Čtení z… a psaní do… souborů

1) otevřít

```
>>> f = open( "vstup.txt", "r" )
>>> g = open( "vystup.txt", "w" )
```

2) číst nebo psát

```
>>> vsechno = f.read()
```

nebo

```
>>> for radek in f:
        g.write( radek )
```

3) zavřít

```
>>> f.close()
>>> g.close()
```

Proč zavírat soubory...

```
>>> g = open("d:/c.txt","w")
>>> for i in range(5000):
        g.write(str(i)+" ")
```

CHYBY

Každý dělá chyby!
(Jenom ne každý si to uvědomuje.)

Chyby v programech

1. syntaktické chyby (syntax error)
   = ty ohlásí překladač
2. běhové chyby (runtime error)
   = ty ohlásí program při běhu
3. sémantické chyby (semantic error)
   = ty neohlásí nikdo
   (až zklamaný uživatel, pokud si jich všimne)

Neodhalené chyby mohou být velmi drahé!
(např. Mars Climate Orbiter, 125 milionů USD)

Výjimka

= zpráva o běhové chybě

```
>>> a
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    a
NameError: name 'a' is not defined

>>> a = 5
>>> a[1]
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    a[1]
TypeError: 'int' object is not subscriptable
```

Typ výjimky

Výjimka je objekt
a NameError, TypeError …apod. je jeho typ.

Obsahuje i další informace, jako text zprávy

```
name 'a' is not defined
```

nebo místo, kde k chybě došlo (celý zásobník volání)

```
File "<pyshell#2>", line 1, in <module>
   a[1]
```

.

Jak odchytit výjimku

```
>>> while True:
        try:
            x = int(input("Zadej cislo: "))
            break
        except ValueError:
            print("   Zkus to znovu...")

Zadej číslo:
   Zkus to znovu...
Zadej číslo: sss
   Zkus to znovu...
Zadej číslo: ghgfdhgfd
   Zkus to znovu...
Zadej číslo: 95
>>> x
95
```

Ještě k except

- Lze napsat více bloků except,
  každý pro jiný typ výjimek
- Lze také neuvést žádný typ
  a tím odchytit jakoukoli výjimku, nedělejte to!

(Každý by měl řešit jen ty problémy, kterým rozumí.)

(Pokud se někde dozvíme o chybě,
měli bychom být rádi a nezahazovat tu informaci!)

Ještě k except

Nastalou výjimku si můžeme uložit do proměnné...

```
>>> while True:
        try:
            x = int( input( "Zadej číslo: " ) )
            break
        except ValueError as vyjimka:
            print( "To nebylo číslo..." )
            print( type(vyjimka) )
            print( vyjimka.args )
            print( vyjimka )
            v = vyjimka
            # vyjimka je lokalni, v je globalni
```

Příklad *(podle thinkcspy3)*

```
def existuje(filename):
    try:
        f = open(filename)
        f.close()
        return True
    except FileNotFoundError:
        return False
```

NEBO lépe

```
import os
...
return os.path.isfile("c:/temp/testdata.txt"):
```

Výjimkami bychom měli ošetřovat jen **výjimečné** situace.

Finally

Už víme, že soubory je potřeba zavírat...

```
g = open("d:/e.txt","w")
try:
    for i in range(10):
        x = int(input("zadej cislo:"))
        g.write( str(x)+" " )
finally:
    g.close()
```

Vyvolání výjimky

proč bychom to měli chtít?

chceme ohlásit výjimečnou situaci,
kterou sami neumíme vyřešit

Například:

funkce má ze vstupu přečíst číslo (a není tam)
máme zjistit údaje z webu a server neodpovídá
...

Vyvolání výjimky

syntaxe:

```
raise ValueError ("Počet trpaslíků < 0")
```

Všechny typy výjimek jsou odvozeny od třídy

BaseException

, můžeme si odvodit vlastní typy.

výjimky vs. návratové kódy

Assert

```
assert 1<2, "Něco není v pořádku"
...
AssertionError: Něco není v pořádku
```

je zkrácený zápis za

```
if not podmínka:
    raise AssertionError( text )
```

POZOR: assert je příkaz, ne funkce! (I v Pythonu3)
=> bez závorek!

Defenzivní programování

Cokoliv se může pokazit, to se pokazí
a je dobré se o tom dozvědět včas.

- ověřování vstupu (chyby od ostatních)
- ...i výstupu (naše vlastní chyby)
- testování vstupních a výstupních podmínek a invariantů
- ověřování situací, které nemohou nastat
- nízká tolerance (co neznáme, odmítnout)
- ...a další

To, že jsi paranoik, ještě neznamená, že po tobě nejdou.
/Woody Allen/