Chapter 13

# Debugging Strategies

Learning by debugging

## Two types of errors :

1. program does not run at all → make it running

2. program runs but gives an incorrect output → check procedures

    Avoidance by good design
    Result validation

## General Strategies

<u>Build upon working elements</u>

1. Think about the general approach to your problem
2. Start building towards incremental success
3. Get each element of your program working before moving on

e.g.
- print intermediate steps on the screen while writing the script
- use a sandbox folder
- use artificial or copied data

## General Strategies

<u>Think about your assumptions</u>

1. Make sure you are editing the version of the program that you are actually using → editing wrong script

   Use which command to get the absolute path of the program

2. Save changes before re-execution

3. Check line endings

   Incorrect line endings in input files → program can combine data lines

```
InFile = open(InFileName, 'rU')
```

   converts all line endings to newline (\n) characters

Chapter 13

## General Strategies

Think about your assumptions

1. Make sure you are editing the version of the program that you are actually using → editing wrong script
   Use which command to get the absolute path of the program

2. Save changes before re-execution

3. Check line endings

4. Check contents of your data file
   Incorrect input files can crash programs
      example : AGTC ..., sequence file that contains - or ?
         Postive or negative numbers,
         . or ,

## **Specific debugging techniques**

<u>Isolate the problem</u>

- Error report : reported line often does not contain error. Check previous steps.

- Comment in/out sections with # or ''' .... ''' (or """ … """ ?) triple quotes for multiple lines.

# … Comment on what you write

## Specific debugging techniques

Write verbose software

- incorporate diagnostic print statements
- if Debug statement, try with authors2.py

```
import sys
Debug = True

# (insert program statements here)

# wherever you want to give feedback, insert these lines
if Debug :
    Print MyList
    # or you can use
    sys.stderr.write(MyList)
```

## **Error messages and their meanings**

### Common Python errors

```
-bash : myscript.py : command not found
```

Program not found in the folder listed in your PATH,
permission not set to executable,
→ set PATH variable, try chmod u+x

```
/Users/lucy/scripts/myprogram.py: line 3: import command not found
```

Problem with python program,

1. Perhaps problems with shebang line: #!
2. misspelled built-in Python function within the program

## Error messages and their meanings

### Common Python errors

bad interpreter not a directory

#! has a / after /usr/bin/env/

r /usr/bin/env: bad interpreter: No such file or directory

Parts in your # ! statement not found
→ copy in statement into the terminal, see if it launches Python

Permission denied

chmod u+x

Chapter 13

# Error messages and their meanings

## Common Python errors

### Name 'x' is not defined

- misspelled variable name in the program

- variable not originally defined
→ Inititialize variable e.g. , MyList=[], or MyString=" "

- function used, but not imported from a module first

- function used without the required module name in dot notation
  e.g. Randint(5) instead of random.randint(5)

Chapter 13

## Error messages and their meanings

### Common Python errors

Indentation error

- 4 commas vs 1 tab

Attribute error

Misspelling of a built-in function
e.g. MyString.lowercase() instead of MyString.lower()

Type error 'xx' object is not callable

Want to get values from a List( ) and not List[ ],
wrong interpretation as a function and not a list

Chapter 13

# Error messages and their meanings

## Common Python errors

### Traceback … zero division error

Division by zero !
- Function returns unexpected 0
- Input data with 0

→ check user and variable input,  to be not blank, non zero, …

### Non-ASCII character '\xe2' in file

Or ävoid non ASCI characters, or type # coding : utf-8 below the #! line

### Invalid syntax

Many things possible : Missing colon after if, else, or for statement
Missing close parenthesis, brackets,
= instead of ==

## Error messages and their meanings

### Shell errors

Illegal byte sequence

- Some command line programs cannot process Unicode characters •, °, ≠, … in a file being read

Improper use of \ > * < ;

# Making your program more efficient

Optimization

- sometimes everything works but just too slow / inefficient

    - multiple ways to do the same thing

Measure time that a program needs :

```
import time
StartTime = time.time()
#perform your commands here
print "Elapsed : %.5f" % 9time.time() - StartTime)
```

Chapter 13

## Making your program more efficient

*try* and *except* to handle errors

```
for Line in File:
    if Line [0] ==">":
        Name=Line.strip()[1:]
    # lines with > are Names
    else:
        #check for a pre-existing key
        if Name in Dict.keys():
            Dict[Name] += Line.strip()
        # not a key so define
        else:
            Dict[Name] = Line.strip()
```

Traditional

```
for Line in File:
    if Line [0] ==">":
        Name=Line.strip()[1:]
    # lines with > are Names
    else :
      try:
        # try to append with +=
        # assumes Name is a key
        Dict[Name] += Line.strip()
        # oops, not a key so define
      except KeyError:
        Dict[Name] = Line.strip()
```

Fast