

Ko pride do napake Python sestavi objekt razreda `Exception` (po slovensko, izjema), oziroma, točneje, enega od razredov, izpeljanih iz `Exception`. Vsak razred predstavlja svojo vrsto napake. Teh je ogromno, pa še nove si lahko izmišljamo. Ko sestavi ta objekt, ga vrže (`throw`) oz. sproži (`raise`). V Pythonu se uporablja slednji izraz, `raise`, v Cju podobnih jezikih pa prvi, `throw`. Če vržene izjeme nihče ne ulovi, se ta izpiše, kot smo vajeni. Tule se bomo naučili loviti, potem pa še metati izjeme.

### Lovljenje napak

Če predvidevamo, da lahko v določenem delu kode pride do napake (kakor bomo tule po domače govorili izjemam), ga zapremo v blok `try-except` (in ne `try-catch`, ko bi se reklo v C-ju in žlahti). Takole je videti.

```
visina = input("Višina: ")
teza = input("Teža: ")
try:
    visina = float(visina)
    teza = float(teza)
except ValueError:
    print(f"Napačen vnos")
```

Višina: velik

Teža: težak

Napačen vnos

Besedi `try` sledi dvopičje in, kot smo vajeni, zamaknjen blok, ki seveda lahko vsebuje tudi več kot eno vrstico. Nato pride `except`, kjer, če smo vljudni, povemo, kakšne vrste napako pričakujemo. Znotraj bloka `except` storimo kaj pametnega - izpišemo kakšno obvestilo ali kaj podobnega.

Blokov `except` je lahko tudi več, če pričakujemo več različnih napak.

Če nismo vljudni, izpustimo vrsto napake in pišemo samo `except:`. Tak `except` ujame vse napake.

Ko pride do napake, se izvajanje bloka prekine. Python poišče prvi blok `except`, ki se ujema z vrsto napake in začne izvajati kodo v njem.

### Lovi samo predvidene napake

Tule bi se morda zdela boljša ideja:

```
try:
    visina = input("Višina: ")
    teza = input("Teža: ")
    visina = float(visina)
    teza = float(teza)
    bmi = teza / visina ** 2
```

```

    print(f"Indeks telesne teže: {bmi:.2f}")
except ValueError:
    print(f"Napačen vnos")

```

Višina: 1.85

Teža: 76

Indeks telesne teže: 22.21

Vendar ni, saj bi bila potem dobra ideja tudi to:

```

try:
    with open("stevilke.txt") as f:
        vrstic = 0
        v = 0
        for vrstica in f:
            v += int(vrstica)
            vrstic += 1
        povp = v / vrstic
        print(f"Povprečje: {povp:.2f}")
except IOError:
    print("Datoteka ne obstaja")
except ValueError:
    print("Napačna vrednost v datoteki")
except ZeroDivisionError:
    print("Datoteka je prazna")
except:
    print("Neznana napaka")

```

Povprečje: 6.33

Pa ni. To je zelo slaba ideja.

`except` ne sme loviti preveč na široko. `except ValueError` smo napisali, da prestrežemo morebitno napako v `int(vrstica)`, torej naj lovi le napako v tej vrstici. Ideja `try-except` ni loviti napak na počez. Če neke napake nismo predvideli, je boljše, da je ne prestrežemo. Program ne sme teči naprej, kot da je vse v redu, če ni. Še manj sme, recimo, izpisati, da datoteka vsebuje napačno vrednost, če problem morda sploh ni v datoteki, temveč je napaka `ValueError` priletela od kod drugod.

Loviti z `except:` pa je sploh skoraj prepovedano. To storite samo, ko kličete neko (tujo) kodo, v kateri se lahko zgodi karkoli.

Pravilneje bi bilo torej tako:

```

try:
    with open("stevilke.txt") as f:
        vrstic = 0
        v = 0
        for vrstica in f:

```

```

    try:
        v += int(vrstica)
    except ValueError:
        print("Napačna vrednost v datoteki")
        continue
    vrstic += 1
try:
    povp = v / vrstic
except ZeroDivisionError:
    print("Datoteka je prazna")
    povp = 0 # da se ima v naslednji vrtici kaj izpisati ...
print(f"Povprečje: {povp:.2f}")
except IOError:
    print("Datoteka ne obstaja")

```

Povprečje: 6.33

`except IOError` je neroden, vendar neizogiben, če želimo datoteko odpreti z `with`. In ja, zoprni je tudi zato, ker lahko (vsaj načelno) pride do `IOError` tudi ob branju datoteke.

### Napakam se raje izogni, kot da jih loviš

Edina napaka, ki jo malo težje predvidimo, je `ValueError` v `int(vrstica)`. Vse ostale lahko predvidimo in preprečimo.

```

import os

if not os.path.exists("stevilke.txt"):
    print("Datoteka ne obstaja")
else:
    with open("stevilke.txt") as f:
        vrstic = 0
        v = 0
        for vrstica in f:
            try:
                v += int(vrstica)
            except ValueError:
                print("Napačna vrednost v datoteki")
                continue
            vrstic += 1
        if vrstic == 0:
            print("Datoteka je prazna")
        else:
            povp = v / vrstic
            print(f"Povprečje: {povp:.2f}")

```

Povprečje: 6.33

## Če ni napake ...

Vrnimo se k indeksu telesne teže. Rekli smo, da je tole slaba ideja.

```
try:
    visina = input("Višina: ")
    teza = input("Teža: ")
    visina = float(visina)
    teza = float(teza)
    bmi = teza / visina ** 2
    print(f"Indeks telesne teže: {bmi:.2f}")
except ValueError:
    print(f"Napačen vnos")
```

Višina: 1.85

Teža: 76

Indeks telesne teže: 22.21

Kako to popraviti? Imamo namreč problem.

```
visina = input("Višina: ")
teza = input("Teža: ")
try:
    visina = float(visina)
    teza = float(teza)
except ValueError:
    print(f"Napačen vnos")
bmi = teza / visina ** 2
print(f"Indeks telesne teže: {bmi:.2f}")
```

Višina: 1.85

Teža: 76

Indeks telesne teže: 22.21

Problem je v tem: če se zgodi napaka, potem sta `visina` in `teza` se vedno niza. Strašno nerodno bi bilo pisati:

```
visina = input("Višina: ")
teza = input("Teža: ")
try:
    visina = float(visina)
    teza = float(teza)
except ValueError:
    print(f"Napačen vnos")
    visina = teza = None

if visina is not None:
    bmi = teza / visina ** 2
```

```

    print(f"Indeks telesne teže: {bmi:.2f}")
Višina: 1.85
Teža: 76
Indeks telesne teže: 22.21
except-u sme slediti else. Kar napišemo znotraj else, se izvede le, če ni prišlo
do izjeme.
visina = input("Višina: ")
teza = input("Teža: ")
try:
    visina = float(visina)
    teza = float(teza)
except ValueError:
    print(f"Napačen vnos")
else:
    bmi = teza / visina ** 2
    print(f"Indeks telesne teže: {bmi:.2f}")

```

## Končno

Končno lahko napišemo tudi `finally`. Ta lahko sledi `exceptu` oz. `else-u`, lahko pa imamo celo samo `try` in `finally`.

Kar se nahaja znotraj bloka `finally` se izvede v vsakem primeru - najsi je prišlo do izjeme ali ne.

```

try:
    1 / 0
finally:
    print("Zgodilo se je nekaj izjemnega.")

```

Zgodilo se je nekaj izjemnega.

```

-----
ZeroDivisionError                                Traceback (most recent call last)
Cell In[25], line 2
      1 try:
----> 2     1 / 0
      3 finally:
      4     print("Zgodilo se je nekaj izjemnega.")

```

ZeroDivisionError: division by zero

Napaka se izpiše, predtem pa se izvede tudi blok `finally`.

## Več o izjemi

Objekt, ki predstavlja izjemo, lahko vsebuje različne podatke, povezane z njo. Če nas zanimajo, `except <vrsta-izjeme>` nadaljujemo z `as <ime>`.

```
try:
    open("te-datoteke-ni.txt")
except IOError as napaka:
    print(napaka)
    print(napaka.filename)
    print(napaka.strerror)
```

```
[Errno 2] No such file or directory: 'te-datoteke-ni.txt'
te-datoteke-ni.txt
No such file or directory
```

`IOError` je zelo zgovoren, drugi manj. Vsaj vrsto napake in sporočilo pa lahko vedno izvemo.

## Proženje napak

Včasih želimo napako sprožiti sami. To je preprosto. Zapomniti si moramo le, da v Pythonu tega ne storimo s `throw`, kot v jezikih s Cjevsko sintakso, temveč z `raise`.

```
raise ValueError("Čudna vrednost.")
```

```
-----
ValueError                                Traceback (most recent call last)
Cell In[37], line 1
----> 1 raise ValueError("Čudna vrednost.")
```

```
ValueError: Čudna vrednost.
```