

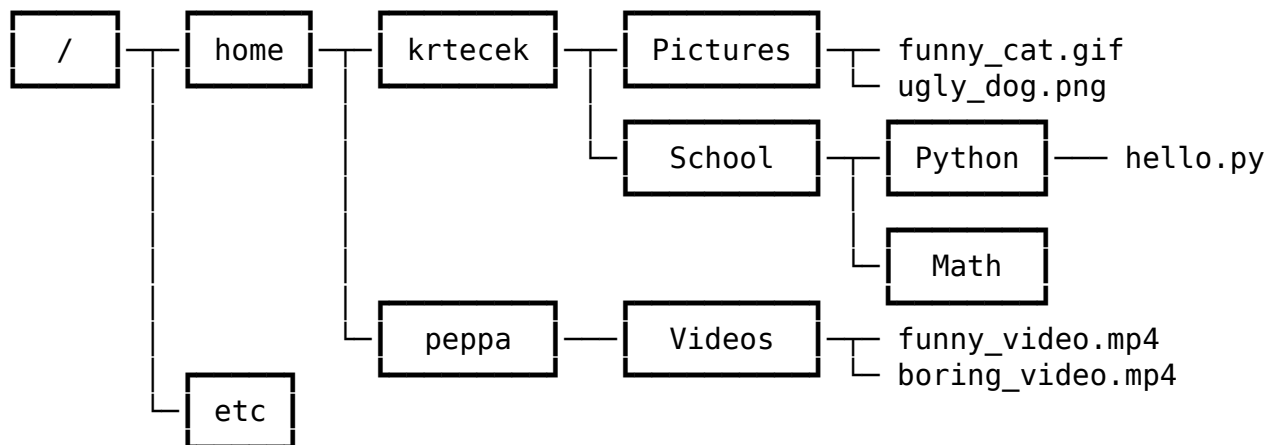
08_Soubory

C2184 Úvod do programování v Pythonu

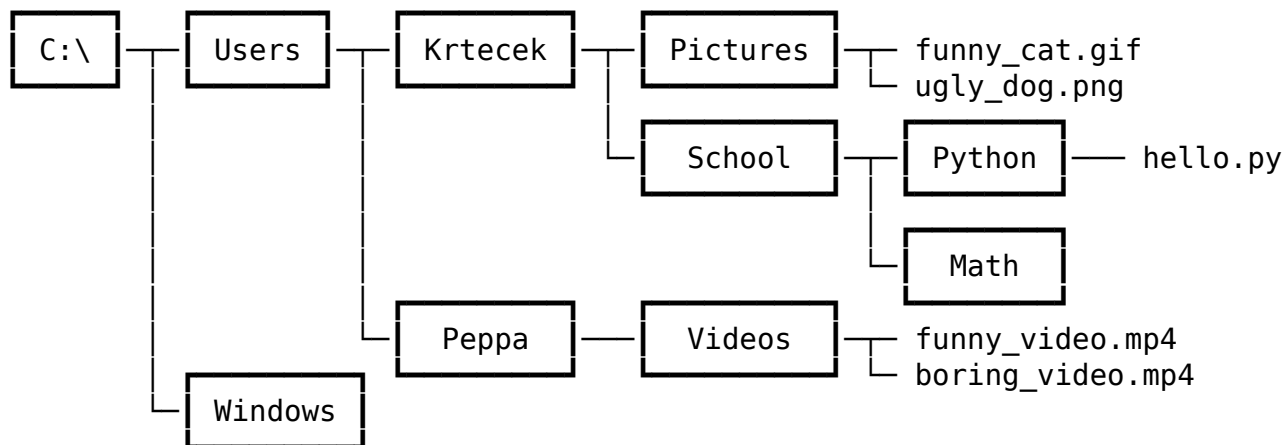
8. Práce se soubory, moduly

Souborový systém

Unix (Linux, MacOS):



Windows:



- **Soubor** (*file*) - sada dat uložených pod konkrétním jménem (např. funny_cat.gif)

- **Adresář, složka** (*directory, folder*) - speciální soubor odkazující na další soubory/složky (např. Pictures)
- **Kořenový adresář** (*root*) - nejvyšší adresář v systému (v Unixu /, ve Windowsu může být víc kořenů: C:\, D:\...)
- **Domovský adresář** (*home directory*) - nastavený pro každého uživatele (např. /home/krtecek), může se zkracovat jako ~
- **Aktuální adresář** (*current working directory*) - kde momentálně jsme, vypisuje se v promptu příkazového řádku
- **Cesta** (*path*) - umístění souboru nebo složky
 - Absolutní cesta - začíná v kořeni souborového systému:
 - /home/krtecek/Pictures/funny_cat.gif (Unix)
 - C:\Users\Krtecek\Pictures\funny_cat.gif (Windows)
 - Relativní cesta - bez kořene, začíná v aktuálním adresáři (řekněme krtecek):
 - Pictures/funny_cat.gif (Unix)
 - Pictures\funny_cat.gif (Windows)
- Speciální znaky v cestě:
 - Oddělovač složek: / (Unix) nebo \ (Windows)
 - . - stejná složka
 - .. - o složku výš (*parent*)

Příklady:

.	=	/home/krtecek
./Pictures/./funny_cat.gif	=	/home/krtecek/Pictures/funny_cat.gif
..	=	/home
../peppa/Videos	=	/home/peppa/Videos
../..	=	/
../../etc	=	/etc
- Příkazový řádek (*bash, powershell, cmd*) - základní příkazy:
 - pwd - vypiš aktuální adresář (*cmd: cd*)
 - ls - vypiš obsah aktuálního adresáře (*cmd: dir*)
 - cd PATH - jdi do adresáře PATH (změň aktuální adresář)
- Python - problémy:
 - Rozdíly mezi Unixem a Windowsem (/ versus \...)
 - V řetězcích je \ speciální znak, takže cestu C:\Users\Krtecek\Pictures\funny_cat.gif musíme zapsat:
 - 'C:\\Users\\Krtecek\\Pictures\\funny_cat.gif'

nebo `r'C:\Users\Kртеcek\Pictures\funny_cat.gif'` (*raw string*)

Modul `pathlib`

- Manipulace s cestami
- Smývá rozdíly mezi Unixem a Windowsem
- Základní typ `Path` reprezentuje cestu

```
[1]: from pathlib import Path
```

- Vytvoření cesty (`Path`):

```
[2]: Path('/home/kртеcek/Pictures/funny_cat.gif')
```

```
[2]: PosixPath('/home/kртеcek/Pictures/funny_cat.gif')
```

- Domovský a aktuální adresář:

```
[3]: Path.home()
```

```
[3]: PosixPath('/home/adam')
```

```
[4]: Path.cwd()
```

```
[4]: PosixPath('/home/adam/School/Python')
```

- Operátor `/` - sřetězení cest (doplní oddělovač `/` nebo `\` podle operačního systému)

```
[5]: Path.home() / Path('Pictures') / Path('funny_cat.gif')
```

```
[5]: PosixPath('/home/adam/Pictures/funny_cat.gif')
```

```
[6]: Path.home() / 'Pictures' / 'funny_cat.gif'
```

```
[6]: PosixPath('/home/adam/Pictures/funny_cat.gif')
```

- Změna z relativní na absolutní cestu

```
[7]: Path('../../funny_cat.gif')
```

```
[7]: PosixPath('../../funny_cat.gif')
```

```
[8]: Path('../../funny_cat.gif').absolute()
```

```
[8]: PosixPath('/home/adam/School/Python/../../funny_cat.gif')
```

```
[9]: Path('../../funny_cat.gif').resolve()
```

```
[9]: PosixPath('/home/adam/funny_cat.gif')
```

- Získání částí cesty

```
[10]: Path('/home/krtecek/Pictures/funny_cat.gif')
```

```
[10]: PosixPath('/home/krtecek/Pictures/funny_cat.gif')
```

```
[11]: Path('/home/krtecek/Pictures/funny_cat.gif').parent
```

```
[11]: PosixPath('/home/krtecek/Pictures')
```

```
[12]: Path('/home/krtecek/Pictures/funny_cat.gif').name
```

```
[12]: 'funny_cat.gif'
```

```
[13]: Path('/home/krtecek/Pictures/funny_cat.gif').stem
```

```
[13]: 'funny_cat'
```

```
[14]: Path('/home/krtecek/Pictures/funny_cat.gif').suffix
```

```
[14]: '.gif'
```

```
[15]: Path('/home/krtecek/Pictures/funny_cat.gif').parts
```

```
[15]: ('/', 'home', 'krtecek', 'Pictures', 'funny_cat.gif')
```

- Dotazování

```
[16]: p1 = Path('/home/krtecek/Pictures/funny_cat.gif')  
p2 = Path('Pictures/funny_cat.gif')
```

```
[17]: p1.is_absolute()
```

```
[17]: True
```

```
[18]: p2.is_absolute()
```

```
[18]: False
```

```
[19]: p2.exists() # existuje?
```

```
[19]: True
```

```
[20]: p2.is_file() # existuje a je normální soubor?
```

[20]: True

```
[21]: p2.is_dir() # existuje a je adresář?
```

[21]: False

Výpis adresářů

- Metoda `.iterdir` - iteruje soubory a adresáře v daném adresáři
- Vrací iterátor, který iteruje v neurčitým pořadí - je vhodné použít `sorted()`

```
[22]: p = Path('.')
p.iterdir()
```

[22]: <generator object Path.iterdir at 0x7f6d800f6eb0>

```
[23]: sorted(p.iterdir())
```

```
[23]: [PosixPath('08_Soubory.ipynb'),
PosixPath('08_Soubory.pdf'),
PosixPath('Logo-Python.png'),
PosixPath('Pictures'),
PosixPath('a.txt'),
PosixPath('b.txt'),
PosixPath('examples'),
PosixPath('funny_cat.gif'),
PosixPath('nakup.txt'),
PosixPath('skarabeus.txt'),
PosixPath('ugly_dog.jpg')]
```

```
[24]: for file in sorted(p.iterdir()):
print(file)
```

```
08_Soubory.ipynb
08_Soubory.pdf
Logo-Python.png
Pictures
a.txt
b.txt
examples
funny_cat.gif
nakup.txt
skarabeus.txt
ugly_dog.jpg
```

Chytrý výpis adresářů

- Metoda `.glob` - iteruje soubory a adresáře vyhovující zadanému výrazu
 - `*` - libovolný počet znaků (tj. i 0 znaků)
 - `?` - jeden libovolný znak
 - `[A-Za-z]` - jeden znak z výčtu

```
[25]: # * = všechny soubory
      for file in sorted(p.glob('*')):
          print(file)
```

```
08_Soubory.ipynb
08_Soubory.pdf
Logo-Python.png
Pictures
a.txt
b.txt
examples
funny_cat.gif
nakup.txt
skarabeus.txt
ugly_dog.jpg
```

```
[26]: # *.txt = soubory končící .txt
      for file in sorted(p.glob('*.txt')):
          print(file)
```

```
a.txt
b.txt
nakup.txt
skarabeus.txt
```

```
[27]: # *n* = soubory obsahující v názvu n
      for file in sorted(p.glob('*n*')):
          print(file)
```

```
08_Soubory.ipynb
Logo-Python.png
funny_cat.gif
nakup.txt
```

```
[28]: for file in sorted(p.glob('Pictures/?ad_giraffe.*')):
      print(file)
```

```
Pictures/bad_giraffe.jpg
Pictures/mad_giraffe.png
Pictures/sad_giraffe.jpg
```

```
[29]: for file in sorted(p.glob('Pictures/[bm]ad_giraffe.*')):  
      print(file)
```

Pictures/bad_giraffe.jpg
Pictures/mad_giraffe.png

- Metoda `.rglob` – rekurzivní glob, tj. prohledává i podadresáře, podpodadresáře atd.

```
[30]: for file in sorted(p.rglob('*.png')):  
      print(file)
```

Logo-Python.png
Pictures/mad_giraffe.png

Manipulace se souborovým systémem

```
[31]: directory = Path('new_dir')  
      directory
```

[31]: PosixPath('new_dir')

- Vytvoření adresáře

```
[32]: directory.mkdir(exist_ok=True) # make directory
```

- Přesun souboru

```
[33]: Path('funny_cat.gif').replace(directory / 'funny_cat2.gif')
```

[33]: PosixPath('new_dir/funny_cat2.gif')

```
[34]: (directory/'funny_cat2.gif').replace('funny_cat.gif')
```

[34]: PosixPath('funny_cat.gif')

- Smazání souboru

```
[35]: Path('ugly_dog.jpg').unlink(missing_ok=True)
```

- Smazání prázdného adresáře

```
[36]: Path('new_dir').rmdir() # remove directory
```

- Další funkce jsou dostupné v modulu `shutil` (na konci prezentace)

Čtení a zápis do souborů

Základní rozdělení formátů

- **Textové** (TXT, HTML, CSV, JSON...) - soubor tvořen posloupností znaků

Vruboun posvátný (*Scarabaeus sacer*), označovaný také jen krátce skarabeus či s
Samička naklade larvy do kuličky uhnětené z trusu, kterou posléze zahrabe do z

- **Binární** (ZIP, JPG, DOCX...) - soubor tvořen posloupností bajtů

Poznámka: Textový soubor je vlastně binární soubor, kde každý bajt nebo skupina bajtů kóduje nějaký znak. Rozdíl je pouze v tom, jak budeme soubor číst.

Základní operace se souborem

- Otevření (*open*)
- Čtení (*read*)
- Zápis (*write*)
- Zavření (*close*) = uložení změn na disk

Otevření a zavření souboru v Pythonu

- Explicitně - hrozí, že zapomeneme soubor zavřít a změny se neuloží

```
[ ]: f = open('my_file.txt')
... # Pracujeme se souborem
... # Pracujeme se souborem
f.close()
```

- Pomocí bloku `with` - doporučený způsob (soubor se zavře automaticky na konci bloku)

```
[ ]: with open('my_file.txt') as f:
... # Pracujeme se souborem
... # Pracujeme se souborem
# Na konci bloku with se soubor automaticky zavře
```

Funkce `open`

- Argumenty:
 - Cesta k souboru (typ `str` nebo `Path`)
 - Mód, kódování...
- Návrátová hodnota:
 - Souborový objekt (*file object*, *file handle*) - popisuje, který soubor je otevřený, mód, kódování, ukazatel na konkrétní pozici v souboru...


```
[37]: with open('skarabeus.txt', mode='r', encoding='utf8') as f:
      print(f)
```

```
<_io.TextIOWrapper name='skarabeus.txt' mode='r' encoding='utf8'>
```

Mód otevření textového souboru (*mode*)

- 'r': pro čtení (*read*) – defaultní mód
 - soubor neexistuje -> FileNotFoundError
 - soubor existuje -> čte od začátku
- 'w': pro zápis (*write*)
 - soubor neexistuje -> vytvoří ho
 - soubor existuje -> přemaže a zapisuje od začátku!
- 'a': pro zápis na konec (*append*)
 - soubor neexistuje -> vytvoří ho a zapisuje od začátku
 - soubor existuje -> doplňuje na konec

Další módy:

- 'x': pro zápis
 - soubor neexistuje -> vytvoří ho
 - soubor existuje -> FileExistsError
- 'r+' pro čtení a zápis
 - soubor neexistuje -> FileNotFoundError
 - soubor existuje -> čte/zapisuje od začátku
- 'w+' pro čtení a zápis
 - soubor neexistuje -> vytvoří ho
 - soubor existuje -> přemaže a čte/zapisuje od začátku

Kódování souborů (*encoding*)

- V souborech reálně nejsou uloženy znaky, ale pouze bajty.
- Kódování popisuje, jak se znaky zakódují do bajtů (*encode*) a zpátky (*decode*).
- Příklady kódování:
 - **UTF-8 (utf8)** – nejmodernější, dokáže zakódovat celou znakovou sadu *Unicode* (některé znaky se ukládají na více bajtů)
 - windows-1250 (cp1250) – střeoevropské
 - windows-1252 (cp1252) – západoevropské

- Znaký *ASCII* (prvních 128 znaků z Unicode) se kódují stejně ve většině kódování
- Defaultní kódování závisí na systému:

```
[38]: import locale
      locale.getpreferredencoding()
```

```
[38]: 'UTF-8'
```

- Příklad špatného rozkódování:

```
[105]: with open('skarabeus.txt', encoding = 'cp1250') as f:
      print(f.read())
```

Vruboun posvátný (Scarabaeus sacer), označovaný také jen krátce skarabeus či skarab, je brouk z čeledi vrubounovití (Scarabaeidae). Žije ve Středomoří. Samička naklade larvy do kuličky uhnětené z trusu, kterou posléze zahrabe do země. Vyvíjející se larva se živí trusem.

```
[106]: with open('skarabeus.txt', encoding = 'utf8') as f:
      print(f.read())
```

Vruboun posvátný (Scarabaeus sacer), označovaný také jen krátce skarabeus či skarab, je brouk z čeledi vrubounovití (Scarabaeidae). Žije ve Středomoří. Samička naklade larvy do kuličky uhnětené z trusu, kterou posléze zahrabe do země. Vyvíjející se larva se živí trusem.

Čtení z textového souboru

- Funguje pouze v módech *r*, *r+*, *w+*
- Metoda `.read` přečte celý soubor (bez argumentu) nebo daný počet znaků (s argumentem)
 - Vrátí načtený řetězec
- Po zavření souboru už nelze číst

```
[107]: with open('skarabeus.txt', mode='r', encoding = 'utf8') as f:
      text = f.read()
      text
```

```
[107]: 'Vruboun posvátný (Scarabaeus sacer), označovaný také jen krátce,
      ↪ \nskarabeus či
      ↪ skarab, je brouk z čeledi vrubounovití \n(Scarabaeidae). Žije ve
      ↪ Středomoří.'
```

Samička naklade larvy do \nkuličky uhnětené z trusu, kterou posléze, ↪zahrabe do země. \nVyvíjející se larva se živí trusem.'

```
[108]: with open('skarabeus.txt', mode='r', encoding = 'utf8') as f:
        first20 = f.read(20)
        next10 = f.read(10)
        rest = f.read()
        print(first20)
        print('-----')
        print(next10)
        print('-----')
        print(rest)
```

Vruboun posvátný (Sc

arabaeus s

acer), označovaný také jen krátce

skarabeus či skarab, je brouk z čeledi vrubounovití

(Scarabaeidae). Žije ve Středomoří. Samička naklade larvy do kuličky uhnětené z trusu, kterou posléze zahrabe do země.

Vyvíjející se larva se živí trusem.

- Metoda `.readline` načte jeden řádek ze souboru jako řetězec.
- Pokud jsme už na konci souboru, vrátí se prázdný řetězec `''`.

```
[43]: with open('nakup.txt', encoding = 'utf8') as f:
        line1 = f.readline()
        line2 = f.readline()
        line3 = f.readline()
        line4 = f.readline() # už jsme na konci souboru
        line5 = f.readline() # už jsme na konci souboru
```

```
[44]: [line1, line2, line3, line4, line5]
```

```
[44]: ['chleba\n', 'jogurt\n', '\n', '', '']
```

- Řádek se načítá vždy se znakem nového řádku na konci.
- Tohoto se zbavíme pomocí metody `.rstrip`.

```
[45]: line1
```

```
[45]: 'chleba\n'
```

```
[46]: line1.rstrip()
```

```
[46]: 'chleba'
```

- Metoda `.readlines` načte všechny řádky (od aktuální pozice) jako seznam

```
[47]: with open('nakup.txt', encoding = 'utf8') as f:  
      lines = f.readlines()  
      lines
```

```
[47]: ['chleba\n', 'jogurt\n', '\n']
```

Čtení souboru pomocí for

- Souborový objekt je zároveň iterátor - iteruje soubor po řádcích

```
[48]: with open('nakup.txt', mode='r', encoding = 'utf8') as f:  
      for line in f:  
          print(line.rstrip())
```

```
chleba  
jogurt
```

- Objekt typu `Path` umí načíst celý soubor (jako `.read`, ale otevře a zavře ho za nás)

```
[49]: Path('nakup.txt').read_text(encoding='utf8')
```

```
[49]: 'chleba\njogurt\n\n'
```

Zápis do textového souboru

- Funguje pouze v módech `w`, `a`, `x`, `r+`, `w+`
- Metoda `.write` zapíše řetězec do souboru
 - Argument musí být vždy pouze řetězec
 - Vrátí počet zapsaných znaků
- Na konec se nekládá automaticky nový řádek, jako u `print`
- Po zavření souboru už nelze zapisovat

```
[50]: with open('novy.txt', mode='w', encoding='utf8') as f:  
      f.write('Alice')  
      f.write('Bob')  
      f.write('Cyril')
```

```
[51]: print(Path('novy.txt').read_text(encoding='utf8'))
```

AliceBobCyril

- Funkci print lze přesměřovat do souboru pomocí parametru file:

```
[52]: with open('novy.txt', mode='w', encoding='utf8') as f:
      print('Alice', 10, True, file=f)
      print('Bob', 20, False, file=f)
```

```
[53]: print(Path('novy.txt').read_text(encoding='utf8'))
```

Alice 10 True
Bob 20 False

- Objekt typu Path umí zapsat celý soubor (jako .write, ale otevře a zavře ho za nás)

```
[54]: Path('novy.txt').write_text('blablabla', encoding='utf8')
```

```
[54]: 9
```

```
[55]: print(Path('novy.txt').read_text(encoding='utf8'))
```

blablabla

- Můžeme otevřít několik souborů současně

```
[56]: with open('a.txt') as fa:
      with open('b.txt') as fb:
          with open('ab.txt', mode='w') as fab:
              for line_a, line_b in zip(fa, fb):
                  print(line_a.rstrip(), line_b.rstrip(), file=fab)
```

```
[57]: with open('a.txt') as fa, open('b.txt') as fb, open('ab.txt',
      ↪mode='w') as fab:
      for line_a, line_b in zip(fa, fb):
          print(line_a.rstrip(), line_b.rstrip(), file=fab)
```

```
[58]: print(Path('a.txt').read_text())
      print('-----')
      print(Path('b.txt').read_text())
      print('-----')
      print(Path('ab.txt').read_text())
```

Alice
Bob
Cyril

10
20

```
30
-----
Alice 10
Bob 20
Cyril 30
```

Metody tell a seek

- Metoda `.tell` vrací aktuální pozici ukazatele, `.seek` nastavuje pozici ukazatele

```
[59]: with open('nakup.txt') as f:
      print(f.tell())
      print(f.readline().rstrip())
      print(f.tell())
      print(f.readline().rstrip())
      print(f.tell())
```

```
0
chleba
7
jogurt
14
```

```
[60]: with open('nakup.txt') as f:
      f.seek(7)
      print(f.readline().rstrip())
      f.seek(0)
      print(f.readline().rstrip())
```

```
jogurt
chleba
```

Speciální souborové objekty

- `sys.stdin` - čte ze standardního vstupu
- `sys.stdout` - zapisuje na standardní výstup
- `sys.stderr` - zapisuje na standardní chybový výstup

```
[61]: import sys
      sys.stdout.write('ahoj')
```

```
ahoj
```

Ukončování řádků

- Unix (Linux a MacOS) ukončuje řádky jedním znakem `'\n'` (LF = *line-feed*)

- Windows ukončuje řádky dvojicí znaků '\r\n' (CRLF = *carriage-return line-feed*)
- Python tento rozdíl skrývá
 - '\n' i '\r\n' se načte jako '\n'
 - '\n' se zapíše jako '\n' v Unixu, jako '\r\n' ve Windowsu
- VSCode vpravo dole vypisuje kódování a ukončování řádku, umožňuje snadnou konverzi

Na co si dát pozor

- Kódování souborů
- Řádky vždy končí bílými znaky (\n), proto je vhodné použití metod `.rstrip` nebo `.strip`
- Když soubor zavřeme, už s ním nemůžeme dále pracovat (read, write); jedině že si ho znovu otevřeme
- Pokud soubor otevíráme v módu 'w', hrozí ztráta dat - současný soubor je ihned nenávratně přepsán novým prázdným souborem

Binární soubory

- Místo řetězců (typ str) čteme a zapisujeme bajty (typ bytes)
- Binární módy otevření: 'rb', 'wb', 'ab', 'xb', 'r+b', 'w+b'
- Metoda `.read` vrací typ bytes
- Metoda `.write` bere typ bytes

```
[62]: with open('funny_cat.gif', mode='r+b') as f:
      content = f.read(100)
      content
```

```
[62]: b'GIF89a\xc8\x00\xfa\x00\xf7\xfd\x00\x16\x0b\x04\x1a\x0e\x0c\x1a\x14\x14\x1b\x12
\x0c\x1d\x18\x1a"\x1c\x1d#\x18\x14$\x15\x0b%\x1e!&
  ↪ #*AI+\x1d\x1a+#\x1c+%,!\x15-
\x1a\x0c-\x1c\x14-&).).*%.JT01B2,*2NZ3%\x1d3+%4"\x134)\x1b61-'
```

- Konverze mezi str a bytes pomocí metod `.encode` a `.decode`:

```
[63]: 'Náhodná věta'.encode() # UTF-8 kódování
```

```
[63]: b'N\xc3\xa1hodn\xc3\xa1 v\xc4\x9bta'
```

```
[64]: 'Náhodná věta'.encode('cp1250') # Windows středoevropské kódování
```

```
[64]: b'N\xe1hodn\xe1 v\xectata'
```

```
[65]: b'N\xc3\xa1hodn\xc3\xa1 v\xc4\x9bta'.decode()
```

```
[65]: 'Náhodná věta'
```

```
[66]: b'N\xc3\xa1hodn\xc3\xa1 v\xc4\x9bta'.decode('cp1250')
```

```
[66]: 'NǺ˘hodnǺ˘ vǺ˘ta'
```

Moduly

Modul (*module*)

- Soubor funkcí a proměnných, které lze importovat

```
[67]: import math
```

```
[68]: math
```

```
[68]: <module 'math' (built-in)>
```

```
[69]: math.cos(0)
```

```
[69]: 1.0
```

```
[70]: math.pi
```

```
[70]: 3.141592653589793
```

Balíček (*package*)

- Modul obsahující další moduly (složka s moduly)

```
[71]: import os
```

```
[72]: os
```

```
[72]: <module 'os' from '/usr/lib/python3.8/os.py'>
```

```
[73]: os.path
```

```
[73]: <module 'posixpath' from '/usr/lib/python3.8/posixpath.py'>
```


Import pomocí from a as

- from - importujeme modul z balíčku nebo proměnnou (funkci) z modulu
- as - přejmenování importovaného modulu/proměnné/funkce

```
[74]: from os import path  
path
```

```
[74]: <module 'posixpath' from '/usr/lib/python3.8/posixpath.py'>
```

```
[75]: from math import factorial, pi  
factorial(6) + pi
```

```
[75]: 723.1415926535898
```

```
[76]: import numpy as np  
np
```

```
[76]: <module 'numpy' from '/usr/lib/python3/dist-packages/numpy/__init__.  
↳py'>
```

```
[77]: from math import factorial as fact, pi as PI  
fact(6) + PI
```

```
[77]: 723.1415926535898
```

Spuštění modulu jako skriptu

- Z příkazového řádku pomocí přepínače -m:
\$ python -m doctest

Zdroje modulů / balíčků

- Standardní knihovna Pythonu
 - Moduly přítomné při instalaci
 - math, os, sys...
 - <https://docs.python.org/3.8/library/>
- PyPI - Python Package Index
 - Moduly doinstalovatelné např. pomocí nástroje pip
 - numpy, sklearn...
 - <https://pypi.org/>
- Vlastní moduly

- Každý pythonovský skript lze importovat jako modul

Pip

- Modul sloužící ke stáhnutí a doinstalování balíčků z PyPI
- Spouštíme ho vždy z příkazového řádku pomocí -m (nesmí se importovat)

```
$ python -m pip install numpy # Nainstaluj balíček numpy
```

```
$ python -m pip show numpy # Vypiš verzi a info o nainstalovaném balíčku
```

```
$ python -m pip search sound # Vyhledej balíčky související se zvukem
```

```
$ python -m pip list # Vypiš seznam nainstalovaných balíčků
```

```
$ python -m pip help # Vypiš nápovědu k pipu
```
- V učebně A4/118 pip nefunguje (balíčky instaluje admin).

Vlastní moduly

- 1 soubor .py = 1 modul
- Máme dlouhý program -> můžeme ho rozdělit na více modulů
 - Z hlavního skriptu (`__main__`) pak importujeme ostatní moduly
 - Stejný modul lze importovat do více skriptů nebo do dalších modulů

Soubor `examples/obsahy.py`:

```
"""  
Modul pro výpočet obsahu různých geometrických útvarů.  
"""
```

```
import math
```

```
def obsah_obdelniku(a: float, b:float) -> float:  
    """Vrať obsah obdélníku o stranách a, b."""  
    return a * b
```

```
def obsah_ctverce(a: float) -> float:  
    """Vrať obsah čtverce o straně a."""  
    return a**2
```

```
def obsah_kruhu(r: float) -> float:  
    """Vrať obsah kruhu o poloměru r."""  
    return math.pi * r**2
```

```
[78]: from examples import obsahy
```

```
[79]: obsah.obsah_ctverce(5.0)
```

```
[79]: 25.0
```

```
if __name__ == '__main__':
```

- Tento blok se vykoná, pouze když modul spouštíme jako skript.
- Když modul importujeme, tento blok se nevykoná.
- (Toto funguje díky tomu, že pokud je modul importován, v magické proměnné `__name__` je název modulu; pokud je spouštěn jako skript, v proměnné `__name__` je `'__main__'`.)

Soubor `ukazkove_moduly/fibonacci.py`:

```
"""
Modul pro generování Fibonacciho posloupnosti.
"""

from typing import List

def fib(n: int) -> List[int]:
    """Vrať prvních n prvků Fibonacciho posloupnosti."""
    result = []
    a, b = 1, 1
    while len(result) < n:
        result.append(a)
        a, b = b, a+b
    return result

def main() -> None:
    """Interaktivní výpis Fibonacciho posloupnosti."""
    pocet = int(input('Zadej požadovaný počet prvků: '))
    posloupnost = fib(pocet)
    print(*posloupnost, sep=', ')

if __name__ == '__main__':
    main()
```

- Import:

```
[80]: from examples import fibonacci as fib
```

```
[81]: fib.fib(10)
```

```
[81]: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

- Spuštění z příkazového řádku:

```
$ python examples/fibonacci.py
```

```
Zadej požadovaný počet prvků: 10  
1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

nebo

```
$ python -m examples.fibonacci
```

```
Zadej požadovaný počet prvků: 8  
1, 1, 2, 3, 5, 8, 13, 21
```

Typická struktura pythonovského programu

- Docstring - co tento modul/skript dělá
- Importy
- Konstanty
- Funkce
- `if __name__ == '__main__': main()`

Užitečné moduly

Standardní knihovna: <https://docs.python.org/3.8/library/>

Python Package Index: <https://pypi.org/>

Modul math

- Matematické funkce

```
[82]: import math  
math.comb(15, 3)
```

```
[82]: 455
```

Modul cmath

- Matematické funkce nad komplexními čísly

```
[83]: import cmath  
cmath.sqrt(-1)
```

```
[83]: 1j
```

```
[84]: cmath.sin(2+1j)
```

```
[84]: (1.4031192506220405-0.4890562590412937j)
```

Modul random

- Generování pseudo-náhodných čísel

```
[85]: import random
```

```
[86]: random.random() # Náhodné reálné číslo z intervalu [0, 1)
```

```
[86]: 0.15120019206798319
```

```
[87]: random.randint(0, 100) # Náhodné celé číslo z intervalu [0, 100]
```

```
[87]: 47
```

```
[88]: random.choice(['červená', 'zelená', 'modrá', 'černá']) # Náhodný  
↳ výběr
```

```
[88]: 'zelená'
```

Modul datetime

- Práce s časovými údaji
- Základní typy:
 - datetime - datum a čas
 - timedelta - trvání
 - date - datum
 - time - čas
 - timezone - časová zóna

```
[89]: from datetime import datetime, timedelta
```

```
start = datetime.now()  
math.factorial(500_000)  
end = datetime.now()  
calculation_time = end - start  
print(calculation_time)
```

```
0:00:02.379853
```

```
[90]: today = datetime.now().date()  
in_a_week = today + timedelta(weeks=1)  
print(f'Today is: {today}')  
print(f'In a week it will be: {in_a_week}')
```

Today is: 2020-12-03
In a week it will be: 2020-12-10

- Formátování času

```
[91]: datetime.now().strftime('%A, %d %B %Y, %H:%M:%S')
```

```
[91]: 'Thursday, 03 December 2020, 20:01:37'
```

- Parsování času

```
[92]: datetime.strptime('1. 7. 2000, 13:30:05', '%d. %m. %Y, %H:%M:%S')
```

```
[92]: datetime.datetime(2000, 7, 1, 13, 30, 5)
```

Modul itertools

- Různé možnosti iterování

```
[93]: import itertools
```

```
[94]: for pair in itertools.combinations(['bílá', 'zelená', 'modrá'], 2):  
      print(*pair)
```

```
bílá zelená  
bílá modrá  
zelená modrá
```

```
[95]: for pair in itertools.permutations(['bílá', 'zelená', 'modrá'], 2):  
      print(*pair)
```

```
bílá zelená  
bílá modrá  
zelená bílá  
zelená modrá  
modrá bílá  
modrá zelená
```

```
[96]: for letter, number in itertools.zip_longest('ABCDE', [1, 2, 3],  
      ↪ fillvalue='?'):  
      print(letter, number)
```

```
A 1  
B 2  
C 3  
D ?  
E ?
```

Modul sys

- Funkce závislé na systému

```
[97]: import sys
```

```
[98]: sys.version # Verze interpretu
```

```
[98]: '3.8.5 (default, Jul 28 2020, 12:59:40) \n[GCC 9.3.0]'
```

- `sys.argv` - seznam argumentů z příkazového řádku
 - Nultý argument = název skriptu

Soubor `examples/suma.py`:

```
import sys

print(sys.argv)
suma = sum(int(x) for x in sys.argv[1:])
print(suma)
```

- Spustíme z příkazového řádku:

```
$ python ukazkove_moduly/suma.py 1 5 8 ['suma.py', '1', '5', '8']
14
```

Moduly os a os.path

- Funkce závislé na operačním systému
 - `os.chdir()` - změň aktuální adresář
- Většina dalších funkcí je i v modulu `pathlib`

```
[99]: import os
print(Path.cwd())
os.chdir('examples')
print(Path.cwd())
os.chdir('.')
print(Path.cwd())
```

```
/home/adam/School/Python
/home/adam/School/Python/examples
/home/adam/School/Python
```

Modul shutil

- Více možností než `os` a `pathlib`

```
[100]: import shutil
Path('stuff').mkdir(exist_ok=True) # Vytvoř adresář stuff
```

```
Path('stuff', 'foo').mkdir(exist_ok=True) # Vytvoř adresář stuff/foo
Path('stuff', 'file1.txt').write_text('blabla') # Vytvoř soubor
↳stuff/file1.txt
Path('stuff', 'foo', 'file2.txt').write_text('blabla') # Vytvoř
↳soubor stuff/foo/file2.txt
```

[100]: 6

```
[101]: shutil.rmtree('stuff') # Smaž adresář nový a všechno v něm
```

```
[102]: shutil.copy('Pictures/funny_cat.gif', Path.home()/'CLICK_HERE.gif')
↳# Zkopíruj soubor
```

[102]: PosixPath('/home/adam/CLICK_HERE.gif')

```
[103]: shutil.copytree('Pictures', Path.home()/'magic_inside') # Zkopíruj
↳adresář a všechno v něm
```

[103]: PosixPath('/home/adam/magic_inside')

Zkoušíme otevřít soubor

- Ask for forgiveness, not for permission.
- Nezjišťujeme, jestli soubor existuje a lze ho otevřít. Prostě ho zkusíme otevřít.

```
[104]: try:
        with open('neexistujici_soubor.txt') as f:
            print(f.read())
    except FileNotFoundError:
        print('Soubor neexistuje.')
    except PermissionError:
        print('Nemáš právo číst soubor.')
    except OSError:
        print('Soubor se nepodařilo otevřít.')
```

Soubor neexistuje.

- OSError je obecnější typ výjimky, zahrnuje FileNotFoundError, FileExistsError, PermissionError...

Další užitečné moduly

- argparse
- requests
- subprocess

- json
- csv
- pickle
- email
- re
- codecs
- warnings
- numpy
- matplotlib
- sklearn
- pandas
- ...