Výjimky a ošetření chyb

Pokud chceme předejít tomu, aby nám spadnul celý programu při objevení chyby, a namísto toho, nám umožnil chybu napravit – například při funkci input(), kdy očekáváme nějaký typ hodnoty, můžeme takovouto situaci ošetřit za pomocí syntaxe :

try – except

její základní tvar je:

```
try:
    '''blok príkazov'''
except MenoChyby:
    '''ošetrenie chyby'''
```

Konstrukce se skládá z dvou částí:

1) Příkazi mezi ***try*** a ***except***
2) Příkazy za except

Blok příkazů mezi ***try*** a ***except*** Python nejprve zpustí a pokud při jejich vykonávání nastane výjimka v podobě chyby, program zkontroluje příkazy za ***except*** a pokud zde najde nastalou vyjimku, zruší chybový stav a použije příkazy které zde najde (např. opakování vstupu).

Pokud při vykonávání příkazu ***try*** vyjimka nenastane, výsledek se předá programu a příkazy za ***except*** se přeskakují.

Pokud při vykonávání příkazu ***try*** nastane chyba, která naní ošetřená v příkazech ***except***, program spadne a v IDLE se vypíše chybová zpráva. Takže jen neošetřená chyba zpoůsobí spadnutí našeho programu.

Příklad možného zápisu pro vstup čísla:

```
def cislo():
    while True:
        try:
            return int(input('zadaj cislo: '))
        except ValueError:
            print('*** chybne zadane cele cislo ***')
```

Příklad zápisu zpracování více výjimek najednou:

```
def zisti(zoznam):
    while True:
        try:
            vstup = input('zadaj index: ')
            index = int(vstup)
            print('prvok zoznamu =', zoznam[index])
            break
        except ValueError:
            print('*** chybne zadane cele cislo ***')
        except IndexError:
            print('*** index mimo rozsah zoznamu ***')
```

V případě, že pro více výjimek existuje stejné řešení, je možné tyto výjimky uvést za sebe oddělené čárkou zapsané jako n-tici. Python, pak bude při zachycení některé z podmínek postupovat dle následujících instrukcí.

```
def zisti(zoznam):
    while True:
        try:
            print('prvok zoznamu =', zoznam[int(input('zadaj index: '))])
            break
        except (ValueError, IndexError):
            print('*** chybne zadany index zoznamu ***')
```

Ošetření chyb za příkazem except by mělo začínat uvedením jména chyby, tím se konkretizuje. Pokud zde neuvedeme žádnou chybu, pravidlo se bude vztahovat na všechny chyby a mi tak riskujeme, že se nedovíme o chybě, kterou bychom standardně opravili.

Jméno chyby nám prozradí Python vždy, když proběhne oznam o dané chybě a za ním je pak i uveden komentář, který by nám měl pomoci pochopit podstatu chyby.

Příklad některých jmen chyb:

```
>>> 1+'2'
    ...
    TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> 12/0
    ...
    ZeroDivisionError: division by zero
>>> x+1
    ...
    NameError: name 'x' is not defined
>>> open('')
    ...
    FileNotFoundError: [Errno 2] No such file or directory: ''
>>> [1,2,3][10]
    ...
    IndexError: list index out of range
>>> 5()
    ...
    TypeError: 'int' object is not callable
>>> ''.x
    ...
    AttributeError: 'str' object has no attribute 'x'
>>> 2**10000/1.
    ...
    OverflowError: int too large to convert to float
>>> import x
    ...
    ImportError: No module named 'x'
>>> def t(): x += 1
>>> t()
    ...
    UnboundLocalError: local variable 'x' referenced before assignment
```

Jak funguje mechanizmus výjimek:

- Python při výskytu chyby, tuto chybu nevypisuje, ale zjistí, zda se nevyskytuje v bloku příkazů try – except.
- Pokud vyjímku zde najde, použije pro ni definované příkazy.
- Pokud vyjimku zde nenajde, vyskočí z momentální funkce a zjišťuje zda volání funkce v nadřazeném příkazu nebylo chráněné příkazem try – except.
- Pokud ano, pak vykoná to, co na tomto místě v této situaci vykonat má a pokračuje dál.
- Pokud ne, pak vyskakuje o úroveň výše a znovu kontroluje zda není v bloku příkazů try – except, a takto pokračuje, až dojde na nejvyšší úroveň – tedy do dialogu IDLE – a zde nám vypíše hlášení o chybě.

Práce se soubory

při práci se soubory může nastat mnoho chyb. Nejčastější chybou je neexistující soubor.

Zde je příklad ošetření tří chyb, pro příklad načtení čísla ze souboru:

```
try:
    with open('x.txt') as subor:
        cislo = int(subor.readline())
except FileNotFoundError:
    print('*** neexistujuci subor ***')
    cislo = 0
except (ValueError, TypeError):
    print('*** prvy riadok suboru neobsahuje cele cislo ***')
    cislo = 10
```

A zde je příklad pro případ, kdy chceme jen zjistit, zda daný soubor existuje:

```
def existuje(meno_suboru):
    try:
        with open(meno_suboru):
            return True
    except (TypeError, OSError, FileNotFoundError):
        return False
```

Vyvolání výjimky - raise

někdy se nám může hodit vyvolat chybovou hlášku (oznam o chybě) v IDLE i přes to, že jsme ji zachytili příkazy ***try*** - ***except***. Zejména při lazení programu.
Pro tento účel má Python příkaz ***raise***, který má několik variant.
První z nich si můžeme, prohlédnou v příkladu pro funkci číslo, kdy funkce nám nejprve nabídne 3 pokusí na správné zadání a pokud se nám to nepodaří, zavolá definovanou chybu (v tomto příkladu ValueError):

```
def cislo():
    pokus = 0
    while True:
        try:
            return int(input('zadaj cislo: '))
        except ValueError:
            pokus += 1
            if pokus >= 3:
                raise
            print('*** chybne zadane cele cislo ***')
```

Pomocí příkazu ***raise*** můžeme vyvolat nejen zachycenou vyjimku, ale i jakoukoliv jiný chybový oznam a to i s vlastním komentářem:

```
raise ValueError('chybne zadane cele cislo')
raise ZeroDivisionError('delenie nulou')
raise TypeError('dnes sa ti vobec nedari')
```

Příklad s metodou **index()**
metoda ***index(),*** při nenalezení hodnoty v seznamu vypíše výjimku ValueError, která ale přímo nespecifikuje, zda položka nebyla nalezena, či zda nešlo o jiný případ této výjimky:

```
def nahrad(zoznam, h1, h2):
    if h1 in zoznam:
        i = zoznam.index(h1)
        zoznam[i] = h2
```

Pomocí try – except, toto můžeme vyřešit efektivněji:

```
def nahrad(zoznam, h1, h2):
    try:
        i = zoznam.index(h1)
        zoznam[i] = h2
    except ValueError:
        pass
```

Tohoto pak můžeme využít i v prohledávání dvojrozměrné tabulky, kde bude hledat první výskyt dané hodnoty a po jejím nalezení nám vrátí číslo řádku a sloupce:

```
def hladaj(zoznam, hodnota):
    for r in range(len(zoznam)):
        for s in range(len(zoznam[r])):
            if zoznam[r][s] == hodnota:
                return r, s
    raise ValueError(f'{hodnota!r} is not in list')
```

Případně můžeme použít i verzi s ***enumerate:***

```
def hladaj(zoznam, hodnota):
    for r, riadok in enumerate(zoznam):
        for s, prvok in enumerate(riadok):
            if prvok == hodnota:
                return r, s
    raise ValueError(f'{hodnota!r} is not in list')
```

Případně zápis s prohledávání skrze indexy:

```
def hladaj(zoznam, hodnota):
    for r, riadok in enumerate(zoznam):
        try:
            s = riadok.index(hodnota)
            return r, s
        except ValueError:
            if r == len(zoznam)-1:     # posledný prechod for-cyklom
                raise
```

Vytváření vlastních výjimek
Když chceme vytvořit vlastní typ výjimek, musíme vytvořit novou třídu, odvozenou od základní třídy **Exception:**

```
class MojaChyba(Exception): pass
```

Pokud použijeme pouze příkaz pass, znamená to, že oproti základní třídě Exception nedefinujeme nic nového. V následujícím příkladu si můžeme prohlédnout využití vlastní třídy výjimek pro ošetření možných chyb. V tomto případě zadání hesla a nebo špatné částky vkladu:

```
class ChybnaTransakcia(Exception): pass

class UcetHeslo:
    def __init__(self, meno, heslo, suma=0):
        self.meno, self.heslo, self.suma = meno, heslo, suma

    def kontrola(self):
        if self.heslo and self.heslo != input(f'heslo pre {self.meno}? '):
            raise ChybnaTransakcia('chybne heslo pre pristup k uctu ' + self.meno)

    def vklad(self, suma):
        self.kontrola()
        try:
            if suma <= 0:
                raise TypeError
            self.suma += suma
        except TypeError:
            raise ChybnaTransakcia('chybne zadana suma pre vklad na ucet ' + self.
meno)
```

Kontrola pomocí příkazu assert
assert je příkaz, který slouží na kontrolu nějaké podmínky. Pokud se tato podmínka nesplní vyvolá se AssertionError i s uvedeným komentářem.
Tvar tohoto příkazu je:

```
assert podmienka, 'komentár'
```

Toto se často používá při ladění, kdy potřebujeme mít jistotu, že určitá podmínka byla splněna, nebo ne:

```
def podiel(p1, p2):
    assert isinstance(p1, int), 'prvy parameter nie je cele cislo'
    assert isinstance(p2, int), 'druhy parameter nie je cele cislo'
    assert p2 != 0, 'neda sa delit nulou'
    return p1 // p2

>>> podiel(2.2, 3)
    ...
    AssertionError: prvy parameter nie je cele cislo
>>> podiel(22, 3.3)
    ...
    AssertionError: druhy parameter nie je cele cislo
>>> podiel(22, 0)
    ...
    AssertionError: neda sa delit nulou
```