Moduly a balíčky

- Vhodné dělit funkcionalitu do více souborů, případně i vytvářet knihovny a sdílet jejich funkcionalitu
- Tyto soubory je pak možné importovat a využívat jejich funkcionalitu i jinde
- Modul má vlastní *jmenný rozsah* => je možné využívat 2 stejnojmenné funkce z různých modulů

  [!info] **modul** … soubor obsahující Python definice a příkazy **název modulu** … název souboru s příponou `.py`

```
# ukázka importování vestavěného modulu "math"
import math
import my_math

math.sqrt(6)

# získá název modulu
math.__name__

# stejný název, jiný modul
math.sqrt
my_math.sqrt
```

Postup vytvoření vlastního modulu

1) Vytvoříme soubor `my_module.py`
2) Jeho obsahem budou funkce, které chceme použít i jinde
3) Poté v jiném skriptu umístěném ve stejné složce modul importujeme

```
import my_module

my_module.my_function(1, "var2")
```

PEP8 okénko

```
# PEP8 - jednotlivé importy musí být na samostatném řádku
import os
import sys

# PEP8 - pořadí importů
import sys      # systémová knihovna
import numpy    # externí knihovna
import my_math  # lokální modul
```

- Je možné importovat jména z jmenného prostoru modulu do aktuálního

```python
from list_operations import subtract_lists, sum_lists

subtract_lists([1, 2], [4, 3])
```

- Taktéž je možné importovat takto všechna jména (nedoporučuje se používat) či modul přejmenovat (případně přejmenovat jednotlivé funkce)

```python
from list_operations import *

import list_operations as loperations

from list_operations import subtract_lists as lsubtract

# přidání této části Kódu zajistí vykonání obsahu pouze při přímem spuštění zdrojového kódu

if __name__ == "__main__":
    print("Only when script is launched.")
```

Odkud se moduly importují?

1) Interpret začíná hledáním modulů vestavěných
2) V případě neúspěchu hledá soubor s daným název `modul.py` v seznamu adresářů uložené v proměnné `sys.path`

- Seznam obsahuje adresář odkud byl skript spuštěn, `PYTHONPATH` s cestami k instalovaným modulů a další

Balíčky

- Způsob jak strukturovat jmenný prostor Python modulu

```python
# jmenný prostor přístupný skrz tečkovou notaci

import A.B
```

Příklad struktury balíčku

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
```

```
            __init__.py
            echo.py
            surround.py
            reverse.py
            ...
    filters/                Subpackage for filters
            __init__.py
            equalizer.py
            vocoder.py
            karaoke.py
            ...
```

- Speciální soubor `__init__.py` zajišťuje že Python bude složku považovat za balíček
- Pro inicializaci stačí prázdný soubor, případně může obsahovat inicializaci balíčku
- Import pak probíhá následujícím způsobem

```
import sound.effects.echo


# aby mohl být modul použít, musí být napsáno jeho plné jméno
sound.effects.echo.echofilter(input, output, delay=0.7, atten=4)

# ALTERNATIVA
from sound.effects import echo


echo.echofilter(input, output, delay=0.7, atten=4)
```

- Importy lze provádět i v rámci balíčku

```
# v rámci složky
from . import echo
# nadřazená složka
from .. import formats
from ..filters import equalizer
```

Instalace externích balíčků

- Většinou instalujeme externě z Python Package Index (PyPI) pomocí nástroje `pip`
- Názorná ukázka instalace

```
# UNIX
$ python3 -m pip install pytest

# Windows
```

```
$ py -m pip install pytest
```

- Podrobnější informace zde
-