

Výnimky

Poznámky k téme

Keď počas behu programu nastane chyba – delenie nulou, neplatný vstup, chýbajúci súbor –, Python vyvolá **výnimku**. Ak ju nikto nezachytí, program skončí s chybovým hlásením. V tejto téme sa naučíš výnimky **zachytávať** a rozumne na ne reagovať: požiadať o nový vstup, použiť náhradnú hodnotu alebo vypísať zrozumiteľnú správu. Naučíš sa aj vyvolať vlastnú výnimku, keď funkcia dostane nezmyselné údaje.

Čo sa naučíš

- zachytiť výnimku blokom `try – except` a pochopiť, ako sa program pri chybe vykonáva,
- rozlišovať typy výnimiek a zachytávať iba tie, ktoré očakávaš,
- používať vetvy `else` a `finally`,
- vyvolať vlastnú výnimku príkazom `raise`,
- napísať kontrolu vstupu, ktorá sa pýta, kým nedostane správnu hodnotu.

1 Čo je výnimka

```
vek = int(input("Vek: "))  
print("O rok budeš mať", vek + 1)
```

Výstup (program skončí chybou):

```
Vek: dva  
Traceback (most recent call last):  
  File "program.py", line 1, in <module>  
    vek = int(input("Vek: "))  
          ~~~~~  
ValueError: invalid literal for int() with base 10: 'dva'
```

`int("dva")` nevie urobiť z textu číslo, a tak vyvolá výnimku `ValueError`. Nik ju nezachytil, takže Python program ukončil a vypísal hlásenie (traceback). Výnimka nesie **typ** (`ValueError`) a **správu** (`invalid literal for int() with base 10: 'dva'`).

2 try a except

Kód, v ktorom môže nastať chyba, dáme do bloku `try`. Do bloku `except` napíšeme, čo sa má stať, keď chyba nastane:

```
try:
    vek = int(input("Vek: "))
    print("O rok budeš mať", vek + 1)
except ValueError:
    print("To nie je celé číslo.")
print("Program pokračuje.")
```

Výstup:

```
Vek: dva
To nie je celé číslo.
Program pokračuje.
```

Ako sa vykonáva:

1. Vykonávajú sa príkazy v bloku `try`.
2. Ak nenastane chyba, blok `except` sa **preskočí**.
3. Ak nastane chyba, **zvyšok bloku try sa preskočí** a vykoná sa `except` – ak zodpovedá typ výnimky.
4. Potom program normálne pokračuje za celou konštrukciou.

Pre vstup 17 by sa vypísalo „O rok budeš mať 18“ a „Program pokračuje.“

2.1 Konkrétny typ výnimky

Za `except` píšeme **typ** výnimky, ktorú chceme zachytiť. Iné výnimky zachytené nebudú – a to je dobre: nechceme potichu „prehltnúť“ chyby, ktoré sme nečakali (napríklad preklep v mene premennej).

Viac typov môžeme ošetriť viacerými vetvami `except`:

```
try:
    a = int(input("Delenec: "))
    b = int(input("Deliteľ: "))
    print("Podiel:", a / b)
except ValueError:
    print("Zadaj celé čísla.")
except ZeroDivisionError:
    print("Nulou deliť nemožno.")
```

Výstup:

```
Delenec: 10
Deliteľ: 0
Nulou deliť nemožno.
```

Rovnaké ošetrenie pre viac typov: `except (ValueError, ZeroDivisionError):`.

2.2 Správa výnimky: as

Zápis `except Typ as e` uloží výnimku do premennej `e`; `str(e)` alebo `print(e)` vypíše jej správu:

```
try:
    cisla = [1, 2, 3]
    print(cisla[10])
except IndexError as e:
    print("Chyba:", e)
```

Výstup:

```
Chyba: list index out of range
```

3 else a finally

Konštrukcia môže mať ešte dve vetvy:

- `else` – vykoná sa, **ak nenastala** žiadna výnimka,
- `finally` – vykoná sa **vždy**, s chybou aj bez nej (napríklad na upratanie, zatvorenie súboru, záverečnú správu).

```
def vydel(a, b):
    try:
        vysledok = a / b
    except ZeroDivisionError:
        print(" delenie nulou")
    else:
        print(" výsledok:", vysledok)
    finally:
        print(" hotovo")

print("vydel(10, 4):")
vydel(10, 4)
print("vydel(1, 0):")
vydel(1, 0)
```

Výstup:

```
vydel(10, 4):
    výsledok: 2.5
    hotovo
vydel(1, 0):
    delenie nulou
    hotovo
```

Do `try` dávaj iba riadky, v ktorých môže nastať očakávaná chyba. Pokračovanie, ktoré má prebehnúť iba pri úspechu, patrí do `else`.

4 Typy výnimiek

Výnimka	Kedy nastane	Príklad
<code>ValueError</code>	hodnota má správny typ, ale nezmyselný obsah	<code>int("abc"), int("3.5")</code>
<code>TypeError</code>	operácia s nevhodným typom	<code>"a" + 1, len(5)</code>
<code>ZeroDivisionError</code>	delenie nulou	<code>5 / 0, 5 % 0</code>
<code>IndexError</code>	index mimo zoznamu alebo reťazca	<code>[1, 2][5]</code>
<code>KeyError</code>	klúč v slovníku neexistuje	<code>{"a": 1}["b"]</code>
<code>NameError</code>	neznáme meno (preklep, nepriradená premenná)	<code>print(x)</code>
<code>AttributeError</code>	hodnota nemá takú metódu	<code>(5).upper()</code>
<code>FileNotFoundError</code>	súbor neexistuje	<code>open("nie.txt")</code>
<code>RecursionError</code>	príliš hlboká rekurzia	funkcia bez základného prípadu

```
>>> int("3.5")
ValueError: invalid literal for int() with base 10: '3.5'
>>> len(5)
TypeError: object of type 'int' has no len()
>>> {"a": 1}["b"]
KeyError: 'b'
>>> "text".upper(1)
TypeError: str.upper() takes no arguments (1 given)
```

Všetky bežné výnimky sú „druhom“ výnimky `Exception`. Zápis `except Exception:` preto zachytí takmer všetko – používaj ho iba výnimočne a nikdy nepíš samotné `except:` bez typu. Zachytil by aj chyby, ktoré chceš vidieť a opraviť.

5 Vyvolanie výnimky: raise

Výnimku môžeme vyvolať aj sami príkazom `raise`. Robí sa to vo funkcii, ktorá dostane nezmyselné údaje a nevie s nimi rozumne pokračovať – namiesto chybného výsledku radšej jasne oznámi problém:

```
def odmocnina(x):
    if x < 0:
        raise ValueError("odmocnina zo záporného čísla")
    return x ** 0.5

print(odmocnina(16))
try:
    print(odmocnina(-4))
except ValueError as e:
    print("Chyba:", e)
```

Výstup:

```
4.0
Chyba: odmocnina zo záporného čísla
```

Kto funkciu volá, sa môže rozhodnúť, či výnimku zachytí. Ak ju nezachytí, program skončí so správou, ktorú sme napísali:

```
def nastav_vek(vek):  
    if not 0 <= vek <= 150:  
        raise ValueError(f"neplatný vek: {vek}")  
    return vek  
  
nastav_vek(200)
```

Výstup (program skončí chybou):

```
Traceback (most recent call last):  
  File "program.py", line 6, in <module>  
    nastav_vek(200)  
  File "program.py", line 3, in nastav_vek  
    raise ValueError(f"neplatný vek: {vek}")  
ValueError: neplatný vek: 200
```

6 Kontrola vstupu

Typické použitie výnimiek: pýtať sa na vstup **dovtedy, kým nie je správny**. Cyklus `while True` skončí `break` až po úspešnom prevode:

```
while True:  
    try:  
        n = int(input("Zadaj celé číslo: "))  
        break  
    except ValueError:  
        print("Neplatný vstup, skús to znova.")  
print("Načítané číslo:", n)
```

Výstup:

```
Zadaj celé číslo: sedem  
Neplatný vstup, skús to znova.  
Zadaj celé číslo: 7.5  
Neplatný vstup, skús to znova.  
Zadaj celé číslo: 7  
Načítané číslo: 7
```

Často treba skontrolovať aj **rozsah**. Pohodlné je zabaliť kontrolu do funkcie, ktorú použiješ v rôznych programoch:

```
def nacistaj_cislo(vyzva, od, do):
    while True:
        try:
            n = int(input(vyzva))
        except ValueError:
            print("  Zadaj celé číslo.")
            continue
        if od <= n <= do:
            return n
        print(f"  Číslo musí byť od {od} do {do}.")

znamka = nacistaj_cislo("Známka (1 – 5): ", 1, 5)
print("Zapísaná známka:", znamka)
```

Výstup:

```
Známka (1 – 5): 0
  Číslo musí byť od 1 do 5.
Známka (1 – 5): abc
  Zadaj celé číslo.
Známka (1 – 5): 11
  Číslo musí byť od 1 do 5.
Známka (1 – 5): 4
Zapísaná známka: 4
```

Rozsah kontrolujeme obyčajnou podmienkou – výnimky použijeme iba tam, kde chybu nevieme ľahko predvídať (prevod textu na číslo, otvorenie súboru).

7 Riešené úlohy

7.1 Bezpečný priemer

Zadanie: Napíš funkciu `priemer(cisla)`, ktorá vráti priemer zoznamu. Pre prázdny zoznam nech vyvolá `ValueError` so zrozumiteľnou správou. Vyskúšaj ju na správnom aj prázdnom zozname.

```
def priemer(cisla):
    if len(cisla) == 0:
        raise ValueError("priemer prázdneho zoznamu neexistuje")
    return sum(cisla) / len(cisla)

for z in [[4, 8, 6], []]:
    try:
        print(z, "→", priemer(z))
    except ValueError as e:
        print(z, "→ chyba:", e)
```

Výstup:

```
[4, 8, 6] → 6.0
[] → chyba: priemer prázdneho zoznamu neexistuje
```

7.2 Súbor s chybnými riadkami

Zadanie: V súbore je na každom riadku jedno číslo, niektoré riadky sú však chybné. Spočítaj súčet platných čísel a vypíš, ktoré riadky boli chybné.

Súbor `merania.txt`:

```
12
7.5
chyba
-3

1e2
```

Riešenie: Každý riadok skúsime previesť na číslo. Chybný riadok nezastaví celý program – iba si ho poznačíme a pokračujeme ďalším. Zachytávame aj `FileNotFoundError`, ak by súbor chýbal.

```
sucet = 0
chybne = []
try:
    with open("merania.txt", encoding="utf-8") as f:
        for cislo_riadku, riadok in enumerate(f, start=1):
            try:
                sucet += float(riadok)
            except ValueError:
                chybne.append(cislo_riadku)
except FileNotFoundError:
    print("Súbor merania.txt neexistuje.")
else:
    print("Súčet:", sucet)
    print("Chybné riadky:", chybne)
```

Výstup:

```
Súčet: 116.5
Chybné riadky: [3, 5]
```

Prázdny riadok (riadok 5) je tiež chybný – `float("\n")` nie je číslo. `1e2` je platný zápis čísla 100.

7.3 Hádaj číslo s kontrolou vstupu

Zadanie: Počítač si myslí číslo od 1 do 50. Program sa pýta na tipy; neplatný vstup (text, číslo mimo rozsahu) nezapočíta ako pokus.

```
import random

tajne = random.randint(1, 50)
pokusy = 0
while True:
    try:
        tip = int(input("Tip: "))
    except ValueError:
        print("To nie je číslo.")
        continue
    if not 1 <= tip <= 50:
        print("Iba čísla od 1 do 50.")
        continue
    pokusy += 1
    if tip < tajne:
        print("Viac.")
    elif tip > tajne:
        print("Menej.")
    else:
        print("Uhádnuté! Počet pokusov:", pokusy)
        break
```

Výstup:

```
Tip: 25
Viac.
Tip: hm
To nie je číslo.
Tip: 40
Menej.
Tip: 99
Iba čísla od 1 do 50.
Tip: 33
Viac.
Tip: 37
Uhádnuté! Počet pokusov: 4
```

8 Časté chyby

- **Samotné except:** (alebo `except Exception:`) zachytí aj chyby, ktoré zachytiť nechceš – napríklad preklep v mene premennej – a program sa tvári, že všetko je v poriadku.
- **Príliš veľký blok try:** keď je v ňom veľa riadkov, nevieš, ktorý chybu spôsobil. Do `try` daj iba rizikové riadky.
- **Výnimka namiesto podmienky:** to, čo vieš jednoducho overiť (`if b != 0`, `if x in slovník`), over podmienkou.
- **Chýbajúci break** v `while True` cykle kontroly vstupu – cyklus sa nikdy neskončí.
- **Zabudnuté, že program po except pokračuje** – premenná, ktorá sa v `try` nestihla priradiť, potom neexistuje.

Zhrnutie

- Výnimka = chyba počas behu; nezachytená výnimka ukončí program.
- `try`: rizikový kód, `except Typ`: ošetrovanie; pri chybe sa zvyšok `try` preskočí.
- Zachytávajú konkrétne typy (`ValueError`, `ZeroDivisionError`, ...), nie všetko naraz.
- `except Typ as e` – správa výnimky; `else` – keď chyba nenastala; `finally` – vždy.
- `raise ValueError("správa")` vyvolá výnimku, keď funkcia dostane neplatné údaje.
- Kontrola vstupu: `while True` + `try` + `break` po úspešnom prevode; rozsah over podmienkou.

Úlohy na precvičenie

1. Načítaj dve čísla a vypíš ich podiel; ošetri neplatný vstup aj delenie nulou.
2. Napíš funkciu `nacitaj_float(vyzva)`, ktorá sa pýta, kým nedostane desatinné číslo (aj s desatinnou čiarkou).
3. Napíš funkciu `obsah_obdĺznika(a, b)`, ktorá pri zápornej strane vyvolá `ValueError`.
4. Zo slovníka cien vypíš cenu zadaného tovaru; ak tovar chýba, ošetri `KeyError` a vypíš správu.
5. Načítaj meno súboru; ak neexistuje, vypíš správu a požiadaj o meno znova.

Úlohy k tejto téme nájdeš na learnwithmente.com v časti Informatika → Python.