Výnimky

Určite sa vám už stalo, že vám program havaroval (neočakávane sa ukončil a vypísal nejaký oznam - chybové hlásenie). Napríklad ak ste pri zadávaní reálneho čísla namiesto desatinnej bodky použili desatinnú čiarku. Počas vykonávania programu môže dôjsť k situácii, keď počítač z nejakého dôvodu nevie ukončiť vykonanie príkazu. Napríklad ak program očakáva zadanie čísla a užívateľ zadá vstupnú hodnotu, ktorú počítač nevie konvertovať (zmeniť) na číslo; pri delení zistí, že menovateľ má hodnotu nula, alebo sa od neho požaduje otvorenie súboru, ktorý nenájde na predpokladanom mieste na disku a pod. Hovoríme, že nastala výnimka. Je na programátorovi, aby príkazy, pri vykonaní ktorých môže nastať výnimka, pri písaní programu ošetril. Robí sa tak programovou konštrukciou try-except, používanie ktorej si ozrejmíme na príkladoch.
„Pohrajme sa“ najprv s požiadavkou zadať z klávesnice celé číslo a vypísať jeho dvojnásobok.
Jednoduchá sekvencia (postupnosť príkazov):

```
vstup = input("Zadaj celé číslo: ")                    # do premennej vstup sa uloží reťazec zadaný z klávesnice
cele_cislo = int(vstup)                                # vstup sa konvertuje na celé číslo a uloží do premennej
print("Dvakrát celé číslo:", 2*cele_cislo)             # obsah premennej cele_cislo sa vynásobí dvoma a zobrazí
```

Ak zadáme korektne (správne) celé číslo, dostaneme správny výsledok. Ak však stlačíme Enter, t.j. vložili sme prázdny reťazec, program havaruje a vypíše chybové hlásenie (posledný riadok hlásenia „Chyba hodnoty: nedovolený znak v int(): 'prázdny reťazec' “):

```
>>>
Zadaj celé číslo:
Traceback (most recent call last):
  File "C:/Python/vynimky3.py", line 2, in <module>
    cele_cislo = int(vstup)
ValueError: invalid literal for int() with base 10: ''
>>>
```

Ako vidieť z hlásení nižšie, podobne program zareaguje aj na vloženie reťazcov 2 a O (písmeno O) alebo na vloženie reálneho čísla 20.0.

```
ValueError: invalid literal for int() with base 10: '2O'
ValueError: invalid literal for int() with base 10: '20.0'
```

Hovoríme, že nastali výnimky a zadanie celého čísla užívateľom by sme mali ošetriť.
Predbežne predpokladajme tú najjednoduchšiu situáciu, t.j. program po vzniku výnimky ukončíme. Ukončiť program možno zavolaním funkcie sys.exit(), ktorá je v knižnici sys, preto ju musíme na začiatku programu importovať (pridať) do programu. K výnimke môže dôjsť pri pokuse konvertovať zadaný reťazec na celé číslo, preto tento príkaz zapíšeme do bloku try. Ošetrenie výnimky sa deje v bloku except a keďže všetky tri výnimky uvedené vyššie sú toho istého charakteru - ValueError, ošetrenie môže mať tvar:

```
import sys
vstup = input("Zadaj celé číslo: ")                                    # nemôže nastať problém
try:                                                                   # začiatok bloku try
   cele_cislo = int(vstup)                                             # problémový príkaz
except ValueError:                                                     # začiatok bloku except s názvom chyby
   print("Zadaný prázdny reťazec alebo nedovolený znak, ukončím program!")          # naše chybové hlásenie
   sys.exit()                                                          # ukončenie behu programu
print("Dvakrát celé číslo:", 2*cele_cislo)                             # vykoná sa, len ak nenastane výnimka!
```

Takéto ošetrenie výnimky môžeme uplatniť aj na zadanie reálneho čísla:

```
import sys
vstup = input("Zadaj reálne číslo: ")
try:
   realne_cislo = float(vstup)
except ValueError:
```

```
    print("Zle zadané reálne číslo, ukončím program!")
    sys.exit()
print("Dvakrát reálne číslo:", 2*realne_cislo)
```

Použitie:

Zadaj reálne číslo: 10,5
Zle zadané reálne číslo, končím!

Zadaj reálne číslo: 1e3 e (exponent) čítame „10 na“, t.j. 1e3 znamená 1.10^3 čo je 1000
Dvakrát reálne číslo: 2000.0

Skúsme teraz trochu sofistikovanejšie (prepracovanejšie) ošetrenie zadania celého čísla. Ukončenie programu pri zle zadanom vstupe zrejme nie je vhodným ošetrením. Nech sa program pri vzniku výnimky neukončí ale znova vráti na začiatok, t.j. na výzvu Zadaj celé číslo. Použijeme nekonečný cyklus while, z ktorého sa vystúpi, ak pri konverzii reťazca na int nedošlo k výnimke, t.j. bolo korektne zadaná celé číslo - nie je dôvod znova opakovať zadanie vstupu. Program po vystúpení z cyklu následne pokračuje príkazmi za blokom except. Ujasnite si, ktoré riadky sú v cykle (podľa odsadenia), ktoré mimo cyklu!

```
while True:
    vstup = input("Zadaj celé číslo: ")
    try:
        cele_cislo = int(vstup)
        break                    # v príkaze vyššie sa neodišlo do bloku except, tzn. má sa pokračovať za cyklom!
    except ValueError:
        print("Zadaný prázdny reťazec alebo nedovolený znak!")
print("Dvakrát celé číslo:", 2*cele_cislo)
```

```
>>>
Zadaj celé číslo:
Zadaný prázdny reťazec!
Zadaj celé číslo: 20
Zadaný nedovolený znak!
Zadaj celé číslo: 10.0
Zadaný nedovolený znak!
Zadaj celé číslo: 50
Dvakrát celé číslo: 100
>>>
```

V bloku try, ak je v cykle, musí byť vždy nejakým spôsobom (v našom prípade príkazom break) zabezpečené vystúpenie z cyklu, ak nenastane výnimka.

Ak požadujeme zadanie nezáporného celého čísla, potrebujeme vyvolať výnimku aj keď bolo celé číslo zadané korektne, ale záporné. Možné ošetrenie:

```
while True:
    vstup = input("Zadaj nezáporné celé číslo: ")
    try:
        nezaporne_cele_cislo = int(vstup)
        if nezaporne_cele_cislo < 0:
            raise ValueError                    # vyvolanie výnimky ValueError
        break
    except ValueError:
        print("Zlý vstup!")
print("Dvakrát celé číslo:", 2* nezaporne_cele_cislo)
```

Alebo sofistikovanejšie:

```
while True:
   vstup = input("Zadaj nezáporné celé číslo: ")
   try:
      nezaporne_cele_cislo = int(vstup)
      if nezaporne_cele_cislo < 0:          # na if sa dostane, ak konverzia na celé číslo nevyvolala výnimku
         raise ValueError                   # číslo je záporné, ošetri výnimku ValueError
         continue                           # pokračuj na začiatku cyklu
      else:
        break                               # číslo nebolo záporné, resp. je nezáporné, vystúp z cyklu
   except ValueError:
      if len(vstup) == 0:
         print("Zadal si prázdny reťazec!")
      elif vstup[0] == '-':
         print("Zadal si záporné číslo!")
      else:
         print("Zadal si nedovolený znak!")
print("Dvakrát celé číslo:", 2*nezaporne_cele_cislo)
```

```
>>>
Zadaj nezáporné celé číslo:
Zadal si prázdny reťazec!
Zadaj nezáporné celé číslo: 1O
Zadal si nedovolený znak!
Zadaj nezáporné celé číslo: -5
Zadal si záporné číslo!
Zadaj nezáporné celé číslo: 0
Dvakrát celé číslo: 0
```

Zrejme vetva „else“ by mohla celkom chýbať, stačí príkaz break na úrovni if (prečo?). Otestujte tiež pre vstupné reťazce medzera alebo tabulátor a kladná alebo záporná celočíselná hodnota.
Keďže predchádzajúce ošetrenie je dosť rozsiahle, je vhodné ho umiestniť do samostatnej funkcie.

```
def vratNezaporneCeleCislo():
   while True:
      vstup = input("Zadaj nezáporné celé číslo: ")
      try:
         nezaporne_cele_cislo = int(vstup)
         if nezaporne_cele_cislo < 0:
            raise ValueError
            continue
         return nezaporne_cele_cislo        # vyvez z funkcie hodnotu premennej nezaporne_cele_cislo
      except ValueError:
         if len(vstup) == 0:
            print("Zadal si prázdny reťazec!")
         elif vstup[0] == '-':
            print("Zadal si záporné číslo!")
         else:
            print("Zadal si nedovolený znak!")
''' KONIEC FUNKCIE '''
# použitie - volanie funkcie v programe:
print("Dvakrát celé číslo:", 2*vratNezaporneCeleCislo())
```

Predchádzajúce ošetrenie bez raise ValueError:

```
def vratNezaporneCeleCislo():
    while True:
        vstup = input("Zadaj nezáporné celé číslo: ")
        try:
            nezaporne_cele_cislo = int(vstup)
            if nezaporne_cele_cislo < 0:
                print("Zadal si záporné číslo!")
                continue
            else:
                break
        except ValueError:
            if len(vstup) == 0:
                print("Zadal si prázdny reťazec!")
            else:
                print("Zadal si nedovolený znak!")
    return nezaporne_cele_cislo
```

Ukážme si ošetrenie výnimiek s viacerými názvami chýb. Majme zoznam napríklad s hodnotami 1, 2, 3, 4 a 5. Pri úlohe vypísať hodnotu zoznamu na zadanom indexe môžu nastať dva problémy a teda dve rôzne výnimky. Jeden problém už poznáme - nepodarilo sa konvertovať vstupný reťazec na celé číslo - názov chyby: ValueError. Iným problémom môže byť fakt, že zadané celé číslo nemusí byť vhodným indexom zoznamu, t.j. hodnota zoznamu so zadaným indexom neexistuje - názov chyby: IndexError.

Príklad ošetrenia:

```
zoz = [1, 2, 3, 4, 5]
while True:
    vstup = input("Vypísať hodnotu zoznamu na indexe: ")
    try:
        index = int(vstup)
        hodnota = zoz[index]
        break
    except ValueError:
        print("Nebolo zadané celé číslo!")
    except IndexError:
        print("Zadal si nedovolenú hodnotu indexu!")

print("Hodnota zoznamu na indexe {0} je {1}".format(index, hodnota))
```

```
>>>
Vypísať hodnotu zoznamu na indexe:
Nebolo zadané celé číslo!
Vypísať hodnotu zoznamu na indexe: 1o
Nebolo zadané celé číslo!
Vypísať hodnotu zoznamu na indexe: 5
Zadal si nedovolenú hodnotu indexu!
Vypísať hodnotu zoznamu na indexe: -1
Hodnota zoznamu na indexe -1 je 5
>>>
```

Odporúčanie

Pri problémových príkazoch je dobré, keď si programátor pri písaní programu vyvolaním všetkých možných výnimiek ujasní názvy chýb, ktoré môžu nastať a ošetrí ich v konštrukcii try-except, ktorá má tvar:

```
try:
        príkaz
        príkaz          problémové príkazy, pri ich vykonávaní môže nastať výnimka, a súvisiace príkazy
        ...
except názov_chyby1:
        príkaz
        príkaz          príkazy, ktoré sa majú vykonať, ak nastane chyba1
        ...
except názov_chyby2:
        príkaz
        príkaz          príkazy, ktoré sa majú vykonať, ak nastane chyba2
        ...
except ...
príkaz                  pokračovanie programu (ak nedošlo k výnimke, bloky except sa preskočia)
...
```

Konštrukcia try-except má ešte veľa rôznych variant, ktorými sa nebudeme zaoberať. Na ilustráciu niekoľko ďalších situácií a možných ošetrení výnimiek.

Ukážka: Jednoduché ošetrenie vstupu, nevypíše chybové hlásenie (except bez názvu konkrétnej chyby), len zopakuje výzvu "Koľko máš rokov: "; vek môže byť akékoľvek celé číslo, aj záporné.

```
while True:                                  # opakuj do nekonečna (alebo kým nenarazíš na break)
    vek = input("Koľko máš rokov: ")         # vek je typu str
    try:                                     # pokús sa vykonať blok try
        vek = int(vek)                       # pokús sa vykonať problémový príkaz, ak nastane
                                             # výnimka, pokračuj blokom except
        break                                # vek je „OK“, vyskoč z nekonečného cyklu while
    except:                                  # blok except bez názvu chyby
        pass                                 # nič nesprav
                                             # koniec bloku while, znova na podmienku True
print (vek)                                  # vykonaj po vystúpení z cyklu while
```

Ukážka: Výpočet podielu dvoch reálnych čísel zadaných z klávesnice v jednom riadku oddelených medzerou; využitie funkcie split (pozri reťazce). Pri analýze si treba uvedomiť, že treba ošetriť 1. nedovolený index, 2. zle zadané reálne číslo a 3. delenie nulou. Pri delení nulou nastáva chyba, ktorej názov ešte nepoznáme, ale ľahko ho získame napríklad pomocou príkazu print(5/0). Ako vidieť nižšie, názov chyby je ZeroDivisionError.

```
>>> print(5/0)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print(5/0)
ZeroDivisionError: division by zero
>>>
```

```
while True:
    vstup = input("Zadaj dve čísla: ").split()          # split vytvorí dva reťazce, vstup[0] a vstup[1]!
    try:
        podiel = float(vstup[0]) / float(vstup[1])
        break
    except IndexError:
        print("Treba zadať dve čísla!")
    except ValueError:
        print("Použitý nedovolený znak")
    except ZeroDivisionError:
        print("Nulou sa nedá deliť!")

print("Podiel = {}".format(podiel))
```

```
>>>
Zadaj dve čísla: 3.5
Treba zadať dve čísla!
Zadaj dve čísla: 3.5 2,5
Použitý nedovolený znak
Zadaj dve čísla:
Treba zadať dve čísla!
Zadaj dve čísla: 3.5 0
Nulou sa nedá deliť!
Zadaj dve čísla: 3.5 2o
Použitý nedovolený znak
Zadaj dve čísla: 3.5 2
Podiel = 1.75
>>>
```

Ukážka: Program počíta aritmetický priemer vopred neznámeho počtu reálnych čísel.

```
def vratRealneCislo(vstup):                    # funkcia s parametrom vstup typu str vracajúca reálne číslo
   while vstup != '':
      try:
         realne_cislo = float(vstup)
         return realne_cislo
      except ValueError:
         print("Chyba - opakuj zadanie reálneho čísla!")
         while True:
            vstup = input("Zadaj reálne číslo: ")
            if len(vstup) != 0:
               break
# ============================== PROGRAM ==============================
pocet_cisel, sucet_cisel = 0, 0

while True:
   vstup = input("Zadaj reálne číslo (ukončenie -> Enter): ")
   if vstup == '': break
   else:
      realne_cislo = vratRealneCislo(vstup)
      pocet_cisel += 1
      sucet_cisel += realne_cislo

# Programátorské ošetrenie delenia
try:
   print("Priemer = {0:.2f}".format(sucet_cisel/pocet_cisel))
except ZeroDivisionError:
   print("Nebolo zadané ani jedno reálne číslo!")

''' Matematické ošetrenie delenia
if pocet_cisel > 0:
   print("Priemer = {0:.2f}".format(sucet_cisel/pocet_cisel))
else:
   print("Nebolo zadané ani jedno reálne číslo!")
'''
```

```
>>>
Zadaj reálne číslo (ukončenie -> Enter): 2
Zadaj reálne číslo (ukončenie -> Enter): 3,5
Chyba - opakuj zadanie reálneho čísla!
Zadaj reálne číslo: 3.5
Zadaj reálne číslo (ukončenie -> Enter): 1o
Chyba - opakuj zadanie reálneho čísla!
Zadaj reálne číslo: 10
Zadaj reálne číslo (ukončenie -> Enter): 2,5
Chyba - opakuj zadanie reálneho čísla!
Zadaj reálne číslo: 2,5
Chyba - opakuj zadanie reálneho čísla!
Zadaj reálne číslo: 2.5
Zadaj reálne číslo (ukončenie -> Enter):
Priemer = 4.50
>>>
```