Debugging

Lecture 4

Centre for Data Science, ITER
Siksha ‘O’ Anusandhan (Deemed to be University),
Bhubaneswar, Odisha, India.

Contents

Introduction

- When a program fails to yield the desirable result, we say that it contains a bug.
- The bug could be an error such as division by zero, invalid type conversion, using a variable not defined, wrong initialization, or some other unintended operation being performed.
- The process of discovering the bugs and making the program bug-free is called debugging.
- debugging: making the program error free.

Testing

- Program testing aims to expose the presence of bugs in the programs.
- To find the bugs in a program, we test it on various inputs.
- We correct the errors found in this process and feel confident that the program will run smoothly.

Testing

- For Example (Finding Maximum of Three Numbers):

```
def max3(n1, n2, n3):
    '''
    Objective: To find maximum of three numbers
    Input Parameters: n1,n2,n3 - numeric values
    Return Value: maxNumber - numeric value
    '''
    maxNumber=0
    if n1> n2 :
        if n1 > n3:
            maxNumber = n1
    elif n2 > n3:
        maxNumber=n2
    else:
        maxNumber = n3
    return maxNumber
def main():
    '''
    Objective: To find maximum of three numbers
    Input Parameter: None
    Return Value: None
    '''
    n1 = int(input('Enter first number: '))
    n2 = int(input('Enter second number: '))
    n1 = int(input('Enter third number: '))
    maximum = max3(n1,n2,n3)
    print('Maximum number is:',maximum)

if__name__='__main__':
    main()
```

Testing

- The function max3 is intended to return the maximum of three numbers.Since the input numbers can be either positive or negative,we may have test cases containing various combina tions of positive and negative numbers.
- We test this script on various permutations of the inputs: 10, 20, 30, namely:

1. 30, 20, 10	4. 20, 30, 10
2. 30, 10, 20	5. 10, 30, 20
3. 20, 10, 30	6. 10, 20, 30

- It so turns out that the result obtained is correct for all input permutations, except for the permutations 20, 10, and 30:

```
⟩⟩⟩
Enter first number: 20
Enter second number: 10
Enter third number: 30
Maximum number is 0
```

- Thus, a bug persists in the program that needs to be detected and removed.

Debugging

- There are various Python debugging tools such as pdb, pudb, pydb, and pydbgr. In this section, we will discuss Python's built-in debugger pdb, which is an interactive tool that helps in examining the code execution step by step.
- Debugging allows us to stop the execution of a program at a chosen instruction called a break point, and evaluate the various expressions in the current context.
- In order to debug a program in Python shell, we need to import the module pdb as well as the script to be debugged.

```
⟩⟩⟩ import pdb
⟩⟩⟩ import max3
⟩⟩⟩ pdb.run('max3.main()')
> <string >(1) <module>()->None
```

- Python debugger uses the prompt (Pdb) to indicate that the program is executing in the debugging mode.

Debugging

- Another way to execute a program in debugging mode is by including the following two lines in the script:

```
import pdb
pdb.set_trace()
```

- Note that including above two lines at the beginning of the program for Finding maximum of three numbers. It will increase every line number by two. By invoking the function set_trace() at the very beginning, the program would run in debugging mode right from the first instruction to be executed.

- However, since we know that the bug exists in the function max3, we may invoke set_trace() within the function body.

Debugging

- Let us have a look at debugging commands:

Command	**Explanation**
h **or** help	it lists all the availble commands.
h commandName	it prints the description about a command.
w **or** where	Prints the stack trace (sequence of function calls currently in execution, most recent function call being at the beginning). Also shows the statement to be executed next.
u **or** up	Moves to the stack frame one level up in the stack trace
d **or** down	Moves to the stack frame one level down in the stack trace
s **or** step	Executes the current statement and moves the control to either next statement, or the function being invoked in the current statement.
n **or** next	Executes the current statement and moves the control to the next statement.unlike step command, if a function is being invoked in the current statement, the invoked function gets executed completely.
r **or** return	Continue execution until the current function returns.
p expression print(expression)	Prints the value of the specified expression in the current context.

Debugging

Command	Explanation
j (jump) lineno	Jumps to the given line number for the next statement to be executed.
l **or** list	List 11 lines in the vicinity of the current statement
b **or** break [[file :]] line[func[,cond]]	Sets the breakpoint at the specified line.Name of the file is one of the optional arguments. When the argument **func** is used, the breakpoint is set at the first executable statement of the specified function. The second argument may be used to denote a condition which must evaluate to **True** for setting the breakpoint.
tbreak [[file :]] line[func[,cond]]	Similar to **break** command. However, the break point being set is automatically removed once it is reached
cl **or** clear [[file :]] line[func[,cond]]	Clears the specified breakpoint. In the absence of an argument,clears all the breakpoints.
c **or** continue	Continue execution untill the breakpoint is reached
a **or** args	Prints the argument list of the current function along with their values.
q **or** quit	Quits from the python debugger

Debugging

- Execution of Finding Maximum of Three Numbers Program in debugging for the inputs 20,10,30 is shown below:

```
 > f:\pythoncode \ch04 \max3.py<module>()
 -> def max3(n1,n2,n3):
 (pdb) h s
 s(step)
      Execute the current line, stop at the first possible
      ocassion (either in a function that is called or in the current function)
 (pdb)s
 > f:\pythoncode \ch04 \max3.py<module>()
 -> def main():
 (pdb)s
 > f:\pythoncode \ch04 \max3.py<module>()
->if__name__='__main__':
 (pdb)s
> f:\pythoncode \ch04 \max3.py <module>()
->main()
 (pdb)s
–call–      > f:\pythoncode \ch04 \max3.py main()
->def main()
```

Debugging

```
(pdb)s
> f:\pythoncode \ch04 \max3.py main()
n1= int(input('Enter first number: '))
(pdb) n
Enter first number: 20
> f:\pythoncode \ch04 \max3.py main()
n1= int(input('Enter second number: '))
(pdb) n
Enter second number: 10
> f:\pythoncode \ch04 \max3.py main()
n1= int(input('Enter third number: '))
(pdb) n)
Enter third number: 30
> f:\pythoncode \ch04 \max3.py main()
->maximum = max3(n1,n2,n3)
(pdb) p (n1, n2, n3)
(20,10,30)
```

Debugging

```
 (pdb) n
> f:\pythoncode \ch04 \max3.py main()
 -> print('Maximum number is',maximum)
 (pdb) n
Maximum number is 0
 –Return–
> f:\pythoncode \ch04 \max3.py main() -< None
 -> print('Maximum number is',maximum)
 (pdb) s
 –Return–
> f:\pythoncode \ch04 \max3.py <module>()-> None
 -> main()
 (pdb) q
 Traceback (most recent call last):
 File "F:/pythoncode/ch04/max3.py",27 line 32, in <module >
 main()
bdb.BdbQuit
```

References

[1] Python Programming: A modular approach by Taneja Sheetal, and Kumar Naveen, *Pearson Education India, Inc.*, 2017.