LAB 1 SUPPLEMENTARY:

BASIC DEBUGGING WITH PYTHON

General Tips on Minimizing Errors

Debugging with ‘print’

Debugging with PDB

Using Google/Stack Overflow

General Tips on Minimizing Errors

Do not panic when you get errors

Outline your code structure ahead of time

Keep your code organized

Test your code often

More tips on avoiding errors by Berkeley online textbook
https://pythonnumericalmethods.berkeley.edu/notebooks/chapter10.00-Errors-Practices-Debugging.html

Basic Debugging with ‘print’

Code block 1

print(“done running block 1”) → Runs when code block 1 doesn’t produce an error

Code block 2

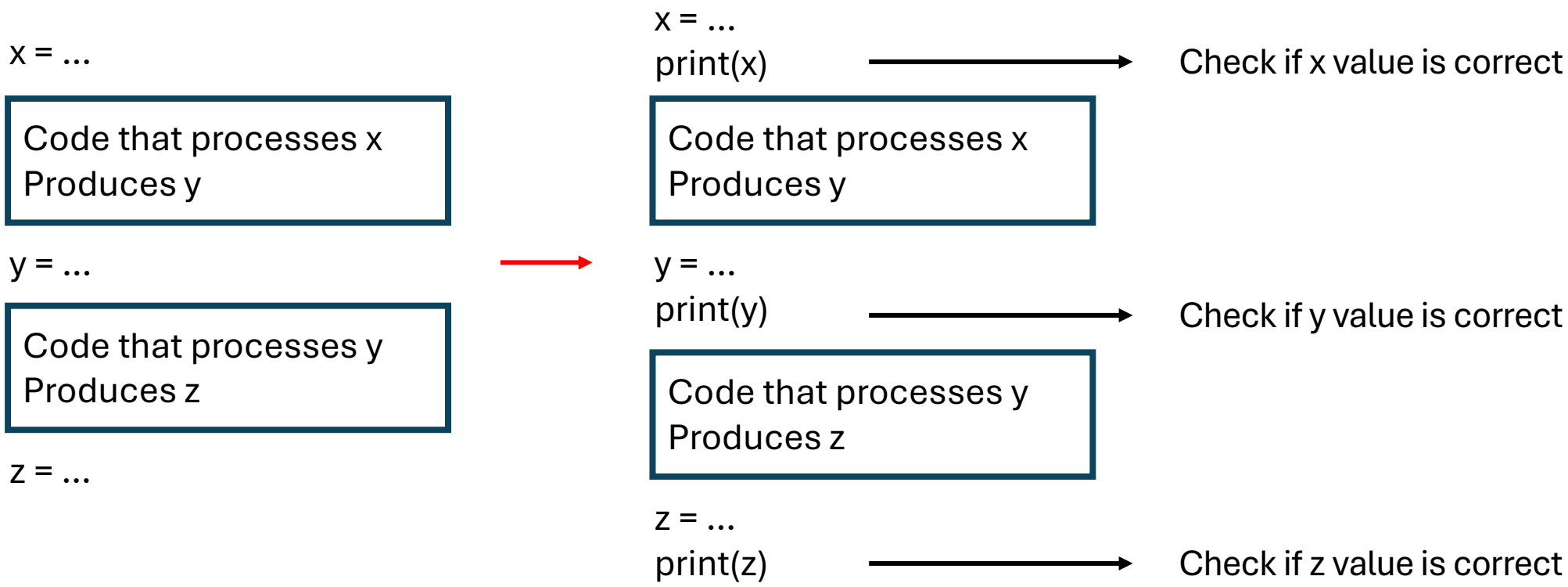
print(“done running block 2”) → Runs when code block 2 doesn’t produce an error

Code block 3

print(“done running block 3”)

.
.
.

Basic Debugging with ‘print’



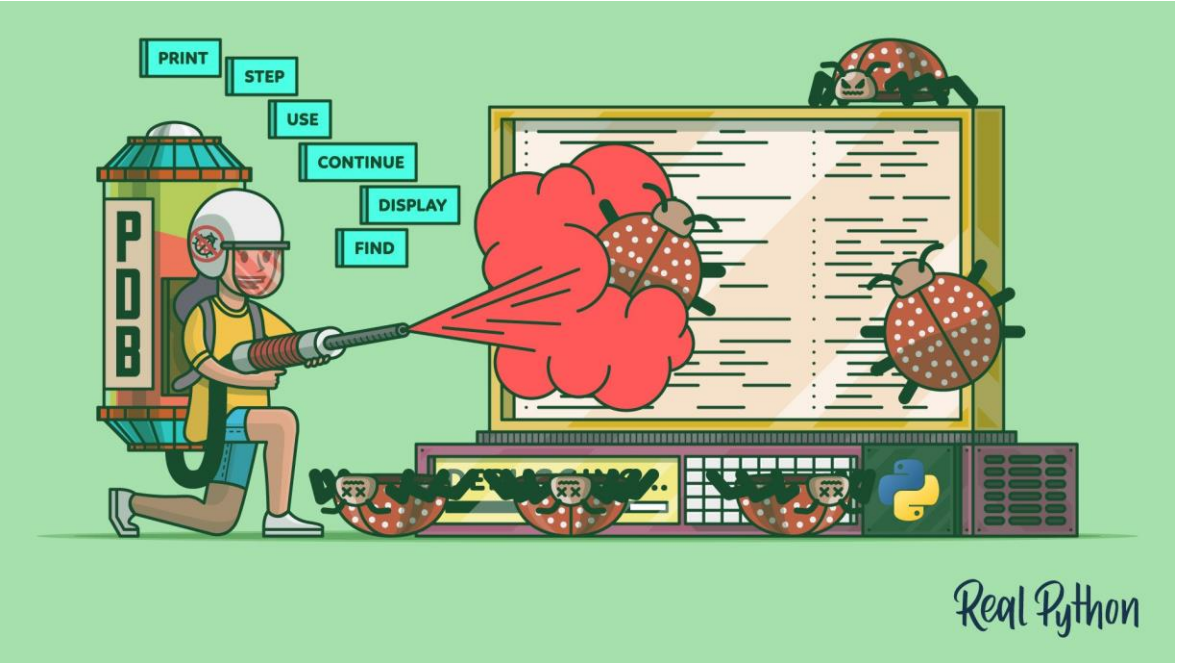
Original code

Debugging each step with print()

Python Debugger

What is Python Debugger (PDB)?

- Debugging tool built in Python
- No installation required
- Provides Interactive environment
- Widely used for systemic debugging



PDB Basics

```
def divide(divisor):

    breakpoint()

    val = 1/divisor

    return val
```

Upon execution of the code, activates Python debugger console at this line

```
divide(0)
```

```
> <ipython-input-24-03e5b3dd8265>(5)divide()
      3     breakpoint()
      4
----> 5     val = 1/divisor
      6
      7     return val
```

```
ipdb>
```

Python debugger console

PDB Useful Commands

```
> <ipython-input-24-03e5b3dd8265>(5)divide()
      3     breakpoint()
      4
----> 5     val = 1/divisor
      6
      7     return val

ipdb>
```

- h: help
- w: where
- n: next
- c: continue
- p: print
- l: list
- q: quit

PDB Useful Commands

```
divide(0)
```

```
> <ipython-input-24-03e5b3dd8265>(5)divide()
      3     breakpoint()
      4
----> 5     val = 1/divisor
      6
      7     return val

ipdb> h

Documented commands (type help <topic>):
========================================
EOF     commands   enable    ll        pp       s                until
a       condition  exit      longlist  psource  skip_hidden      up
alias   cont       h         n         q        skip_predicates  w
args    context    help      next      quit     source           whatis
b       continue   ignore    p         r        step             where
break   d          interact  pdef      restart  tbreak
bt      debug      j         pdoc      return   u
c       disable    jump      pfile     retval   unalias
cl      display    l         pinfo     run      undisplay
clear   down       list      pinfo2    rv       unt

Miscellaneous help topics:
==========================
exec  pdb

ipdb> h l
Print lines of code from the current stack frame

ipdb>
```

h: help

w: where

n: next

c: continue

p: print

l: list

q: quit

Provide documentation for the command

PDB Useful Commands

```
divide(2)

> <ipython-input-1-f4a9c3842530>(7)divide()
      3     val = 1/divisor
      4
      5     breakpoint()
      6
----> 7     return val

ipdb> w
    [... skipping 27 hidden frame(s)]
  <ipython-input-2-cfdb0f794c84>(1)<module>()
----> 1 divide(2)

> <ipython-input-1-f4a9c3842530>(7)divide()
      3     val = 1/divisor
      4
      5     breakpoint()
      6
----> 7     return val

ipdb>
```

h: help

w: where

n: next

c: continue

p: print

l: list

q: quit

Print a stack trace (i.e. the order of code being executed), with the most recent frame at the bottom.

PDB Useful Commands

```
divide(2)

> <ipython-input-1-03e5b3dd8265>(5)divide()
      3     breakpoint()
      4
----> 5     val = 1/divisor
      6
      7     return val

ipdb> n
> <ipython-input-1-03e5b3dd8265>(7)divide()
      3     breakpoint()
      4
      5     val = 1/divisor
      6
----> 7     return val

ipdb>
```

h: help
w: where
n: next
c: continue
p: print
l: list
q: quit

Executes the next line of code

PDB Useful Commands

```
divide(2)

> <ipython-input-1-a669feb31165>(5)divide()
      3     breakpoint()
      4
----> 5     val = 1/divisor
      6
      7     breakpoint()

ipdb> c
> <ipython-input-1-a669feb31165>(9)divide()
      5     val = 1/divisor
      6
      7     breakpoint()
      8
----> 9     return val

ipdb>
```

h: help
w: where
n: next
c: continue
p: print
l: list
q: quit

Run the code until the next breakpoint()

PDB Useful Commands

```
divide(2)
```

```
> <ipython-input-1-f4a9c3842530>(7)divide()
      3     val = 1/divisor
      4
      5     breakpoint()
      6
----> 7     return val

ipdb> p val
0.5

ipdb>
```

h: help
w: where
n: next
c: continue
p: print
l: list
q: quit

Print the value of the desired variable

PDB Useful Commands

```
divide(2)
```

```
> <ipython-input-1-f4a9c3842530>(7)divide()
      3     val = 1/divisor
      4
      5     breakpoint()
      6
----> 7     return val

ipdb> l
      2
      3     val = 1/divisor
      4
      5     breakpoint()
      6
----> 7     return val

ipdb>
```

h: help
w: where
n: next
c: continue
p: print
l: list
q: quit

Similar to where but print the lines of code from the current stack frame

PDB Useful Commands

```
divide(2)
```

```
> <ipython-input-1-f4a9c3842530>(7)divide()
      3     val = 1/divisor
      4
      5     breakpoint()
      6
----> 7     return val

ipdb> q

---------------------------------------------------------------------------
BdbQuit                                   Traceback (most recent call last)
<ipython-input-2-cfdb0f794c84> in <module>
----> 1 divide(2)

<ipython-input-1-f4a9c3842530> in divide(divisor)
      5     breakpoint()
      6
----> 7     return val

<ipython-input-1-f4a9c3842530> in divide(divisor)
      5     breakpoint()
      6
----> 7     return val

~\anaconda3\lib\bdb.py in trace_dispatch(self, frame, event, arg)
     86             return # None
     87         if event == 'line':
---> 88             return self.dispatch_line(frame)
     89         if event == 'call':
     90             return self.dispatch_call(frame, arg)

~\anaconda3\lib\bdb.py in dispatch_line(self, frame)
    111         if self.stop_here(frame) or self.break_here(frame):
    112             self.user_line(frame)
--> 113             if self.quitting: raise BdbQuit
    114         return self.trace_dispatch
    115

BdbQuit:
```

h: help
w: where
n: next
c: continue
p: print
l: list
q: quit

Exit the debugger console

Useful video tutorial of basic usage of Python debugger:
https://www.youtube.com/watch?v=aZJnGOwzHtU

Using Google/Stack Overflow