

Programovanie so zlyhaním a súbory

Chyby

Pokiaľ vám Python vypíše chybu, tak to môže byť:

- ▶ `SyntaxError`
- ▶ výnimka (exception)

Asi ste si všimli, že výnimiek je celá rada, najčastejšie `TypeError`, `ValueError`, `IndexError`, `KeyError`, *atď.*

raise

Ak chcete vyhodit výnimku, použijete klíčové slovo `raise`, napr.:

```
def is_prime(x):  
    if type(x) is not int:  
        raise TypeError("Only ints can be primes!")  
    else:  
        ...
```

try, except

```
def safe_div(a, b):  
    try:  
        result = a/b  
    except ZeroDivisionError:  
        print("Division by zero!")  
        result = None  
  
    return result
```

pass

Slovo `pass` v Pythone znamená neurob nič. Je potrebné v prázdnych indetovaných blokoch (skutočne prázdny blok je `SyntaxError`).

Napr:

```
def do_nothing():  
    pass
```

alebo:

```
for i in range(100):  
    pass
```

Samotná existencia tohoto vyzerá ako filozofický problém, ale má to naozaj využitie. . .

try, except, pass

```
def safe_div(a, b):  
    try:  
        return a/b  
    except:  
        pass
```

try, except, else, finally

```
def safe_div(a, b):  
    try:  
        result = a/b  
    except ZeroDivisionError:  
        print("Division by zero!")  
        result = None  
    except TypeError:  
        print("Only numbers allowed!")  
        result = None  
    else:  
        print("No exceptions :-)")  
    finally:  
        print("Tried to divide {} / {}".format(a, b))  
  
    return result
```

Typy súborov

- ▶ Textové - tie, ktoré dokážete editovať (.txt, .py, .ipynb, ...)
- ▶ Binárne (.jpg, .docx, ...)

V tomto predmete budete otvárať Pythonom len textové súbory, ale pracovať s binárnymi je tiež možné.

Práca so súbormi

Súbory sú dáta uložené na disku. Načítanie z disku je rádovo pomalšie ako z RAM. Prvým krokom v práci so súbormi bude teda načítanie súboru do RAM:

```
f = open("subor.na_pripone_nezalezi")  
  
obsah = f.read()  
  
f.close()  
  
print(type(obsah)) # str
```

IO môže vždy zlyhať

- ▶ Čo ak `f.read()` zlyhá?
- ▶ Čo ak zabudnete zavolať metódu `close`?

Prečo zatvárať súbory

with syntax

Aby ste nikdy nezabudli zatvoriť súbor, používajte kľúčové slová `with-as`:

```
with open("subor.pripona") as f:  
    obsah = f.read()    # indentovany blok
```

Nakonci bloku, si `with` "po sebe uprace".

with bližšie

Použitie slova `with` je možné na tzv. *context manager* objektoch – objekty, ktoré si po sebe dokážu upratať.

S konštrukciou `try-finally` by ste mohli docieľiť *podobného efektu*:

```
f = open("subor.cokolvek")

try:
    obsah = f.read()
except:
    print("Something bad has happened!!!")
finally:
    f.close()
```

Veľké súbory

Niektoré súbory sú obrovské, oveľa väčšie ako vaša RAM. V tom prípade musíte načítavať súbor postupne. Tu je viac možností a bude to závisieť na type vášho súboru.

Najjednoduchšie je ísť po riadkoch pomocou jednoduchého for-cyklu:

```
with open("subor.pripona") as f:
    for riadok in f:
        print(riadok)
```

Ak zavoláte metódu `read` s argumentom n (číslo), tak načítate n znakov:

```
with open('subor') as f:
    tri_znaky = f.read(3)
    print(len(tri_znaky))
```

Čo je vlastne v premennej f?

Objekt je tzv *file descriptor* (FD).

- ▶ Ako sme videli na predchádzajúcom slide, FD je iterovateľný.
- ▶ **Je iterovateľný ale len jeden-krát!!!**
- ▶ užitočné metódy: read, write, close, readlines, seek, tell.

Viac info na oficiálnej stránke Pythonu

```
i = 0
with open('my_file.txt') as f:
    for line in f:
        i += 1
        print(i, ': ', line)

for line2 in f:
    print(line2) # no output
```

Zápis do súboru

Read	Write
<code>open('my_file.txt')</code>	<code>open('my_file.txt', 'w')</code>
<code>content = f.read()</code>	<code>f.write(content)</code>

POZOR: Pokiaľ súbor už existuje tak ho prepíšete:

```
with open('hello.txt', 'w') as f:  
    f.write('Hello world!')
```

Módy open

Reťazec 'w' v zápise znamená *write*

Character	Meaning
'r'	open for reading (default)
'w'	open for writing
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists

... takže, ak si nechcete prepísať existujúci súbor, tak použite 'x'

Špeciálne znaky

Znak v reťazci	Poznámka
'\n'	nový riadok
'\r'	návrat vozíku (nerobím si srandu)
'\\'	vytvorí jednoduché opačné lomítko

Viac o nových riadkoch **napr. tu**.

Relatívna a absolútna cesta

Pokiaľ chcete otvoriť súbor v inom adresári, použite *absolútnu* alebo *relatívnu* cestu:

```
with open('../ina_zlozka/dalsia_zlozka/subor.txt'):  
    ...
```

Pozor

Pozor na opačné lomítka.

```
with open('C:\\Users\\Documents\\subor.txt'):  
    ...
```

Balíčky

“In the name of God, stop a moment, cease your work, look around you.” — Leo Tolstoy

Pre väčšinu štruktúrovaných súborov, s ktorými sa stretnete už existuje *parser*.

- ▶ CSV (Comma-Separated Values): balíček `csv`, funkcia z balíčka `pandas` `read_csv`
- ▶ Tabuľkové súbory: `numpy.genfromtxt`, `pandas.read_table`
- ▶ Excel-ovská tabuľka: `pandas.read_excel`

Komplexná práca so súbormi

- ▶ `os.path` - práca s cestami a zložkami
- ▶ `shutil` a `os` - mazanie súborov, kopírovanie, atď
- ▶ `glob` - veľmi užitočné pokiaľ máte viac súborov a chcete ich roztriediť podľa názvu.
- ▶ `pathlib` - skoro všetko nad