Upravljanje greškama (error handling)

Igor Buzov, dipl. inf., viši predavač

Centar umjetne inteligencije Lipik

Ponavljanje

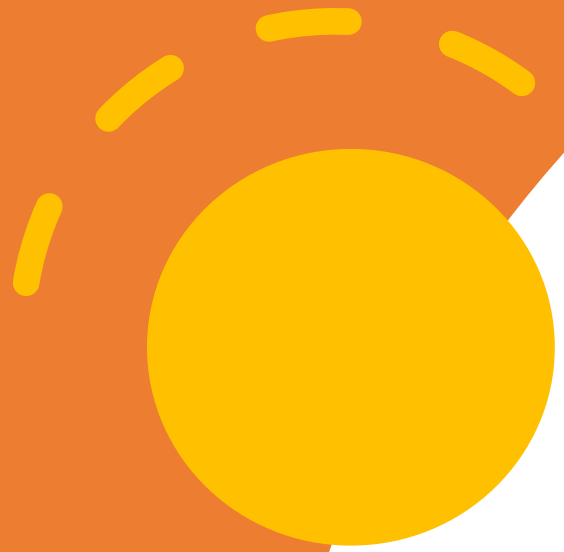
- Što je rječnik (dictionary) i kako se razlikuje od liste?
- Napravite samostalno primjer jednog rječnika
- Što je u vašem rječniku key, a što value?
- Po čemu se pretražuju podaci u rječniku?

Ponavljanje – popravite kod

```
namjestaj = {
    "naziv": "stol"
    "boja": "crna"
}
print (namjestaj)
namjestaj[boja] = "bijela"
print (namjestaj)
```

Ponavljanje – što predstavljaju varijable "ime" i "jezik"?

```
favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'rust',
    'phil': 'python',
}
ime= "jen"
if ime in favorite_languages:
    jezik = favorite_languages[ime]
    print ("najdraži jezik te osobe je: ", jezik)
```



Otvorite novi file pod nazivom error_handling.py

Ponavljanje – program za izračun cijena s maržom

- Napravite listu u kojoj se nalazi nekoliko cijena i nazovite je nabavne_cijene
- Zatražite od korisnika da unese postotak marže (npr. 0.20 za 20%)
- Napravite vlastitu funkciju koju kad pozovete će napraviti novu listu s ažuriranim cijenama, pod nazivom cijene_s_marzom
- Ispišite na kraju listu s nabavnim cijenama te ispišite listu s cijenama s maržom

Što će se desiti u vašem programu, kad pri unosu marže, korisnik umjeto broja unese neki tekst (tipa "pero")?

Vrste pogrešaka u Pythonu

- Greške u sintaksi (syntax errors) - pogreške u gramatici i strukturi Python koda
- Python ih pronalazi prije nego što program uopće počne s izvođenjem (parsing phase**)**. Program se neće ni pokrenuti
- Primjeri:
  - Zaboravljena zagrada – print ("Hello world"
  - Neispravno napisana zagrada – wlihe True:
  - Neispravno uvlačenje koda
- Ove greške se moraju "ručno" popraviti u kodu, ne mogu se uhvatiti

Vrste pogešaka u Pythonu

- Logičke pogreške (Logical Errors) / Pogreške tijekom izvođenja (Runtime Errors / Exceptions)
- To su greške koje se događaju **tijekom izvođenja programa**. Sintaksa koda je ispravna, Python ga razumije i pokrene, ali se tijekom rada dogodi nešto neočekivano ili ilegalno.
- Događaju se kada program radi, ali naiđe na uvjet koji ga sprječava da nastavi normalno izvršavanje. U Python terminologiji, kada se to dogodi, iznimka (exception) je "podignuta" (eng. raised).
- Primjeri:
  - ZeroDivisionError: Pokušaj dijeljenja broja s nulom (10 / 0)
  - ValueError: Funkcija primi argument ispravnog tipa, ali neispravne vrijednosti (npr. int("abc")).

Runtime Errors vs Exceptions

- Pojmovi "Runtime Errors" i "Exceptions" su povezani, katkada sinonimi, ali postoje razlike
- Runtime Error (Pogreška tijekom izvođenja):
  - Ovo je općenitiji pojam koji opisuje kada se greška događa. To je bilo kakva greška koja se ne otkrije prije pokretanja programa (kao sintaksna greška), već se pojavi dok program radi
  - To je problem koji sprječava normalno izvršavanje programa u nekom trenutku tijekom njegovog rada

Runtime Errors vs Exceptions

- Exception (Iznimka):

- Ovo je Pythonov mehanizam za rukovanje tim runtime greškama. Kada se runtime error dogodi, Python "podigne" (eng. raises) iznimku.
- Iznimka je objekt koji sadrži informacije o grešci (tip greške, poruku, gdje se dogodila - traceback).
- try-except blokovi služe za "hvatanje" tih podignutih iznimki i programsko reagiranje na njih.

- Tracebackovi se čitaju odozdola prema gore, opis greške je u zadnjem retku

Syntax error vs Runtime error

print ("Hello)

```
  File "main.py", line 1
    print ("Hello)
                  ^
SyntaxError: EOL while scanning string literal
```

print ("Hello", name)

```
Traceback (most recent call last):
  File "main.py", line 1, in <module>
    print ("Hello", name)
NameError: name 'name' is not defined
```

```
fruit = ["apples", "oranges", "grapes"]
vegetables = ["carrot", "onion", "broccoli"]

grocery_list = fruit + vegetables + drinks
print(grocery_list)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [37], in <cell line: 4>()
      1 fruit = ["apples", "oranges", "grapes"]
      2 vegetables = ["carrot", "onion", "broccoli"]
----> 4 grocery_list = fruit + vegetables + drinks
      5 print(grocery_list)

NameError: name 'drinks' is not defined
```

Error handling / upravljanje iznimkama

- Kada pišemo programe, često se susrećemo sa situacijama koje nismo predvidjeli, a koje mogu dovesti do "rušenja" programa (tj. program prestane raditi s porukom o grešci). Te greške koje se događaju tijekom izvođenja programa nazivamo iznimkama (eng. exceptions).

- Primjeri iznimki:
  - Korisnik unese tekst umjesto broja.
  - Program pokušava dijeliti s nulom.
  - Pokušavate otvoriti datoteku koja ne postoji.
  - Pokušavate pristupiti elementu liste na indeksu koji ne postoji.
- Cilj error handlinga je uhvatiti te iznimke i reagirati na njih na kontroliran način, umjesto da dopustimo programu da se sruši.
- Glavne naredbe za error handling u Pythonu su: try, except, else, i finally.

Try i except naredbe

- Try naredba se koristi za provjeru postoji li pogreška u programu
- Ako nema pogreške, program se nastavlja normalno izvoditi s kodom koji se nalazi dalje unutar try naredbe
- U slučaju pojave greške (iznimke), izvršava se kod pod except naredbom
- Try naredba može imati više except izraza

Primjer 1 (ZeroDivisionError)

```
def podijeli (a, b):
    try:
        result = a / b
        print ("Juupiii! Rješenje iznosi: ", result)
    except ZeroDivisionError:
        print ("Sorry, ne mogu dijeliti s nulom")


podijeli (16, 4) # prvi primjer
Podijeli (16, 0) # drugi primjer
```

- Try i except imaju kod uvučen ispod sebe i završavaju s dvotočkom
- ZeroDivisionError je naziv ugrađene iznimke koju Python sam podiže kad dođe do diljenja s nulom
- Za iznimku koja se bavi baš tim tipom grešaka, ne smijemo napisati neki drugi tekst

Primjer 2 (ValueError)

```
try:
    broj1 = int(input("Unesite prvi broj:"))
    broj2 = int(input("Unesite drugi broj:"))
    zbroj = broj1 + broj2
    print (zbroj)
except ValueError:
    print("Molim unijeti brojčane vrijednosti za brojeve!")
```

Primjer 3 (IndexError) – dodatak: više except statementa

```
moja_lista = ["jabuka", "banana", "kruška"]
print (moja_lista)
try:
    odgovor = int(input("Unesite indeks voća iz liste: "))
    odabrani_element = moja_lista[odgovor]
    print ("Voće koje ste odabrali je: ")
    print (odabrani_element)
except IndexError:
    print ("Ne postoji indeks s tim brojem!")
except ValueError:
    print ("Upisati broj, a ne tekst!!")
```

Primjer 4 (KeyError)

```
osoba = {
    "ime": "Marko",
    "prezime": "Marković",
    "godine": 25
}
try:
    unos = input ("Unesite 'ime', 'prezime' ili 'godine' osobe: ",)
    print (osoba[unos])
except KeyError:
    print ("Nema tog ključa u podacima!")
```

Zadatak 1

Popravite vaš program s početka predavanja, gdje ste računali cijene s maržom

Ubacite odgovarajući try/except koji bi pokrivao situaciju da korisnik unese tekst umjesto broja

'None' tip podatka

- None je posebna konstanta u Pythonu koja predstavlja odsutnost vrijednosti ili nulu objekta
- To nije isto što i 0, prazan string "", ili prazna lista []; to je specifična vrijednost koja kaže "nema ničega ovdje"
- Primjer liste u kojoj jedna vrijednost je None

voce= ["jabuka", None, "šljiva"]

Primjer 5 (TypeError)

```
cijena = 25
kolicina = None

print (kolicina)
try:
    ukupno = cijena * kolicina
    print (ukupno)
except TypeError:
    print ("Nemamo sve podatke")
```

Else i finally naredbe

Za veću preglednost koda, naredbe koje bi se uredno izvršavale, se ne stavljaju sve u try: blok – koristi se else blok

Za kraj programa, može se koristiti i finally: blok koji nam služi za naredbe koje će izvršiti završno zatvranaje, čišćenje programa ili krajnju poruku

Primjer 6 – else i finally

```
cijena = 25
kolicina = None

print (kolicina)
try:
    ukupno = cijena * kolicina
except TypeError:
    print ("Nemamo sve podatke")
else:
    print ("Ukupno iznosi: ", ukupno)
    print ("Ukupno s pdvom: ", ukupno * 1.25)
finally:
    print ("kraj programa")
```

Zadatak 2 – Izrada programa za izračun BMI indeksa

- Zatražite od korisnika da unese svoju težinu (u kg) i visinu (u metrima) – pohranite u zasebne varijable
- Formula za bmi glasi: težina / visina^2
- Naravno, želite se osigurati da korisnik zabunom ne unese tekst umjesto brojki. Također, ne želite ni da visina bude 0!
- U bloku koji nje try:, ispišite rezultat BMI indeksa
- Završite program s finally blokom, sa završnom pozdravnom porukom
- Program se vrti dok korisnik ne odustane i upiše kraj