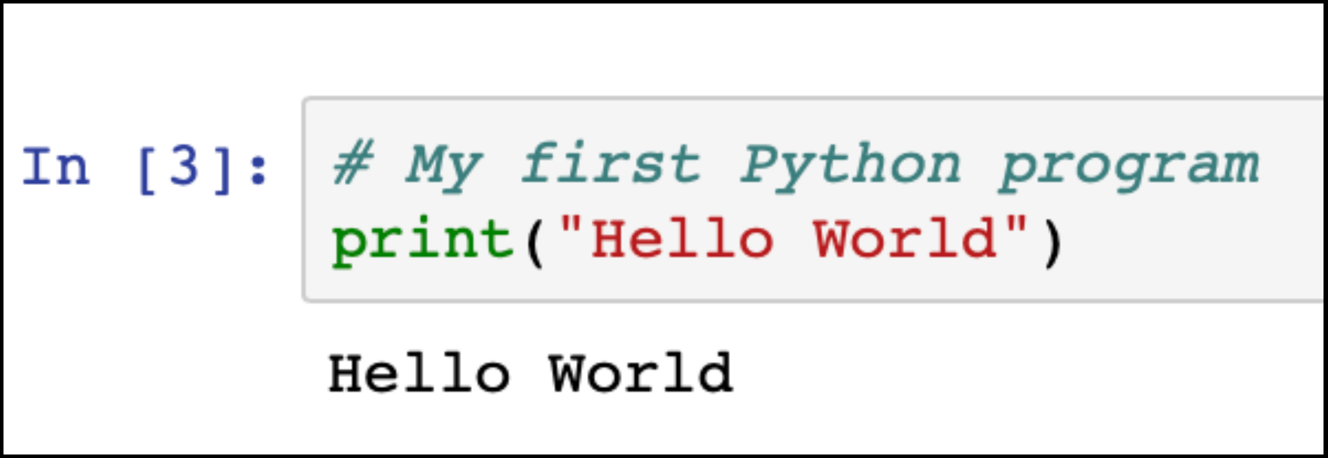
Algorithm

A precise sequence of simple steps to solve a problem

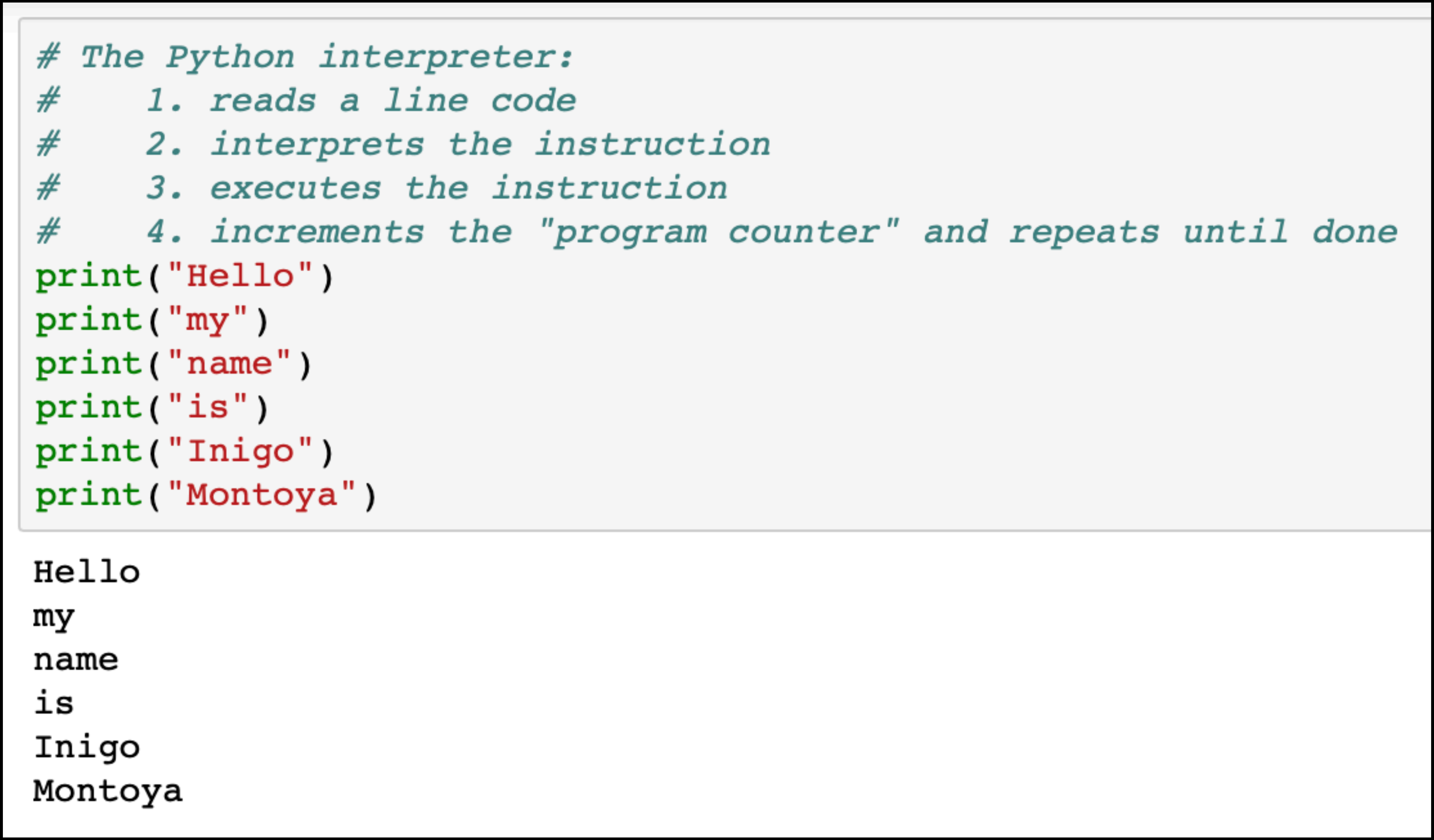
Python

translating an algorithm into a computer program



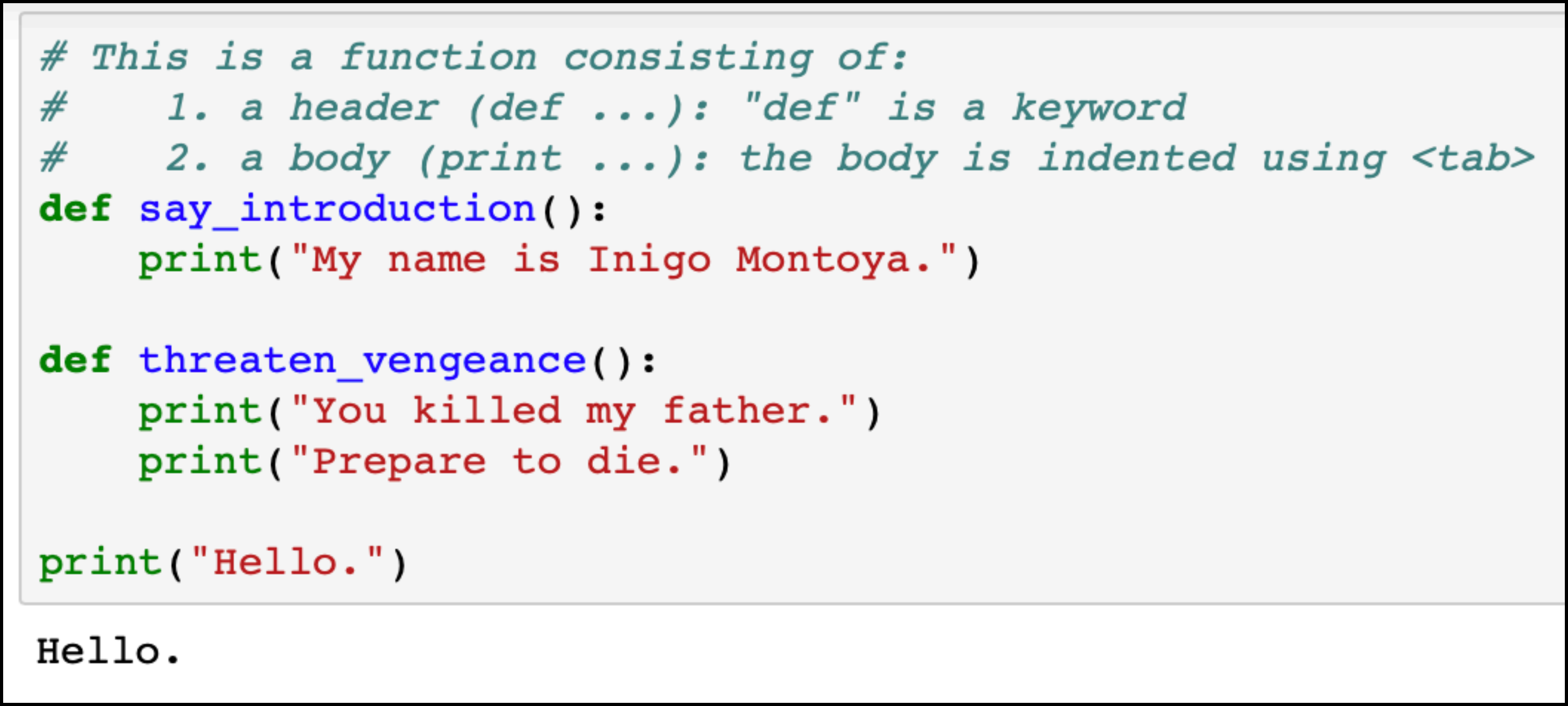
```
# My first Python program
print("Hello World")
```

```
Hello World
```



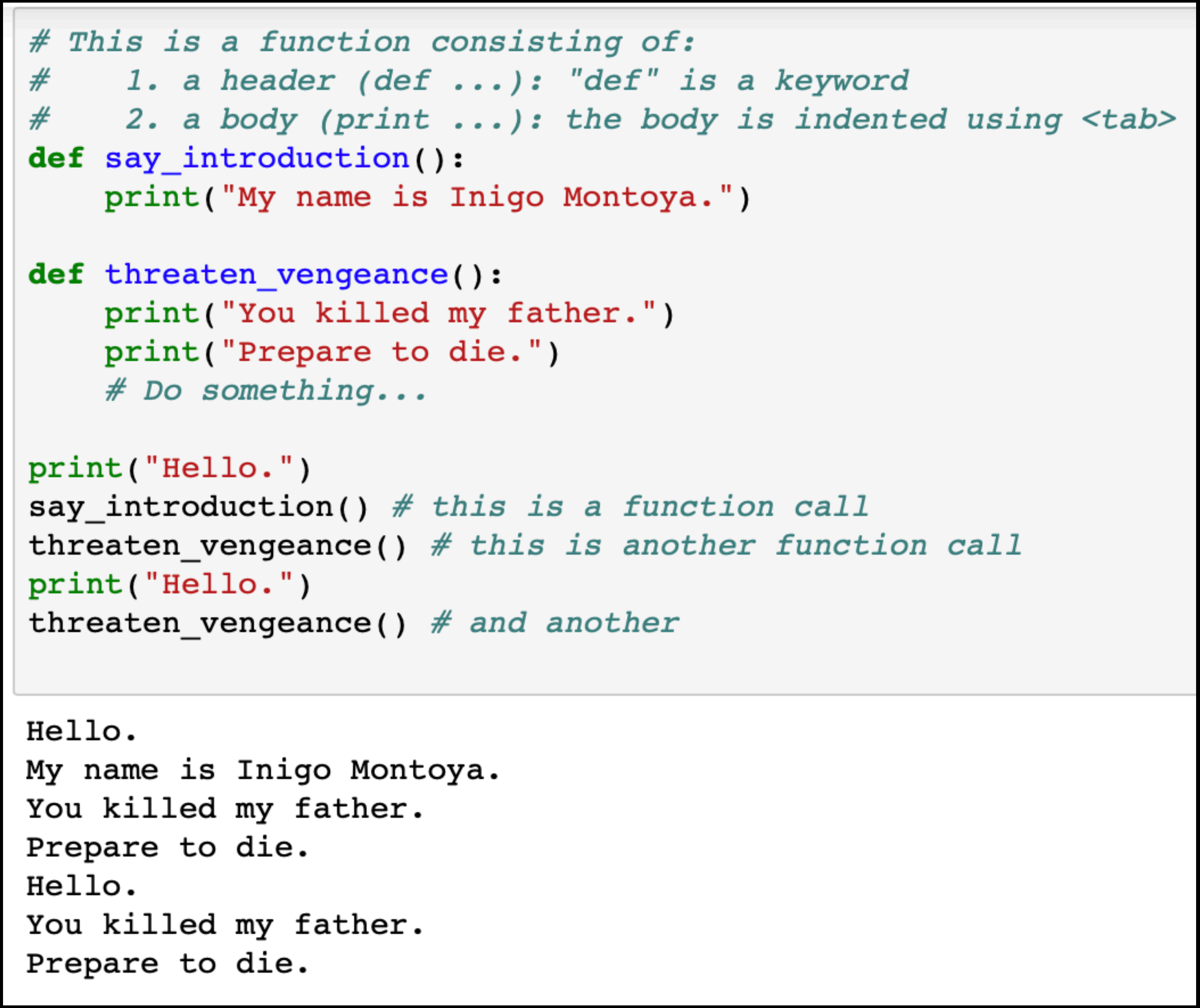
```python
# The Python interpreter:
#     1. reads a line code
#     2. interprets the instruction
#     3. executes the instruction
#     4. increments the "program counter" and repeats until done
print("Hello")
print("my")
print("name")
print("is")
print("Inigo")
print("Montoya")
```

```
Hello
my
name
is
Inigo
Montoya
```



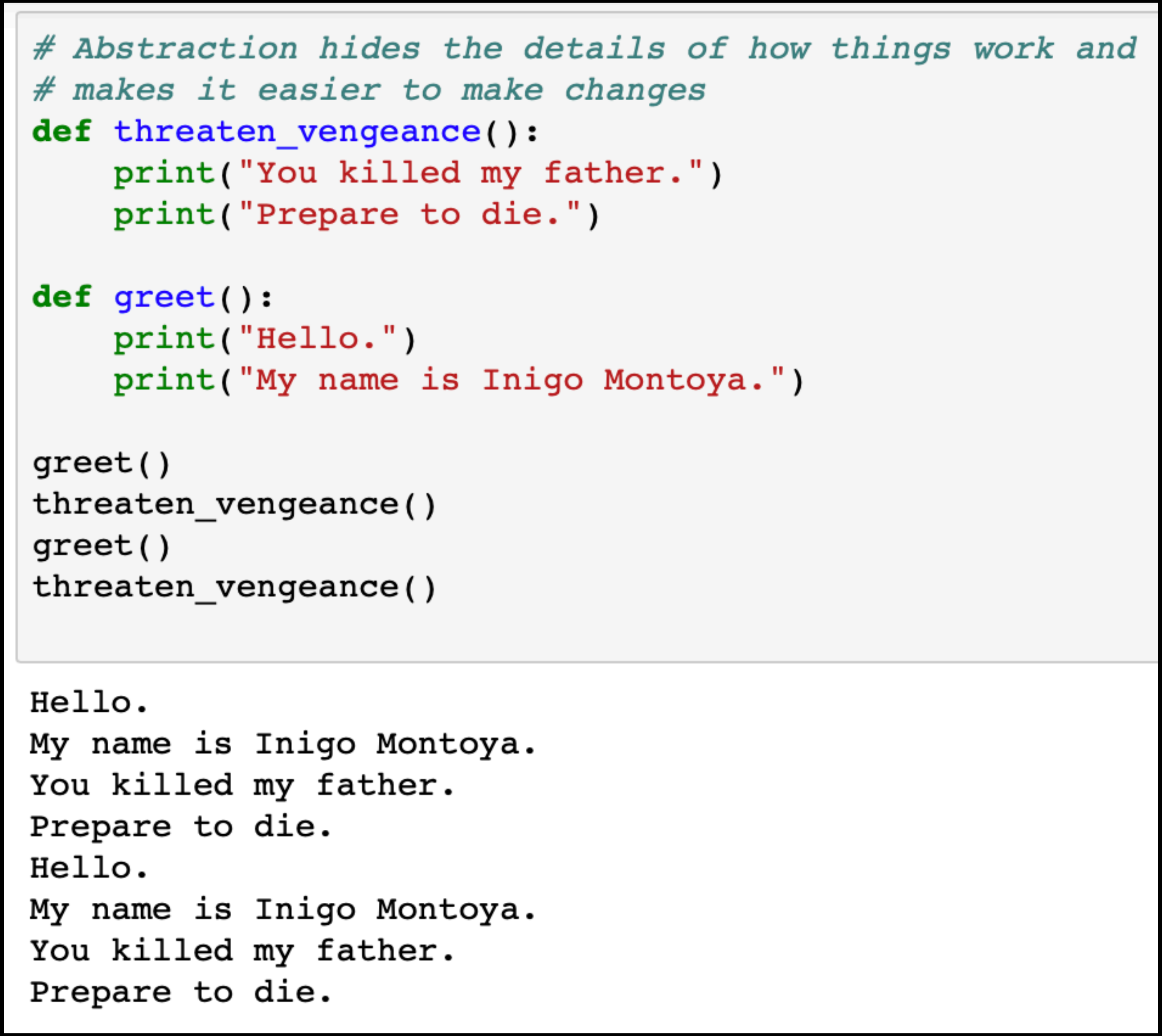
```python
# This is a function consisting of:
#    1. a header (def ...): "def" is a keyword
#    2. a body (print ...): the body is indented using <tab>
def say_introduction():
    print("My name is Inigo Montoya.")

def threaten_vengeance():
    print("You killed my father.")
    print("Prepare to die.")

print("Hello.")
```

```
Hello.
```



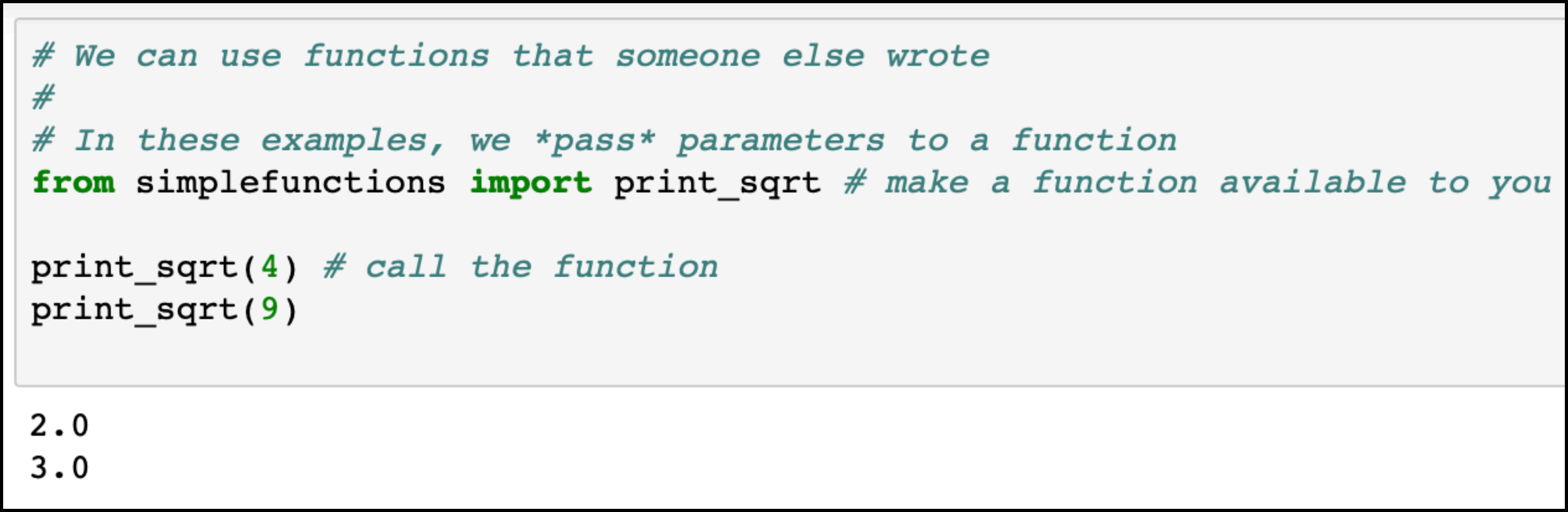
```python
# This is a function consisting of:
#    1. a header (def ...): "def" is a keyword
#    2. a body (print ...): the body is indented using <tab>
def say_introduction():
    print("My name is Inigo Montoya.")

def threaten_vengeance():
    print("You killed my father.")
    print("Prepare to die.")
    # Do something...

print("Hello.")
say_introduction() # this is a function call
threaten_vengeance() # this is another function call
print("Hello.")
threaten_vengeance() # and another
```

```
Hello.
My name is Inigo Montoya.
You killed my father.
Prepare to die.
Hello.
You killed my father.
Prepare to die.
```



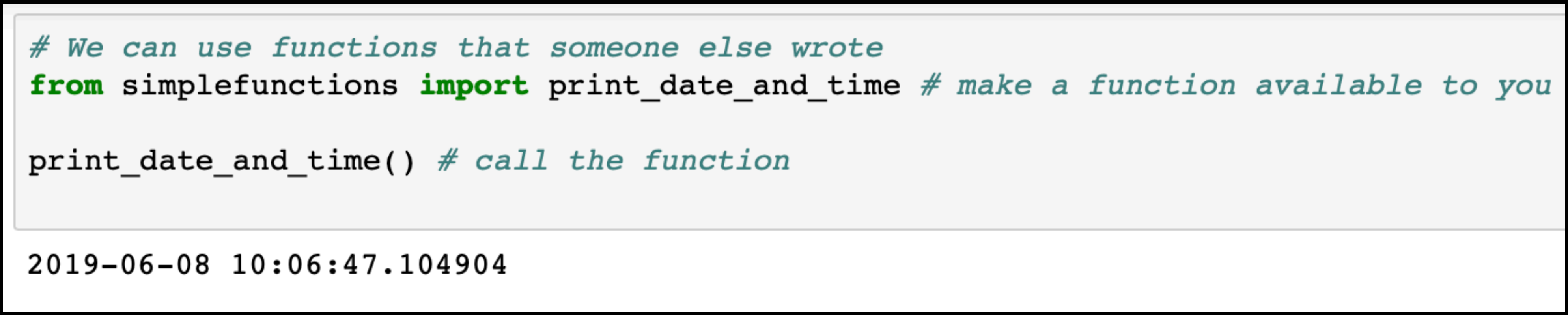
```
# Abstraction hides the details of how things work and
# makes it easier to make changes
def threaten_vengeance():
    print("You killed my father.")
    print("Prepare to die.")

def greet():
    print("Hello.")
    print("My name is Inigo Montoya.")

greet()
threaten_vengeance()
greet()
threaten_vengeance()
```

```
Hello.
My name is Inigo Montoya.
You killed my father.
Prepare to die.
Hello.
My name is Inigo Montoya.
You killed my father.
Prepare to die.
```



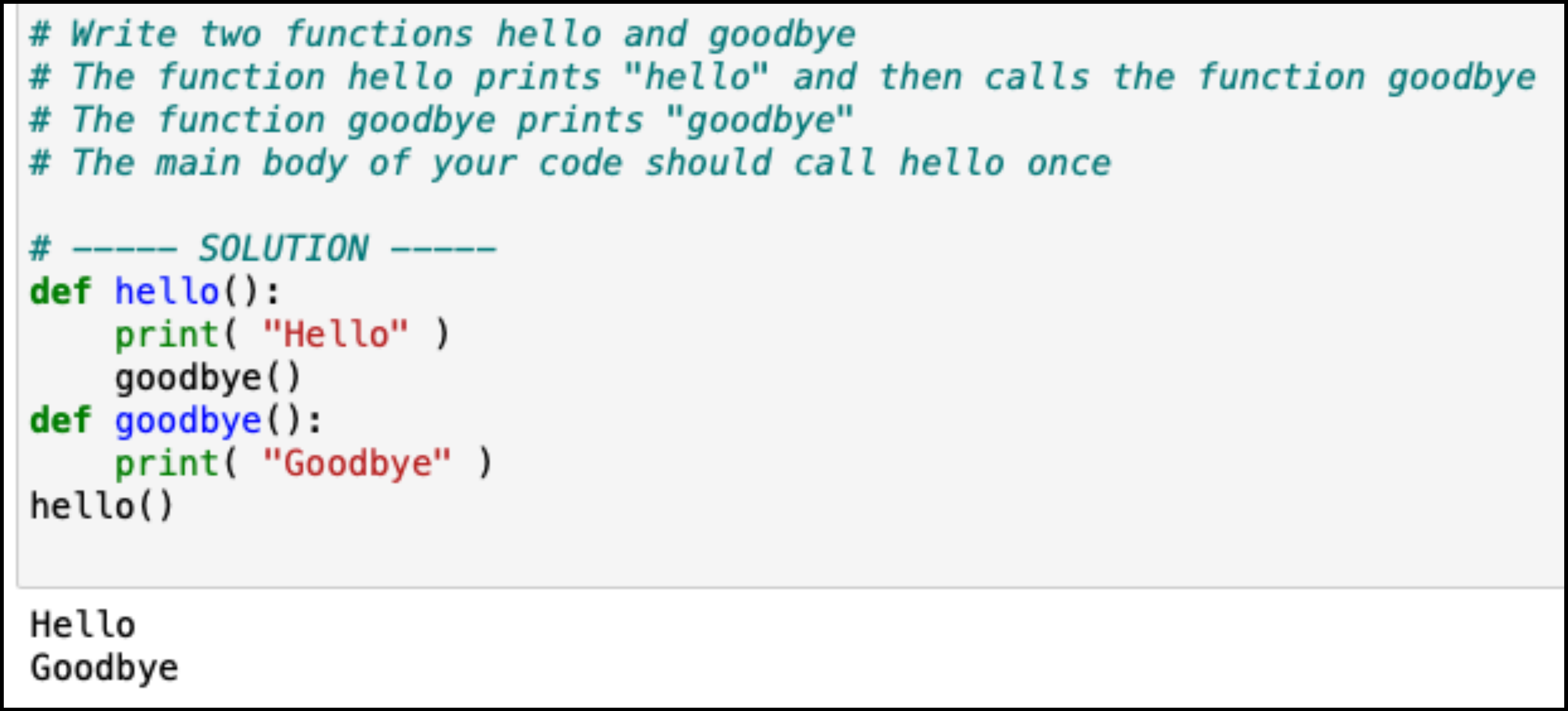
```
# We can use functions that someone else wrote
#
# In these examples, we *pass* parameters to a function
from simplefunctions import print_sqrt # make a function available to you

print_sqrt(4) # call the function
print_sqrt(9)
```

```
2.0
3.0
```



```python
# We can use functions that someone else wrote
from simplefunctions import print_date_and_time # make a function available to you

print_date_and_time() # call the function
```

```
2019-06-08 10:06:47.104904
```



```
# Write two functions hello and goodbye
# The function hello prints "hello" and then calls the function goodbye
# The function goodbye prints "goodbye"
# The main body of your code should call hello once

# ----- SOLUTION -----
def hello():
    print( "Hello" )
    goodbye()
def goodbye():
    print( "Goodbye" )
hello()
```

```
Hello
Goodbye
```

Type: int

```
meaning_of_life = 42

print( meaning_of_life )

output: 42
```

Type: floating-point

```
a = 6.02
```

Type: string

```
last_letter = "z"

print( last_letter )

output: z
```

Type: string

```
print( "hello" )

output: hello



hello = 5

print( hello )

output: 5
```

Type: string

```
print( "4 + 7" )        output: 4 + 7

print( 4 + 7 )          output: 11
```

Type: string

```
print( 4 + 7 )

output: 11


print( "hello " + "my name" )

output: hello my name
```

Type: conversion

```
print( float(4) )               4.0

print( int(3.14) )              3

print( str(4) + str(2) )        42

print( int("4") + int("2") )    6
```

Type: boolean

```
x = True       # not same as x = “True”

y = False      # not same as y = “False”
```

Type: functions

```
max(3,4) -> 4

f = max
f(3,4)   -> 4
```

Type: functions

```
min(3,4) -> 3

min = max
min(3,4) -> 4
```

Expressions and Operators

addition	+	
subtraction	-	
multiplication	*	
division	/	4/3 -> 1.3333333333333333
int division	//	4//3 -> 1
exponentiation	pow	pow(2,3) -> 8
modulus (mod)	%	9 % 4 -> 1

Summary

- Variables
  - store information in computer memory
  - int, float, string, booleans, functions
- Expressions and Operators
  - arithmetic
  - similar to functions
  - assignment

Practice

```
a = 5
b = 3
c = a + b
d = "c: " + str(c)
```

Practice

```
a = 5
b = 3
c = a + b
d = "c: " + str(c)    c: 8

b = 30
a = b
```

Practice

```
a = 5
b = 3
c = a + b
d = "c: " + str(c)     c: 8

b = 30
a = b                  a -> 30

print(e)
```

Practice

```
a = 5
b = 3
c = a + b
d = "c: " + str(c)    c: 8

b = 30
a = b                 a -> 30

print(e)              error

4 = a
```

Practice

```
a = 5
b = 3
c = a + b
d = "c: " + str(c)     c: 8

b = 30
a = b                  a -> 30

print(e)               error

4 = a                  error
```

Practice

```
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )
```

Practice

```
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )                    error
```

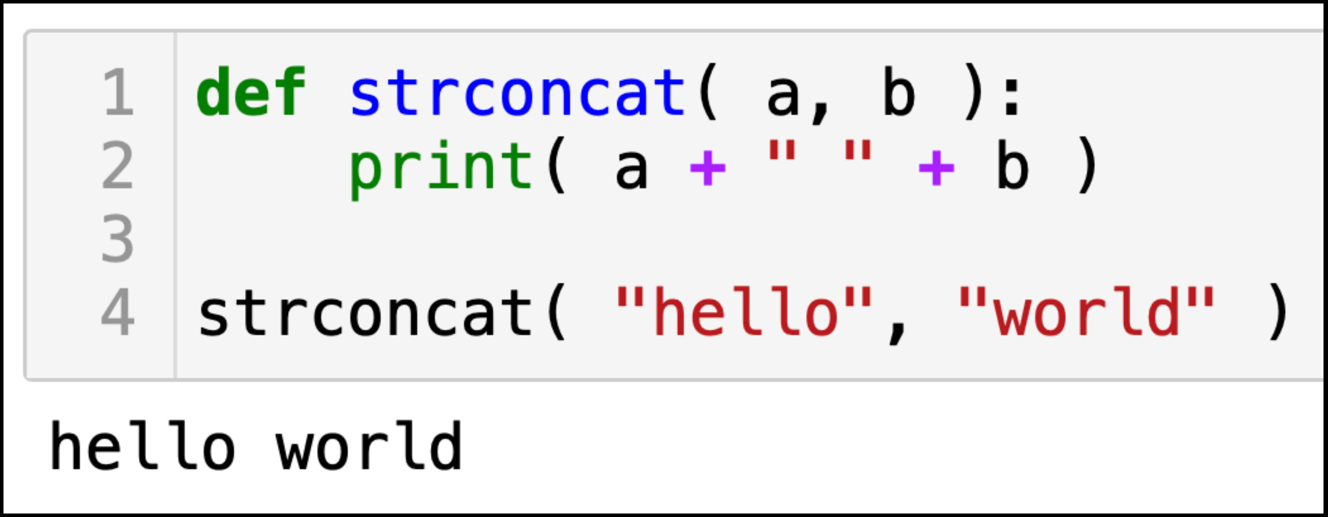
Practice

```
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )                  error

print( ??? )                    3 hello
```

Practice

```
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )                    error

print(str(b) + "  " + c )) 3 hello
```

Passing Values



```python
def strconcat( a, b ):
    print( a + " " + b )

strconcat( "hello", "world" )
```

```
hello world
```

Returning Values

```
# the function sqrt takes as input a number and returns a number
from math import sqrt
x = sqrt(4)

# the operator "+" takes as input two numbers and returns a number
x = 8 + 12

# the function len takes as input a string and returns an integer
x = len("eggplant")

# the return value of one function can be the input to another
x = int(8.485) + 12
x = int(sqrt(72)) + 12
```

Returning Values

```
def compute_four():
    return 24 / 4 - 2

x = compute_four()                 4

y = 24 / compute_four()            6

print( compute_four() )            4
```

Passing & Returning

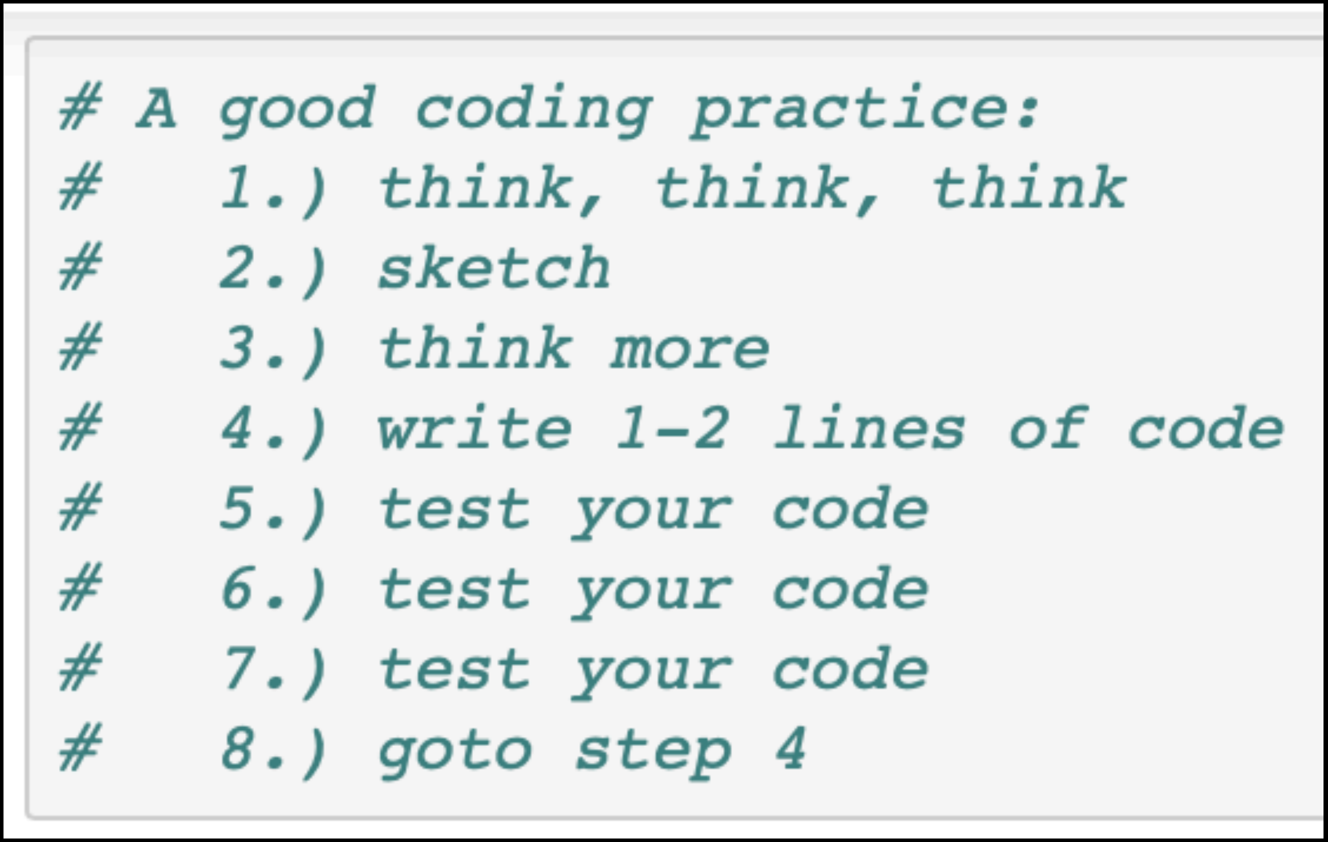
```
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)                        4

x = add_five(z)
print(x)                        9

print(add_five(z))              9
```

Passing & Returning

```
def return_two_things(x,y):
  return(x+y,x*y)

(s,p) = return_two_things(2,5)
```

Passing & Returning

```
def return_two_things(x,y):
  return(x+y,x*y)
  print(x,y)

(s,p) = return_two_things(2,5)
```



```
# A good coding practice:
#   1.) think, think, think
#   2.) sketch
#   3.) think more
#   4.) write 1-2 lines of code
#   5.) test your code
#   6.) test your code
#   7.) test your code
#   8.) goto step 4
```