MODULY
Jmenný prostor (namespace)

1. V Pythonu je všechno objekt
2. Každý objekt má svůj jmenný prostor

Funkce dir vypíše obsah jmenného prostoru

```python
dir()
a=5
dir()
dir(a)

class C:
    x = 5
dir(C)
```
Jak použít („importovat“) modul

1.způsob:
import math
a = math.sqrt( 2 )

import math as mm
a = mm.sqrt( 2 )

2.způsob:
from math import sqrt
a = sqrt( 2 )

3.způsob:
from math import * # => tohle NEDĚLEJTE!
pi
pi = sqrt( 2 )
pi
Jak vytvořit modul

= jen uložit zdrojový kód, jako jakýkoliv jiný

Příkazy uvnitř modulu

Provádí se jen při prvním příkazu import

Proměnná __name__

V hlavním programu má hodnotu '__main__', jinak jméno modulu.

```python
if __name__ == '__main__':
    # co se má provádět jen pro program
```
Proč vytvářet moduly

- rozdělení programu na části
- oddělení kódu, který bychom mohli použít i v jiných projektech
The Python Standard Library
https://docs.python.org/3/library/

- součást instalace Pythonu pro Windows
- ...v Linuxu se doinstalovává obvyklými prostředky
- všestranná knihovna
- moduly napsané v jazyku Python i v jazyku C

...později.
Zajímavé moduly

datetime: Basic date and time types

time: Various functions to manipulate time values

math: Mathematical functions

random: Generate pseudo-random numbers

os.path: Common pathname manipulations

turtle: Turtle graphics

urllib.request: Extensible library for opening URLs

html.parser: Simple HTML and XHTML parser

tkinter: Python interface to Tcl/Tk

Mnoho dalších (ne-standardních) doinstalovatelných
BALÍČEK
(package)
Balíček
Více souborů/modulů, adresářová struktura.
Soubor `__init__.py` v každém (pod)adresáři.

`__init__.py`
• může být prázdný
• příkaz
  ```python
  __all__ = ["aaa", "bbb", "ccc"]
  = seznam modulů pro příkaz
  from balik import *
  ```

V Programování 1 vytvářet nebudeme... (pokračování)
I když přece jen...

- Balíčky jsou užitečné
- Můžeme využívat toho, co už někdo naprogramoval
- Nemusíme vynalézat kolo

1. The Python Package Index (PyPI) https://pypi.org/

586,574 projects  6,308,467 releases  12,617,345 files
(k 21.11.2024)

- Můžeme sami přispět
  a ostatní mohou využívat naše balíčky
Instalace balíčku

**PIP**

```bash
pip install matplotlib
pip list
pip uninstall matplotlib
```
I když přece jen...

- Balíčky jsou užitečné
- Můžeme využívat toho, co už někdo naprogramoval
- Nemusíme vynalézat kolo
- The Python Package Index (PyPI) [https://pypi.org/](https://pypi.org/)

| Projects | Releases | Date             |
|----------|----------|------------------|
| 586,574  | 6,308,467 | 21.11.2024       |
| 498,239  | 5,145,347 | 04.12.2023       |
| 418,408  | 3,974,402 | 29.11.2022       |
| 342,624  | 3,056,532 | 01.12.2021       |
| 274,367  | 2,218,437 | 26.11.2020       |
| 204,421  | 1,536,788 | 10.12.2019       |

- Můžeme sami přispět
  a ostatní mohou využívat naše balíčky

NPRG030 Programování I, 2024/25 [9] 12 / 23 21.11.2024 13:05:59
Jak moc důvěřujete cizímu kódu?

Jak poznáte, co je dobré a důvěryhodné?
(cca 80.000 nových balíčků za rok!)

(Cizí) balíčky mohou vyžadovat další (cizí) balíčky

Balíčky se vyvíjejí a mohou mít různé verse
  (stejně jako všechno ostatní)...

...a různé verse balíčků nemusí být zpětně kompatibilní
  = konec klidného spaní. viz “Dependency hell”
Problémy:

Důvěryhodnost
https://www.root.cz/zpravicky/dve-desitky-balicku-v-pypi-instaluji-malware-pro-kradeni-informaci/

Zpětná kompatibilita
https://docs.microsoft.com/en-us/office/troubleshoot/excel/wrongly-assumes-1900-is-leap-year

Závislosti, různé verze téhož balíčku
(a možná nejsou úplně kompatibilní)

https://about.sourcegraph.com/blog/nine-circles-of-dependency-hell
https://blog.tidelift.com/dependency-hell
Částečné řešení – virtuální prostředí

- balíčky jsou někde uložené
- někde se pamatuje KDE_JSOU_BALÍČKY =>
- vyrobíme nové místo pro ukládání balíčků
- a změníme údaj (cestu) KDE_JSOU_BALÍČKY

```python
python -m venv mojemisto
mojemisto\Scripts\activate
(mojemisto) pip install matplotlib
(mojemisto) pip list
(mojemisto) python -m idlelib.idle
(mojemisto) deactivate
pip list
```
Částečné řešení – virtuální prostředí

Není to úplné řešení problémů s kompatibilitou ani s tím, co všechno se nám nainstaluje, ani se současnou závislostí na různých verzích stejného balíčku...

vlastně to neřeší žádný z problémů zmíněných výše

...ale nepřepisujeme globální instalaci a omezíme nepořádek na jeden zvolený adresář.

Některá IDE vytvářejí nové virtuální prostředí automaticky pro každý projekt (to zase má své náklady).
JAK TESTOVAT 2
modul unittest

- součást Standardní knihovny
- klon JUnit od Kenta Becka a Ericha Gammy
  https://web.archive.org/web/20150315073817/http://www.xprogramming.com/testfram.htm
- dříve nazývaný PyUnit
  http://pyunit.sourceforge.net/pyunit.html
- existují další podobné moduly
Testovací případ

- měl by být nezávislý na okolí (všechno si připravit sám)
- podtřída třídy `unittest.TestCase`
- testy jsou funkce začínající `test...`

- `assert`
- `self.assertEqual`
- `with self.assertRaises`

- `unittest.main()` spustí všechny testy
- `setUp(self)` a `tearDown(self)`
Test Driven Development (TDD)

1. nejdříve napsat testy
2. přesvědčit se, že všechny testy selžou
3. pak psát kód tak dlouho, až všechny testy projdou

https://code.visualstudio.com/docs/python/testing
Pozitivní testy x Negativní testy

- program a) dělá, co má dělat b) nedělá, co nemá dělat
- když má program selhat,
  měl by selhat co nejdříve a mělo by to být slyšet
- Selhání (Failure) vs. Chyba (Error)
- výjimky vs. chybové kódy

```python
isinstance(s, str)
Např.: assert isinstance("1234", int)
```

```python
self.assertRaises(výjimka, funkce, parametry)
Např.: self.assertRaises( ValueError, int, "abc" )
```

NPRG030 Programování I, 2024/25 [9] 21 / 23 21.11.2024 13:05:59
Druhy testů

jednotkové testy (unit-testy)
  testy jednotlivých částí

integrační testy
  jestli části programu budou spolupracovat

regresní testy
  jestli program ještě funguje tak, jak fungoval včera

Funkční a nefunkční požadavky

CI: Countinuous integration (Průběžná integrace)
Poznámka

Jak importovat modul z určité cesty

```python
import sys
sys.path.append("D:/Tom")  # pozor na lomítka!
import mujmodul
```