Create 2 new lists height and weight
height = [1.87,  1.87, 1.82, 1.91, 1.90, 1.85]
weight = [81.65, 97.52, 95.25, 92.98, 86.18, 88.45]

import numpy as np

# Create 2 numpy arrays from height and weight
np_height = np.array(height)
np_weight = np.array(weight)

#bmi calculation
bmi=np_weight / np_height ** 2

#results
print(bmi)
```

venv

Przykład 5d – tworzenie środowiska c.d.:

```
 …
#uruchamianie przykładowego skryptu
(myenv) D:\tmp\myenv>py numpy_example.py
[23.34925219 27.88755755 28.75558507 25.48723993 23.87257618 25.84368152]
[27.88755755 28.75558507]
[179.63  214.544 209.55  204.556 189.596 194.59 ]

(myenv) D:\tmp\myenv>
# po zakończeniu pracy:
(myenv) D:\tmp\myenv>Scripts\deactivate.bat
D:\tmp\myenv>
```

virtualenv

virtualenv:

- jest alternatywnym modułem do tworzenia środowiska,
- działa z Python'em w wersjach 2.7+ oraz 3.3+,
- w stosunku do 'venv' tworzy bardziej rozbudowane środowisko,
- wymaga osobnej instalacji : py –m pip install virtualenv.

Moduły glob, os, zipfile, tarfile

Moduły glob, os, zipfile, tarfile:

- **glob** - umożliwia wykorzystanie wzorców ścieżek do plików w sposób znany z powłok systemów unixowych,
- **os** – udostępnia szereg funkcji związanych z bezpośrednią interakcją z systemem operacyjnym.
- **zipfile**, **tarfile** – udostępniają funkcje do tworzenia archiwów, kompresji i ekstrakcji plików z poziomu skryptu.

Moduł glob i os

Przykład 6:

```
import os, glob

path=os.getenv('SystemRoot') + '/*.*[it]'    #wzorzec
print(glob.glob(path))       #glob zwraca listę
for p in glob.iglob(path):    #iglob zwraca generator (iterator)
    print(p)
    #print(os.path.normpath(p))
```

```
#output:
['c:/Windows\\pyshellext.amd64.dll', 'c:/Windows\\twain.dll', 'c:/Windows\\twain_32.dll']
c:/Windows\bootstat.dat
c:/Windows\brmx2001.ini
…
```

Moduł zipfile

Przykład 7:

```
import zipfile

#tworzenie
try:
    fzip=zipfile.ZipFile('tmp/my.zip','x')
    fzip.write('p6.py')
    fzip.close()
except FileExistsError:
    print('Archiwum już istnieje!')
#listing
fzip=zipfile.ZipFile('tmp/my.zip','r')
for f in fzip.namelist():
    print(f)
#ekstrakcja
fzip.extract(fzip.namelist()[0],'tmp')
fzip.close()
```

```
#output:
p6.py
```

Zadanie 1

Napisz funkcje, która pobiera liczbę i wyświetla jej pierwiastek kwadratowy. Obsłuż wszystkie wyjątki, które mogę wystąpić w wyniku działania programu.

Zadanie 2

Napisz klasę, której obiekt będzie przechowywać adres e-mail. Konstruktor ma przyjmować napis, będący adresem. Jeśli zostanie podany niewłaściwy adres, konstruktor ma zgłaszać wyjątek odpowiedniej klasy (własnej klasy). Walidacja adresu ma być realizowana przy pomocy wyrażeń regularnych.

Przykład:

a=adres('j.kowalski.@gmail.com') generuje wyjątek, b=adres('j.kowalski@gmail.com') jest OK.

Zadanie 3

Utwórz za pomocą ‘venv’ własne środowisko. Doinstaluj do niego pakiet ‘pycodestyle’ i sprawdź zgodność przykładowego skryptu ze standardem ‘pep8’ (jest to standard określający zalecany sposób pisania/formatowania kodu).

Zadanie wykonaj z linii komend.

Zadanie 4

Napisz skrypt, który wykorzystuje moduł 'tarfile' i testuje skuteczność dostępnych w tym module algorytmów kompresji. Program powinien wyświetlić informacje o liczbie plików i ich sumarycznym rozmiarze przed kompresją oraz rozmiary archiwów po kompresji (taz, gz, gz2, xz).

Dodatkowe warunki działania programu to:

- program powinien utworzyć katalog testowy ('testy') w katalogu domowym użytkownika, którego ścieżka powinna być pobrana ze zmiennych środowiskowych systemu,
- testowe pliki do kompresji powinny być pobrane z katalogu Python'a (bez podkatalogów),
- program powinien być zabezpieczony przed przypadkowym nadpisaniem jakichkolwiek plików.

Zadanie 4, cd.

Przykład (wyjście programu):

```
Test dir:
  C:\Users\mwisniew\testy
Python's dir:
  C:/Users/mwisniew/AppData/Local/Programs/Python/Python38/
Liczba plików: 18, rozmiar: 5682097 B
Tworzenie:
   C:\Users\mwisniew\testy\ppath.tar x: 0
Tworzenie:
   C:\Users\mwisniew\testy\ppath.tar.gz x:gz 0
Tworzenie:
   C:\Users\mwisniew\testy\ppath.tar.bz2 x:bz2 0
Tworzenie:
   C:\Users\mwisniew\testy\ppath.tar.xz x:xz 0
Wyniki:
  tryb:  x: ; size:  5580800 B
  tryb:  x:gz ; size:  2394751 B
  tryb:  x:bz2 ; size:  2228208 B
  tryb:  x:xz ; size:  1816448 B
```