OSNOVE PROGRAMIRANJA U PAJTONU

PREDAVANJE 12:
OBRADA GREŠAKA U PROGRAMU

Miloš Kovačević

Đorđe Nedeljković

Marija Petronijević

Dušan Isailović

SADRŽAJ PREDAVANJA

- Trag greške i izuzeci
- Defanzivno programiranje
- Kritične sekcije
- Try – except struktura

GREŠKE U PROGRAMU

Greške u programu: sintaksne i semantičke.

Sintaksne greške otkriva interpreter pre pokretanja programa pa se lako otkrivaju i ispravljaju.

Semantičke greške su logički propusti koji prekidaju rad programa, ili još gore, prolaze neopažano!

U programiranju (Pajtonu), programerima stoje na raspolaganju dva mehanizma za borbu protiv grešaka:

Da se greška predupredi pre nego što nastane – defanzivno programiranje

Da se izvrši procedura oporavka, po pojavi greške – obrada izuzetaka.

MARFIJEV ZAKON

Svet računara i programiranja – svet u kome vlada Marfijev zakon.

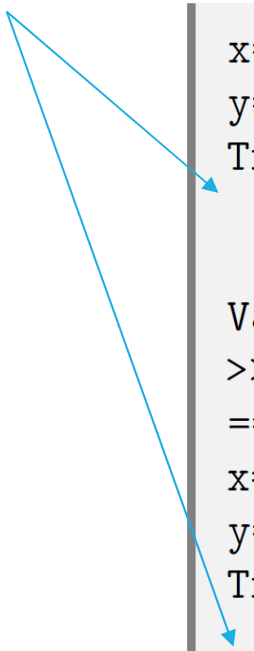
Marfijev zakon: sve što može da se desi loše, desiće se.

Programeri treba da predvide
da korisnici ne poštuju specifikaciju za učazne veličine,
kao i da delovi računarskog sistema mogu otkazati u toku rada programa!

TRAG GREŠKE

```
1 x = int(input('x= '))
2 y = int(input('y= '))
3 print('x/y=', x/y)
```

Trag greške – informacija o mestu nastanka greške

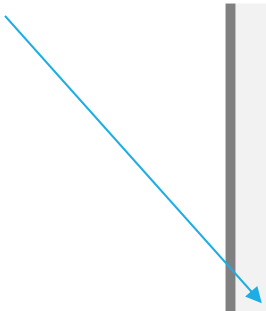


```
x= 2
y= 3.1
Traceback (most recent call last):
  File "C:/programi/p8_1a.py", line 2, in <module>
    y = int(input('y= '))
ValueError: invalid literal for int() with base 10: '3.1'
>>>
==== RESTART: C:/programi/p8_1a.py ====
x= 10
y= 0
Traceback (most recent call last):
  File "C:/programi/p8_1a.py", line 3, in <module>
    print('x/y=', x/y)
ZeroDivisionError: division by zero
```

OBJEKAT GREŠKE

```
1  def podeli(x,y):
2      return x/y
3
4  x, y = 1, 0
5  z = podeli(x,y) # deljenje nulom
6  print('x/y', z)
```

Objekat greške – sadrži informaciju o tipu i tragu greške



```
Traceback (most recent call last):
  File "C:/programi/p8_1b.py", line 5, in <module>
    z = podeli(x,y)
  File "C:/programi/p8_1b.py", line 2, in podeli
    return x/y
ZeroDivisionError: division by zero
```

Objekti greške pripadaju različitim klasama:
klasa ZeroDivisionError odnosi se na deljenje nulom.

ČESTE KLASE GREŠAKA U PAJTONU

tip	dešava se
`ValueError`	kada je argument operacije ili funkcije ispravnog tipa, ali ima pogrešnu vrednost
`NameError`	prilikom pokušaja pristupanja objektu preko nepostojećeg imena
`TypeError`	prilikom pokušaja operacije ili funkcije, sa argumentom pogrešnog tipa
`IndexError`	prilikom pristupanja sekvenci objekata, kada je navedeni indeks van trenutnog opsega
`OverflowError`	prilikom izračunavanja realnog izraza, kada je vrednost prevelika
`ZeroDivisionError`	prilikom deljenja sa nulom
`IOError`	pri neuspeloj ulazno-izlaznoj operaciji (na primer, pri čitanju iz nepostojeće datoteke)
`KeybordInterrupt`	kada korisnik sa tastature unese sekvencu <Ctrl>-<C>

Svi tipovi osim poslednjeg nastaju usled propusta u radu programa.

DEFANZIVNO PROGRAMIRANJE

Zasniva se na principu bolje sprečiti nego lečiti

Pre kritične obrade, proverava se da li sve potrebne veličine odgovaraju po tipu i vrednosti želejnoj specifikaciji.

Problem 8.1 — Deljenje u defanzivi. Uneti sa tastature dva cela broja i ispisati njihov količnik. U slučaju neadekvatnog unosa, obavestiti korisnika primerenom porukom. ■

Da li su brojevi celi?. Da li je delilac različit od nule?

DEFANZIVNO DELJENJE

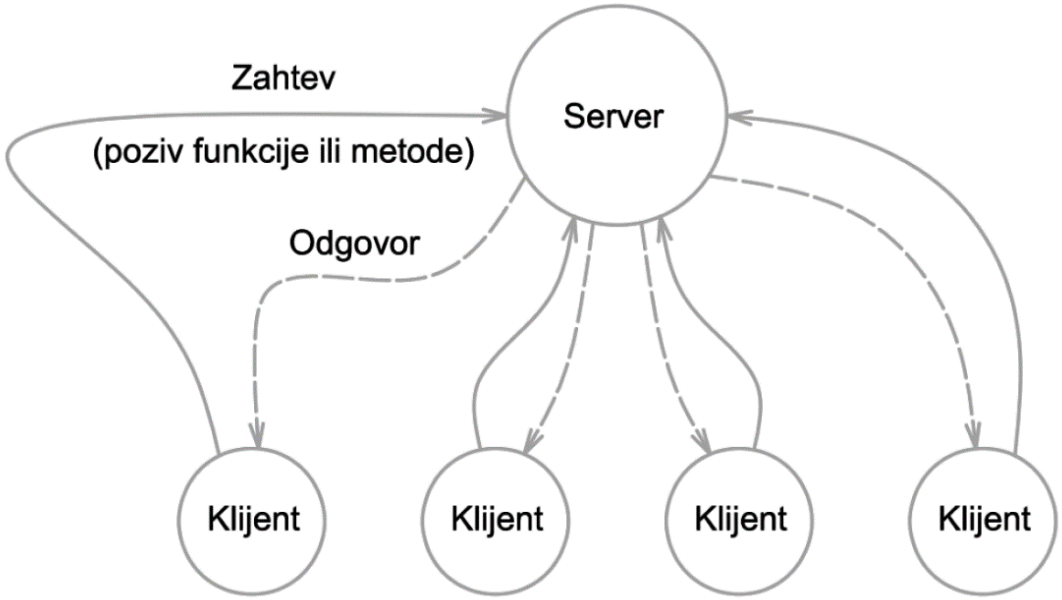
```
def je_ceo(x):

    if len(x) == 0:
        return False
    elif x[0] == '-':
        return x[1:].isdigit()
    else:
        return x.isdigit()

x = input('x= ')
y = input('y= ')
if je_ceo(x) and je_ceo(y):
    x, y = int(x), int(y)
    if y != 0:
        print('x/y=', x/y)
    else:
        print('y ne sme biti 0!')
else:
    print('uneti brojevi moraju biti celi!')
```

KLIJENT-SERVER MODEL

Klijent je program ili njegov deo (npr. funkcija) koji za obavljanje svog posla potražuje uslugu od nekog drugog programa (ili dela programa).

Server je program (ili njegov deo) koji pruža uslugu klijentu.

Primeri: internet čitač i web server,
program koji koristi funkciju iz nekog modula.



DEFANZIVNO PROGRAMIRANJE – SERVER ZA DELJENJE

```
# server: deli_server.py
# proverava da li je tekst validan ceo broj
def je_ceo(x):
    # .... kao u prethodnom listingu

# serverska funkcija
# ako je status: 0 - sve ok, 1 - deljenje sa 0
#                2 - x ne valja, 3 - y ne valja
def deli(x,y):
    if je_ceo(x):
        if je_ceo(y):
            x, y = int(x), int(y)
            if y != 0:
                return (0, x/y)
            else:
                return (1, 0)
        else:
            return (3, 0)
    else:
        return (2, 0)
```

Vraća informaciju (status) o rezultatu obrade!

DEFANZIVNO PROGRAMIRANJE – DELI KLIJENT

```
# test klijent: deli_klijent.py
import deli_server as server

# prekida rad sa ctrl-c
while True:
    x, y = input('x '), input('y ')
    status, z = server.deli(x, y)
    if status == 0:
        print ('x/y', z)
    elif status == 1:
        print('deljenje nulom!')
    elif status == 2:
        print('ne valja x!')
    else:
        print('ne valja y!')
```

Primetiti da se rad može prekinutu sa ctrl-c (`KeyboardInterrupt`)

OBRADA IZUZETAKA

Greške pri izvršavanju nazivaju se i izuzeci
(pretpostavka je da se retko dešavaju!)

Nedostaci defanzvnog pristupa:

- za veliki broj ulaza teško je sprovesti sve moguće provere
- program usporava sa radom zbog retkih problematičnih ulaza
- često nije moguće predupredити greške (otkaz mrežne veze ili diska)

Obrada izuzetaka zasniva se na principu hvatanja greške i pokretanja procedure oporavka.

KONTROLNA STRUKTURA `try except`

```
# obrada izuzetaka
# server
def deli(x,y):

    poruka, kol = None, 0
    try:
        kol = int(x)/int(y)
    except:
        poruka = 'Greška u podacima'
    return (poruka, kol)

# test klijent (ctrl-c za prekid)
while True:

    x = input('x = ')
    y = input('y = ')
    p, k = deli(x, y)
    if p == None:
        print('x/y =', k)
    else:
        print(p)
```

Kritična sekcija – deo programa u kome može doći do greške

Procedura oporavka (samo u slučaju greške)

Preporučuje se obrada svake greške ponaosob!

```
def deli(x,y):

    poruka, kol = None, 0
    try:
        kol = int(x)/int(y)
    except ValueError:
        poruka = 'x i y moraju biti celi!'
    except ZeroDivisionError:
        poruka = 'y ne sme biti 0!'
    except:
        poruka = 'nepoznata greška!'
    return (poruka, kol)

# test klijent (ctrl-c za prekid)
try:
    while True:
        x = input('x = ')
        y = input('y = ')
        p, k = deli(x, y)
        if p == None:
            print('x/y =', k)
        else:
            print(p)
except KeyboardInterrupt:
    print('zdravo!')
```

BEZUSLOVNO IZVRŠAVANJE PRI OBRADI IZUZETAKA

```
def f(x):

    try:
        print(int(x))
    except ValueError:
        print('nije ceo broj!')
    finally:
        print('gotovo')

def g(x):

    try:
        print(int(x))
    except ValueError:
        print('nije ceo broj!')

    print('gotovo')
```

```
def h(x):

    try:
        print(int(x))
        return
    except ValueError:
        print('nije ceo broj!')
    finally:
        print('gotovo')
```

Blok finally obavezno se izvršava u svakoj varijanti izvršavanja funkcije!

EKSPLICITNO PROSLEĐIVANJE IZUZETAKA

Često server nema dovoljno informacija da obradi izuzetak.

Zato server prosleđuje izuzetak klijentu, koji bolje zna šta da radi u slučaju greške – naredba `raise`.

```
import time as t
def deli(x,y):

    try:
        return int(x)/int(y)
    except:
        print('serverski dnevnik:', t.asctime(), 'greška!')
        raise
```

Prosleđivanje greške onome ko poziva funkciju vidi šta ćeš sa greškom

EKSPLICITNO PROSLEĐIVANJE IZUZETAKA

```
# test klijent (ctrl-c za prekid)
try:
    while True:
        x = input('x = ')
        y = input('y = ')
        try:
            print('x/y =', deli(x, y))
        except ValueError:
            print('program: x i y moraju biti celi!')
        except ZeroDivisionError:
            print('program: y ne sme biti 0!')

except KeyboardInterrupt:
    print('zdravo!')
```

```
x = 1
y = 0
serverski dnevnik: Mon Jun 19 15:09:03 2017 greška!
program: y ne sme biti 0!
```