Načini debagovanja u programskom jeziku Python

Dimitrije Sekulić, Sandra Radojević, Maja Gavrilović, Matija Pejić

Matematički fakultet, Beograd

April 27, 2020

Sadržaj

Uvod

- Greške pri programiranju se svima dešavaju
- Debagovanje je proces nalaženje i otklanjanje grešaka u programu.
- Ono podrazumeva sledeće:
  1. Znamo kako program treba da radi
  2. Opažamo da je do baga došlo
  3. Pronalazimo bag
  4. Uklanjamo bag

Izuzeci u Pyhon-u

```
def student(name):
    students = {
        'Pera': '107/2016',
        'Mika': '16/2016'
        'Laza': '252/2015'
    }

    print('Index of student Pera is ' + studenti[name])

student('Pera')
```

```
  File "primer.py", line 5
    'Laza': '252/2015'
          ^
SyntaxError: invalid syntax
```

Izuzeci u Pyhon-u

Kada u programu postoji sintaksna greška prevodilac izbacuje izuzetak i ispisuje **poruku o grešci**. Ona sadrži:

1. Tip greške
2. Opis greške
3. Traceback

Neki izuzeci se ne mogu izbeći, takve izuzetke hvatamo korišćenjem **try** i **except** blok.
Naš program se prevede ali ne dobijamo željeni rezultat takvu grešsku nazivamo **Semantička greška**.

Debagovanje naučnom metodom

Predstavlja formalan pristup pronalaženju problema koji je zasnovan na sledećim koracima:

1. Posmatraj
2. Napravi hipotezu
3. Predvidi
4. Testiraj
5. Zaključi

Da bi efikasno primenili ovaj način debagovanja, potrebno je da dobro vladamo tehnikama reprodukcije grešaka, automatizacijom i izolacijom grešaka, kao i da metodu ne primenjujemo za "lake" greške.

Debagovanje print metodom

Print je jednostavna, ali moćna metoda za debagovanje. Ako je koristimo adekvatno, one postaje jako sistematična i korisna. Za lepši ispis složenih tipova podataka možemo koristiti biblioteku pprint.
Korisna je i biblioteka logging, gde su nam, izmedju ostalih, na raspolaganju klase:

1. Logger
2. LogRecord
3. Handler
4. Filter
5. Formatter

PDB debager

- Pdb predstavlja interaktivni program za otklanjanje grešaka.
- Prati izvršavanje programa korak po korak, pruža pomoć pri rešavanju bagova.
- Debagovanje programa pokrećemo:
  1. iz komandne linije
  2. iz samog programa

1. Pokrećemo skript koristeći Pdb komandom python -m pdb imeprograma.py arg1 arg2
2. Umećemo deo koda u program na mesto odakle želimo da započnemo proces debagovanja.
   import pdb; pdb.set_trace()
   pdb.set_trace() postavlja debager za pozivajući stek okvir.

Primer

Primer 1.py

```
my_list = [1,9,13,3,12]
new_list = list(map(lambda x: x*2,my_list))

def sub(a,b):
  print(a)
  return a-b

diff = sub(40,2)
my_list_sum = sum(my_list)
experiment = sum(new_list) / sub(diff,my_list_sum)
```

```
> prvi.py(1)<module>()
-> my_list = [1,9,13,3,12]
(Pdb) n
> prvi.py(2)<module>()
-> new_list = list(map(lambda x: x*2,my_list))
(Pdb) n
> prvi.py(3)<module>()
-> def sub(a,b):
(Pdb) n
> prvi.py(6)<module>()
-> diff = sub(40,2)
(Pdb) s
--Call--
> prvi.py(3)<module>()
-> def sub(a,b):
(Pdb) n
> prvi.py(4)sub()
->print(a)
(Pdb) n
40
>prvi.py(5)sub()
->return a-b
(Pdb) n
--Return--
>prvi.py(5)sub->38
->return a-b
```

Tamo gde želimo da istražujemo postavljamo tačke prekida.

```
import pdb; pdb.set_trace()
experiment = sum(new_list) / sub(diff,my_list_sum)
```

Ako sada program pokrenemo sa python 1.py, prva linija za izvršavanje korišćenjem debagera biće experiment = sum(new_list) / sub(diff,my_list_sum). U slučaju da dodamo i ove argumente –m pdb izvršavanje kreće od prve linije.

```
>prvi.py(1)<module>()
->my_list = [1,9,13,3,12]
(Pdb)c
40
>prvi.py(9)<module>()
->experiment = sum(new_list) / sub(diff,my_list_sum)
(Pdb) n
38
ZeroDivisionError: division by zero
```

Tačke prekida možemo postavljati i komandom b (break).

```
(Pdb) b 9
```

Šta je PyCharm

PyCharm je integrisano razvojno okruženje koje se koristi za programiranje u jeziku Python. Pruža analizu koda, grafički debager, integraciju sa verzijom kontrolnog sistema(git) i druge pogodnosti.
Bitni pojmovi Pycharm Debugger-a:

1. Detaljno Debagovanje
2. Posmatranja
3. Inline Debugger
4. Evaluacija izraza

Tačke prekida i pokretanje Debugger-a

U okruženju PyCharm tačke prekida postavljamo klikom na levu marginu (oznaka tačke prekida je crveni kružić).
Prilikom kompilacija možemo odabrati opciju Debug, nakon čega dobijamo zaseban prozor za Debagovanje (Debug Tool Window)

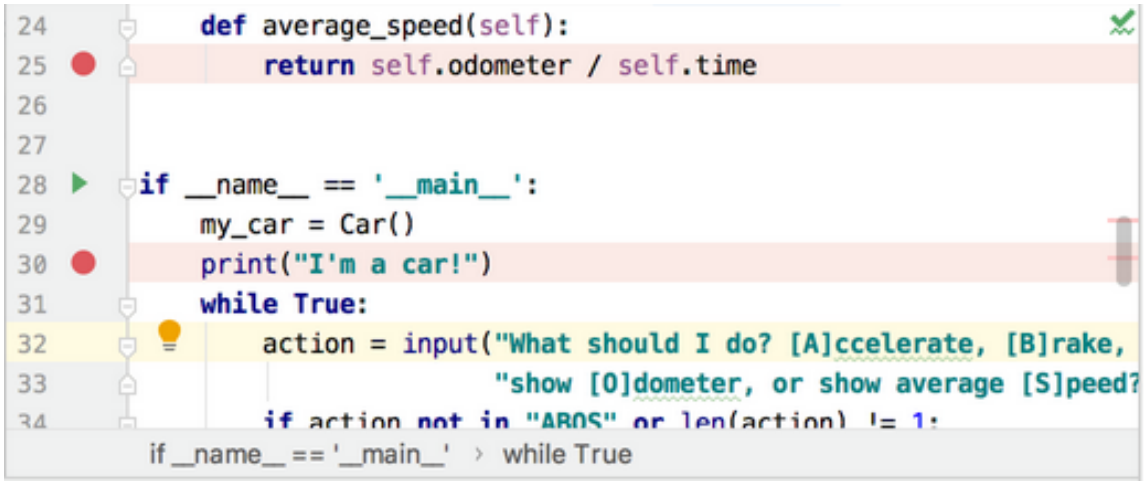

Figure: Postavljanje Tačaka prekida.

Opcije Debugger-a

Sve opcije Debugger-a se nalaze u Debug Tool Window.
Inline Debbuger je opcija koja nam pruža da u vidu komentara u editoru vidimo sve vrednosti promenljivih.
Evaluacija izraza je opcija koja nam omogućava da izračunamo bilo koji izraz sa trenutnim vrednostima promenljivih u kodu, kao i da dodeljujemo vrednosti promenljivim.
Posmatranja su zaseban prozor u kome se nalaze sve promenljive koje su trenutno definisane kao i njihove vrednosti, u Posmatranja mozemo ručno dodati bilo koju promenljivu (čak i one koje su trenutno ne definisane), njihova vrednost će biti **Null**.
Detaljno Debagovanje prestavlja skup opcija za iteriranje kroz kod korak po korak.

Paragraphs of Text

Sed iaculis dapibus gravida. Morbi sed tortor erat, nec interdum arcu. Sed id lorem lectus. Quisque viverra augue id sem ornare non aliquam nibh tristique. Aenean in ligula nisl. Nulla sed tellus ipsum. Donec vestibulum ligula non lorem vulputate fermentum accumsan neque mollis.

Sed diam enim, sagittis nec condimentum sit amet, ullamcorper sit amet libero. Aliquam vel dui orci, a porta odio. Nullam id suscipit ipsum. Aenean lobortis commodo sem, ut commodo leo gravida vitae. Pellentesque vehicula ante iaculis arcu pretium rutrum eget sit amet purus. Integer ornare nulla quis neque ultrices lobortis. Vestibulum ultrices tincidunt libero, quis commodo erat ullamcorper id.

Bullet Points

- Lorem ipsum dolor sit amet, consectetur adipiscing elit
- Aliquam blandit faucibus nisi, sit amet dapibus enim tempus eu
- Nulla commodo, erat quis gravida posuere, elit lacus lobortis est, quis porttitor odio mauris at libero
- Nam cursus est eget velit posuere pellentesque
- Vestibulum faucibus velit a augue condimentum quis convallis nulla gravida

Blocks of Highlighted Text

Block 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Block 2

Pellentesque sed tellus purus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vestibulum quis magna at risus dictum tempor eu vitae velit.

Block 3

Suspendisse tincidunt sagittis gravida. Curabitur condimentum, enim sed venenatis rutrum, ipsum neque consectetur orci, sed blandit justo nisi ac lacus.

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Table

Treatments	**Response 1**	**Response 2**
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table: Table caption

Theorem

Theorem (Mass–energy equivalence)

$E = mc^2$

Primer

Majin primer

```
my_list = [1,9,13,3,12]
new_list = list(map(lambda x: x*2,my_list))

def sub(a,b):
  print(a)
  return a-b

diff = sub(40,2)
my_list_sum = sum(my_list)
experiment = sum(new_list) / sub(diff,my_list_sum)
```

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

The End