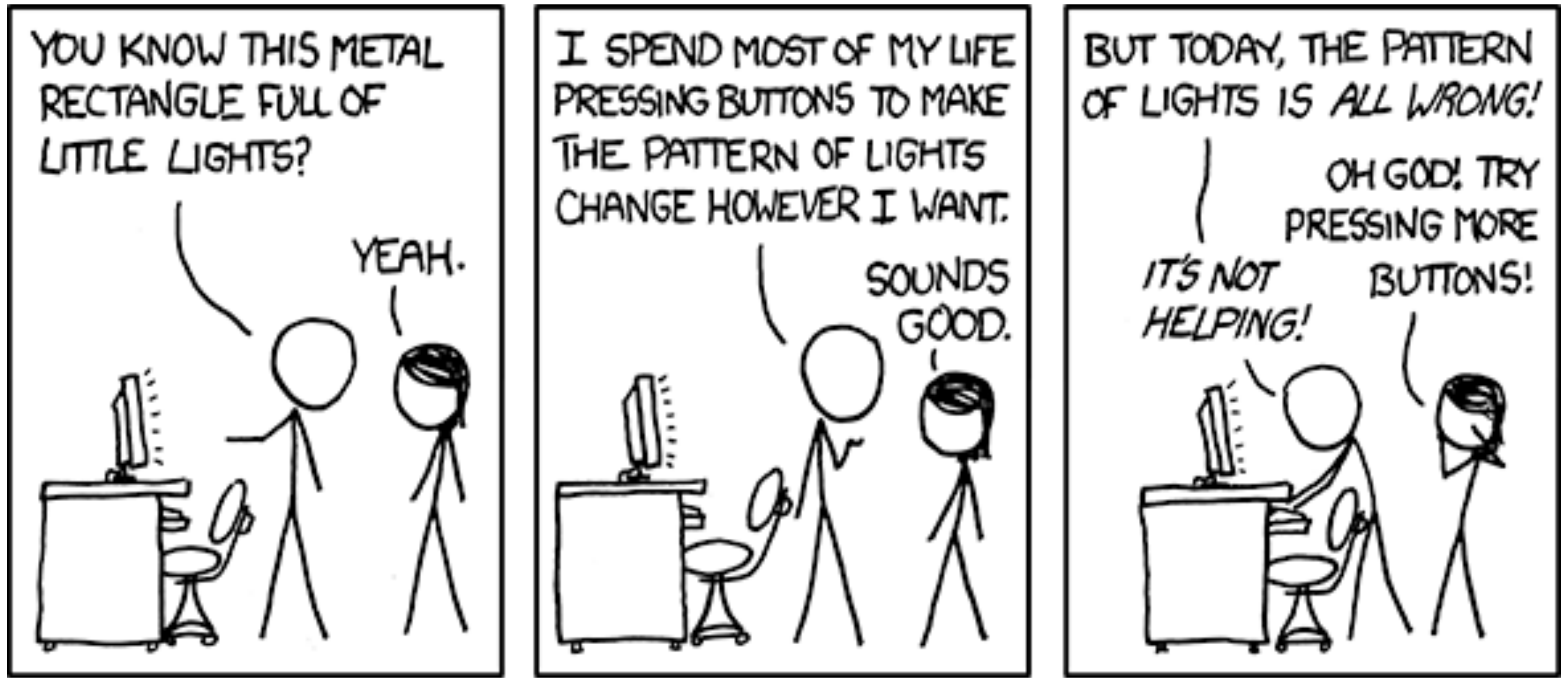
Python Scripting - Debugging


YOU KNOW THIS METAL RECTANGLE FULL OF LITTLE LIGHTS?
YEAH.
I SPEND MOST OF MY LIFE PRESSING BUTTONS TO MAKE THE PATTERN OF LIGHTS CHANGE HOWEVER I WANT.
SOUNDS GOOD.
BUT TODAY, THE PATTERN OF LIGHTS IS ALL WRONG!
OH GOD! TRY PRESSING MORE BUTTONS!
IT'S NOT HELPING!

Fall 2018
PCfB Class 7
October 12, 2018

A FATAL EXCEPTION ØE HAS OCCURRED.
AAAA!
THE CURRENT APPLICATION WILL BE TERMINATED.
AAAA!
PRESS ANY KEY TO CONTINUE.
HERE! TAKE IT ALL! JUST GO AWAY! PLEASE!
I WISH I'D THOUGHT OF DRESSING AS A BLUE SCREEN OF DEATH.
I FEEL LIKE I SHOULD SEND MICRO-SOFT SOME OF THIS.

Outline

- Types of errors
- Debugging tools/tips
- General structure of my scripts

Syntax Errors

- Detected before program is run
- Part of your code is not understood by the interpreter

```
  File "./rev_comp_v1.py", line 28
    print revseq.translate(trantab)
        ^
SyntaxError: invalid syntax
```

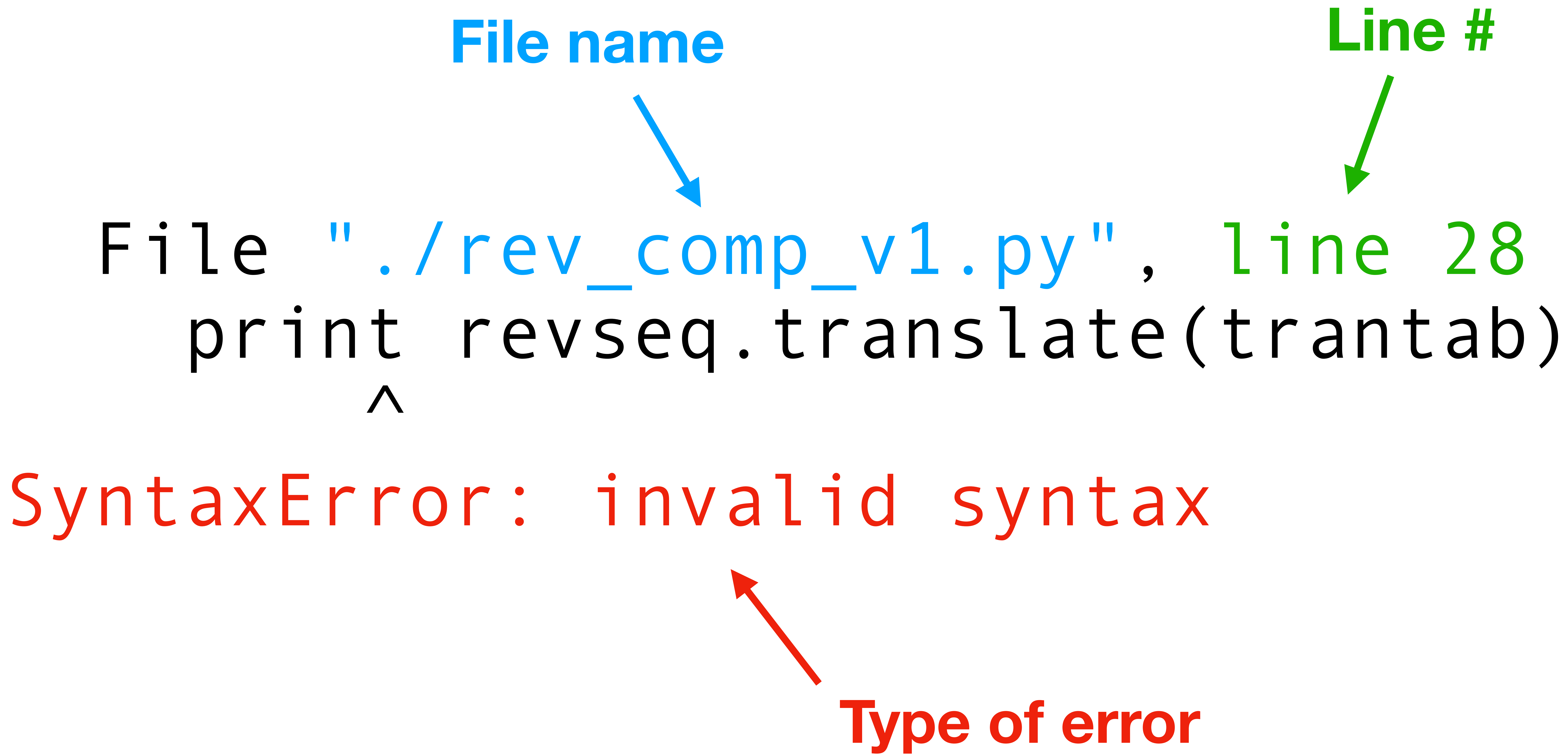
```
  File "./rev_comp_v1.py", line 28
    print revseq.translate(trantab)
              ^
SyntaxError: invalid syntax
```

Type of error

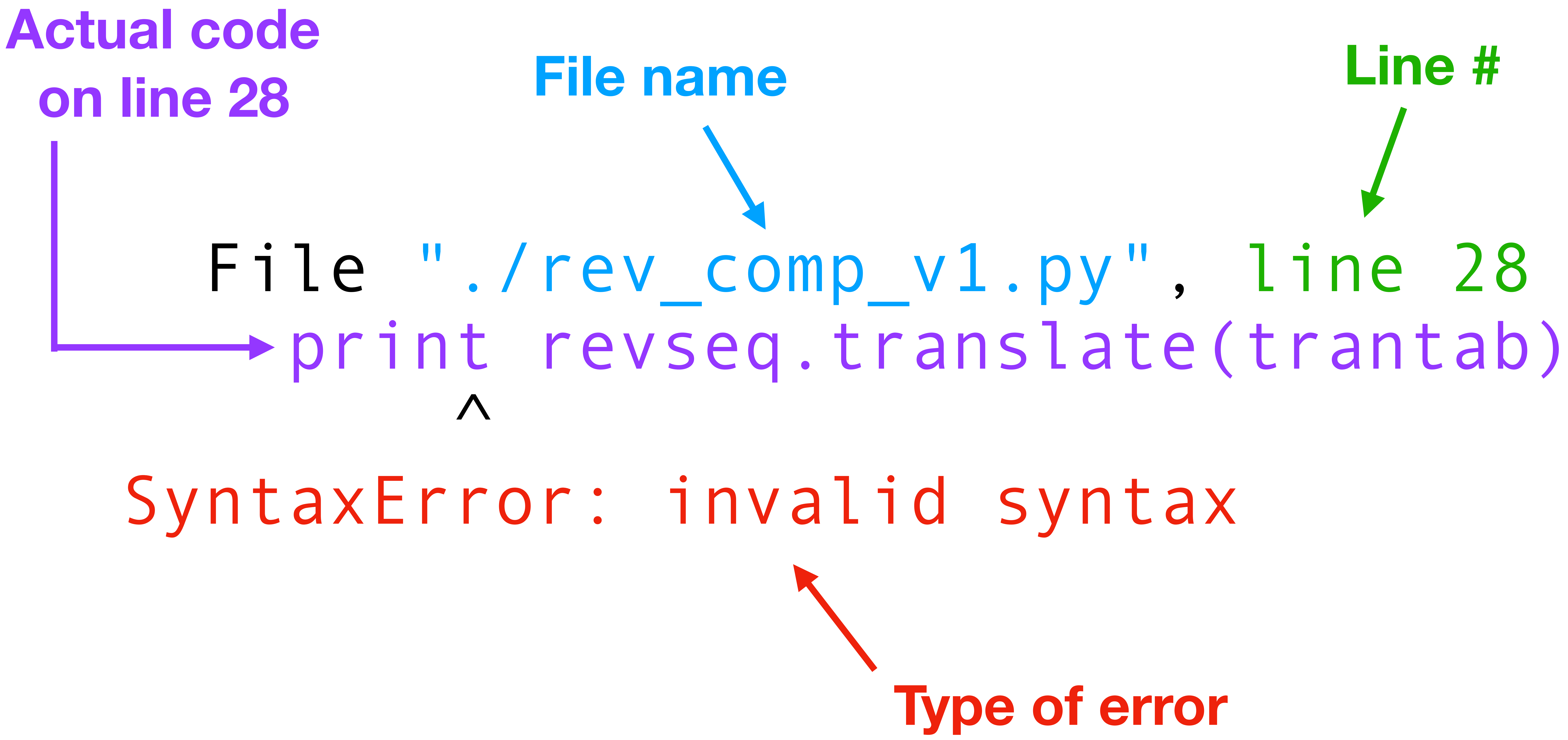
File name

```
  File "./rev_comp_v1.py", line 28
    print revseq.translate(trantab)
        ^
SyntaxError: invalid syntax
```

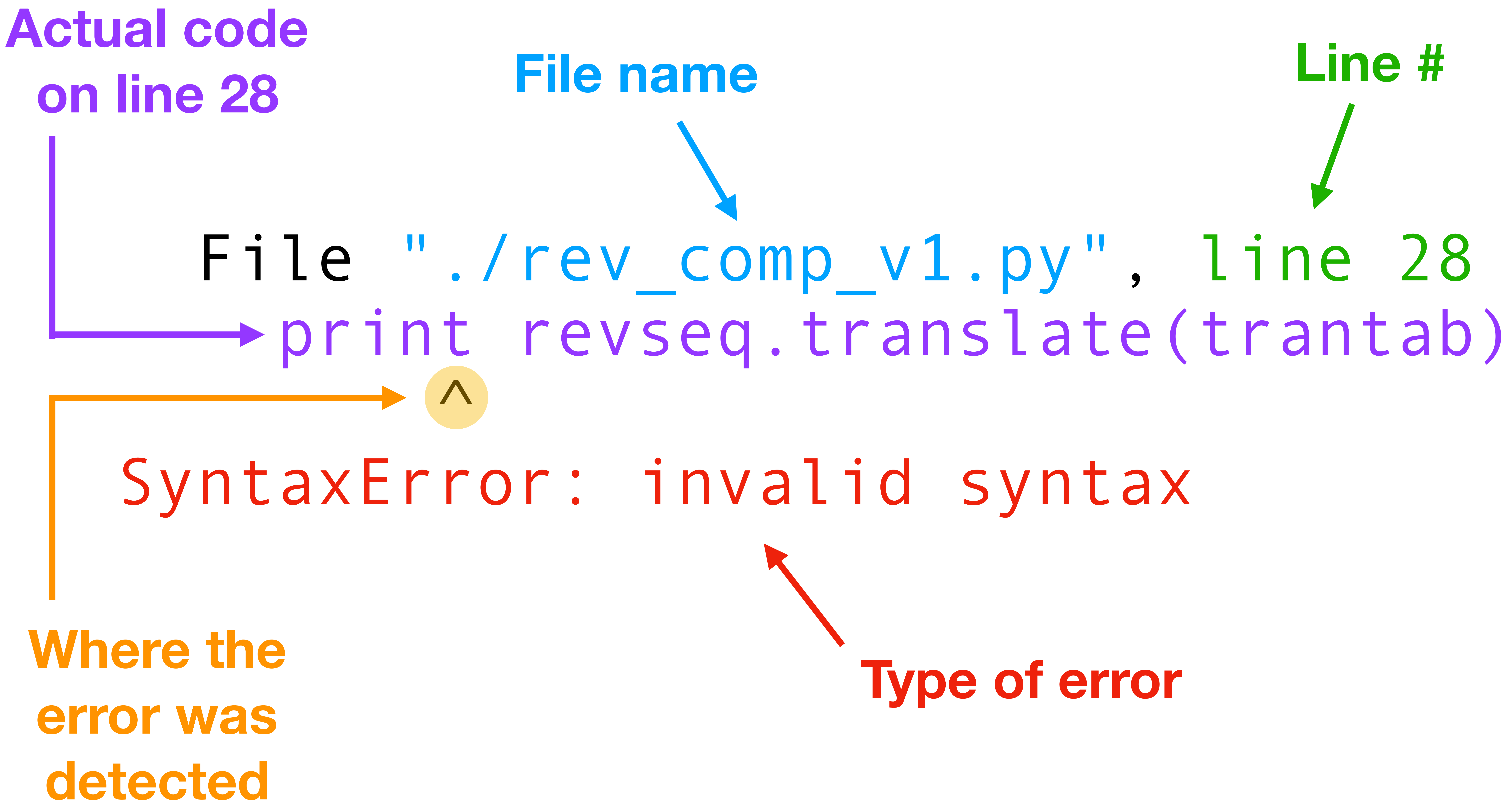
Type of error



```
File "./rev_comp_v1.py", line 28
    print revseq.translate(trantab)
        ^
SyntaxError: invalid syntax
```



```
    File "./rev_comp_v1.py", line 28
      print revseq.translate(trantab)
                ^
SyntaxError: invalid syntax
```



```
  File "./rev_comp_v1.py", line 28
    print revseq.translate(trantab)
        ^
SyntaxError: invalid syntax
```

Runtime Errors (aka, Exceptions)

- Occur when program is executed
- 29 standard exception types
  (https://www.tutorialspoint.com/python/standard_exceptions.htm)
- Format = Traceback

```
Traceback (most recent call last):

  File "./rev_comp_v3.py", line 32, in <module>
    main()

  File "./rev_comp_v3.py", line 17, in main
    print_rev_comp(seq)

  File "./rev_comp_v3.py", line 28, in print_rev_comp
    print revseq.translate(trantab)

AttributeError: 'builtin_function_or_method' object
has no attribute 'translate'
```

```
Traceback (most recent call last):

  File "./rev_comp_v3.py", line 32, in <module>
    main()

  File "./rev_comp_v3.py", line 17, in main
    print_rev_comp(seq)

  File "./rev_comp_v3.py", line 28, in print_rev_comp
    print revseq.translate(trantab)

AttributeError: 'builtin_function_or_method' object
has no attribute 'translate'
```

Type of error

File names **Line #s**

```
Traceback (most recent call last):

  File "./rev_comp_v3.py", line 32, in <module>
    main()

  File "./rev_comp_v3.py", line 17, in main
    print_rev_comp(seq)

  File "./rev_comp_v3.py", line 28, in print_rev_comp
    print revseq.translate(trantab)

AttributeError: 'builtin_function_or_method' object
has no attribute 'translate'
```

Type of error

File names **Line #s**

```
Traceback (most recent call last):

  File "./rev_comp_v3.py", line 32, in <module>
    main()

  File "./rev_comp_v3.py", line 17, in main
    print_rev_comp(seq)
```

Where error actually occurred

```
  File "./rev_comp_v3.py", line 28, in print_rev_comp
    print revseq.translate(trantab)

AttributeError: 'builtin_function_or_method' object
has no attribute 'translate'
```

Type of error

File names **Line #s**

```
Traceback (most recent call last):

  File "./rev_comp_v3.py", line 32, in <module>
    main()

  File "./rev_comp_v3.py", line 17, in main
    print_rev_comp(seq)

  File "./rev_comp_v3.py", line 28, in print_rev_comp
    print revseq.translate(trantab)
AttributeError: 'builtin_function_or_method' object
has no attribute 'translate'
```

Where error actually occurred

Type of error

Tip #1: Look up!

- Problem not necessarily on the line where the error was detected
- Could be on a proceeding line

Tool #1: `print` statements

- Prior to the error to check the status of important variables
- Within loops to check whether conditions have been met

Tool #2: Comments

- Temporarily remove sections of code to isolate problem

Use # to comment out a single line

Use ‘ ’ ‘ ’ ’ ’ to comment out multiple lines

Tip #2: Interactive interpreter

- Don’t forget about the command line interface
- Easy way to test commands

Example: rev_comp.py

```
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ----------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions --------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions ----------------

if __name__ == "__main__":
    main()
```

Import modules

```
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ----------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------

if __name__ == "__main__":
    main()
```

Import modules

Function definitions

```
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ----------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------

if __name__ == "__main__":
    main()
```

Import modules

Main body of script

Function definitions

```
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ----------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------

if __name__ == "__main__":
    main()
```

```
#!/usr/bin/env python
```

Import modules

```
# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------
```

Main body of script

```
# --------- Start of main() ----------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------
```

Function definitions

```
# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------
```

Only execute main() if script is called directly

```
if __name__ == "__main__":
    main()
```

__name__

- When a script is called directly:
    - __name__ == “__main__”
- When a script is imported as module
    - __name__ == “module name”

- Functions can be imported directly to other programs

```
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ---------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------

if __name__ == "__main__":
    main()
```

Only execute main() if script is called directly

Main body of script

```
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ----------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------

if __name__ == "__main__":
    main()
```

optparse module

```python
#!/usr/bin/env python

# --------- Start Importing modules ---------
from __future__ import division
import optparse
from string import maketrans
# --------- Done Importing modules ---------

# --------- Start of main() ---------------
def main():
    usage = '%prog [options] seq1 [seq2 ...]'
    p = optparse.OptionParser()
    opts, args = p.parse_args()

    for seq in args:
        print_rev_comp(seq)

# --------- End of main() ------------------

# --------- Start of Funtions ---------------
def print_rev_comp(seq):
    revseq = seq[::-1].upper()
    intab = "ACTG"
    outtab = "TGAC"
    trantab = maketrans((intab, outtab)
    print revseq.translate(trantab)
# --------- End of Funtions -----------------

if __name__ == "__main__":
    main()
```

```
fasta2phy.py -f lassa_seqs.fasta
```

Executing script in working directory

```
./fasta2phy.py -f lassa_seqs.fasta
```

Debugging Exercises

Traceback error format

- Reports the sequence of function calls that led to the error
- Lowest level is where the error actually occurred

Traceback example

Code:

```python
def fav_ice_cream():
    ice_creams = [
        "chocolate",
        "vanilla",
        "strawberry"
    ]
    print(ice_creams[3])

fav_ice_cream()
```

Error:

```
---------------------------------------------------------------------------
IndexError    Traceback (most recent call last)
<ipython-input-1-70bd89baa4df> in <module>()
      6     print(ice_creams[3])
      7
----> 8 fav_ice_cream()

<ipython-input-1-70bd89baa4df> in fav_ice_cream()
      4         “vanilla",        “strawberry"
      5     ]
----> 6     print(ice_creams[3])
      7
      8 fav_ice_cream()

IndexError: list index out of range
```