Quiz I Review Session

Functions, Libraries, Variables and Error Messages

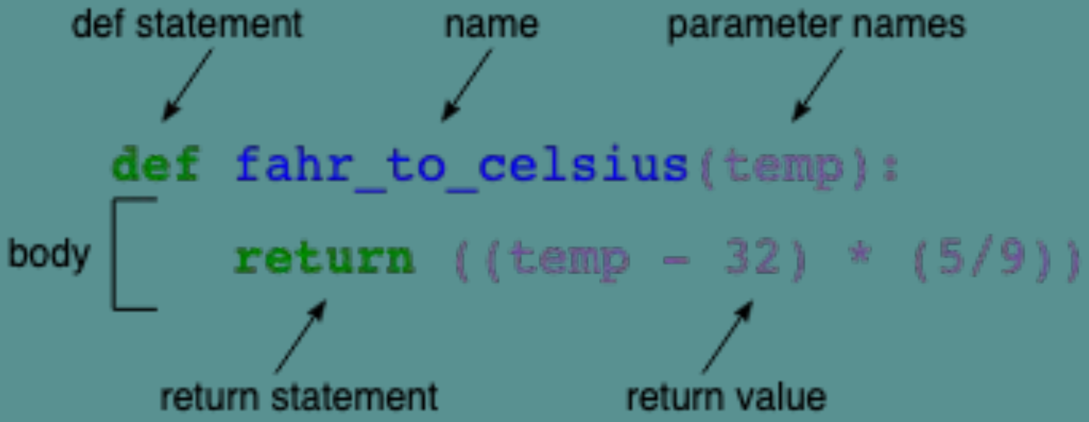
Topics

1. Using function definitions and function calls
2. Identify the argument(s) and returned values of a function call
3. Use libraries to import functions in categories like math, randomness, and graphics
4. Tracing function calls to understand how Python keeps track of nested function calls
5. Tracing variables and recognize the difference between local and global scope
6. Interpret and react to basic error messages caused by programs

Function Definitions

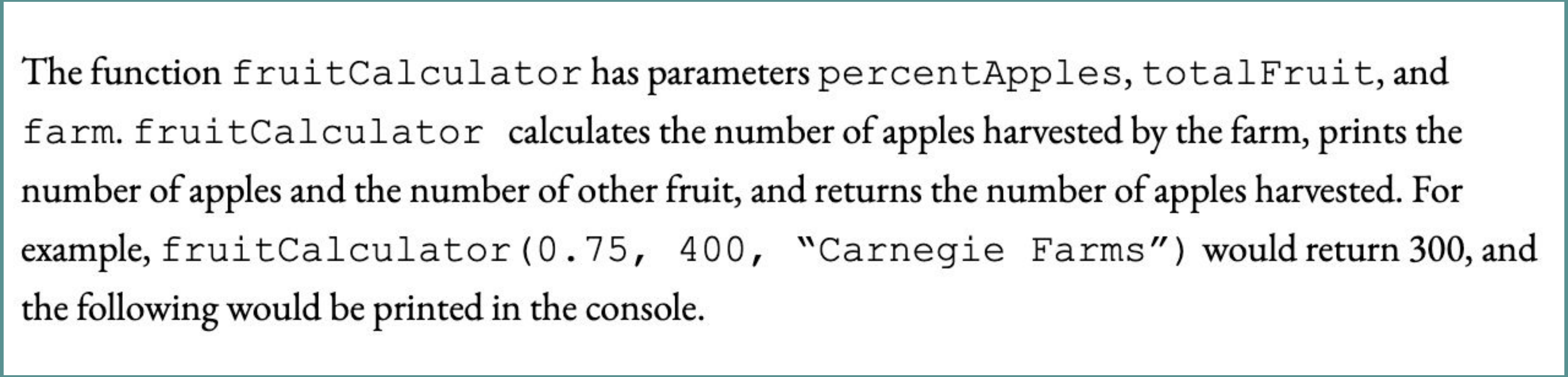
- Function definitions are used to define an algorithm that can be used several times.

- No execution occurs while the function is being defined, execution occurs on a function call



```
def fahr_to_celsius(temp):
    return ((temp - 32) * (5/9))
```

Function Writing Exercise

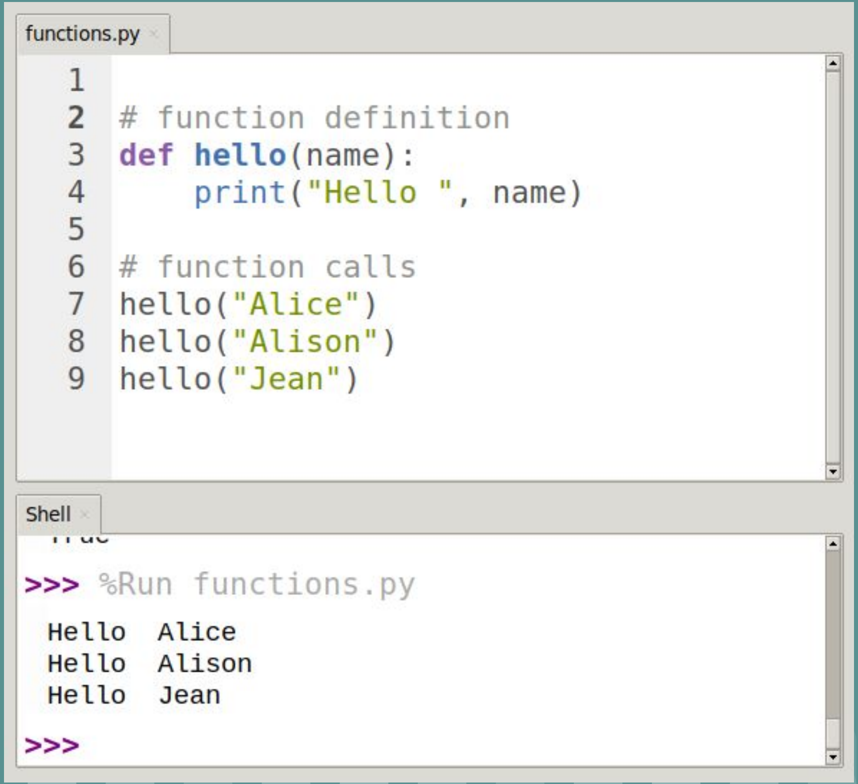


The function `fruitCalculator` has parameters `percentApples`, `totalFruit`, and `farm`. `fruitCalculator` calculates the number of apples harvested by the farm, prints the number of apples and the number of other fruit, and returns the number of apples harvested. For example, `fruitCalculator(0.75, 400, “Carnegie Farms”)` would return 300, and the following would be printed in the console.

Function Calls

- To execute a function we must call the function with a set of arguments.

- A function will always return a value. If there are no return statements it will return None by default.



```python

# function definition
def hello(name):
    print("Hello ", name)

# function calls
hello("Alice")
hello("Alison")
hello("Jean")
```

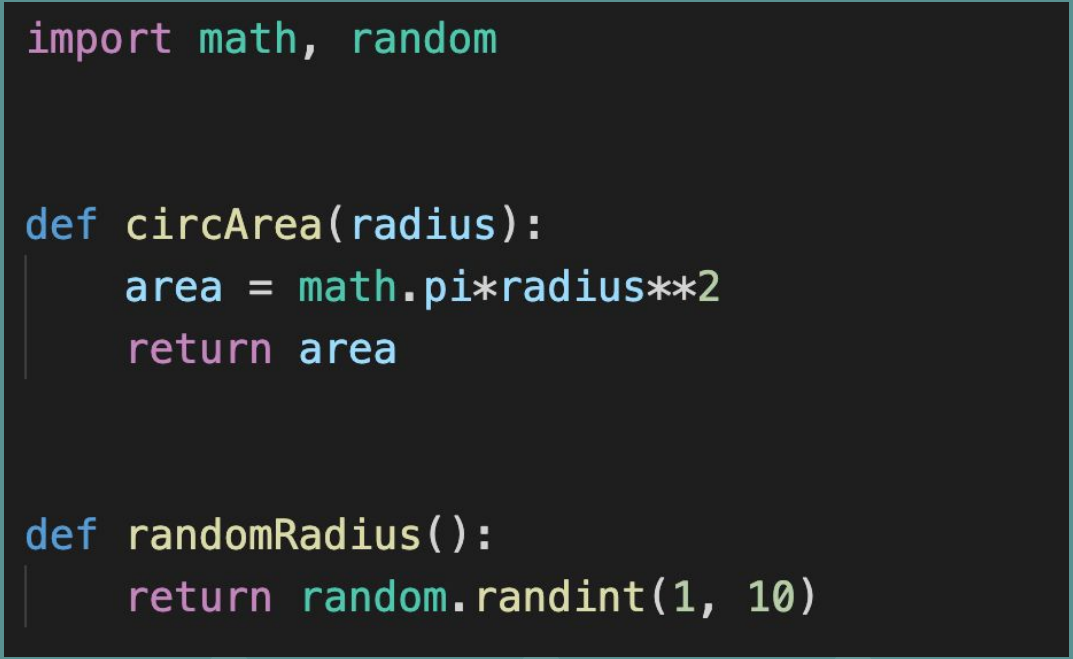
```
>>> %Run functions.py
 Hello  Alice
 Hello  Alison
 Hello  Jean
>>>
```

Function Calls Example

```
name = "Neil"
sport = "soccer"

def training(num):
    name = "Emily"
    num = num + 2
    print(name, "has trained up a level!")
    # Pause Here 1
    return num

def level():
    agi = 8
    awr = 7
    sta = 10
    ovr = (agi + awr + sta) / 3
    print(name, "has an overall rating of", ovr)
    awr = training(awr)
    print(name, "has a new overall rating of", (agi+awr+sta)/3)
    # Pause Here 2

level()
```

Libraries

- We can use functions that have been defined by others already through libraries.
- Libraries must be imported before they are used. You can find documentations of the different libraries there are online.

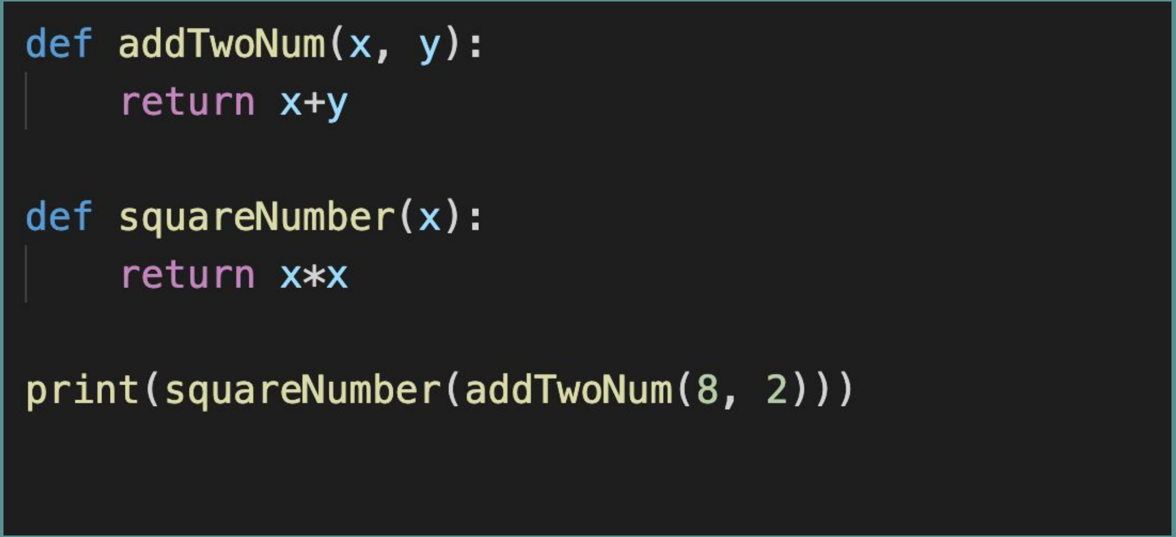


```python
import math, random


def circArea(radius):
    area = math.pi*radius**2
    return area


def randomRadius():
    return random.randint(1, 10)
```

Function Tracing

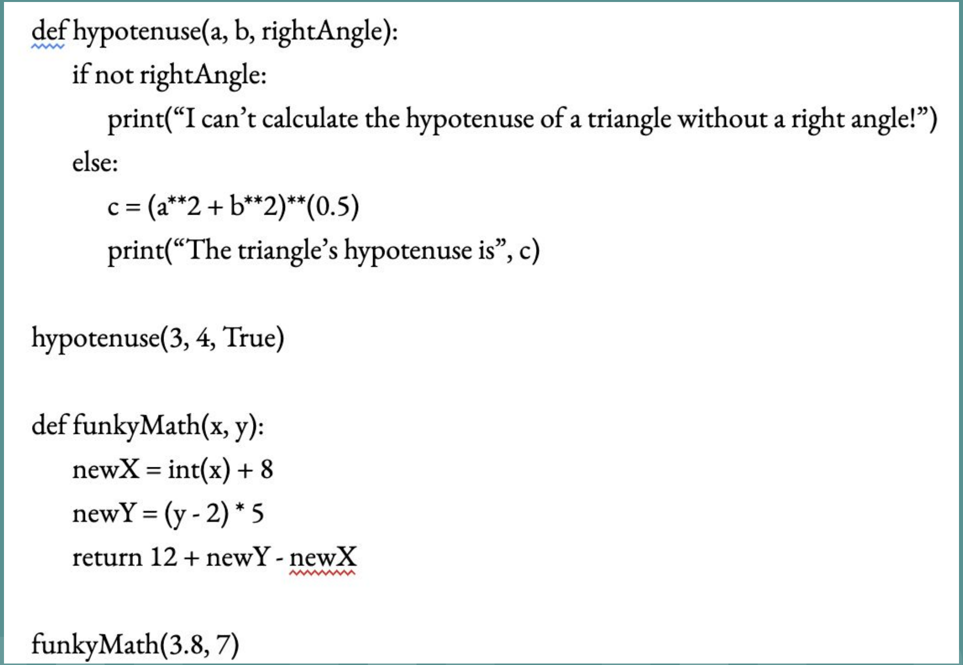
- Functions are executed from top to bottom.

- If a function call is used as an argument, it must be called before the outer function.

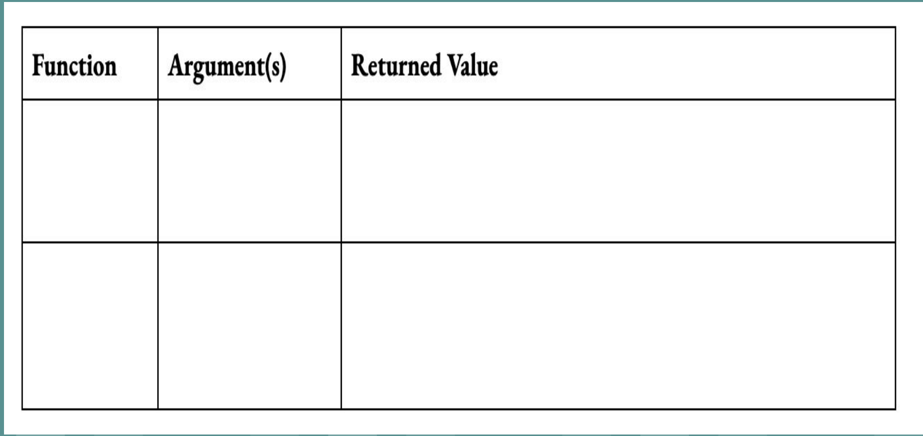


```python
def addTwoNum(x, y):
    return x+y

def squareNumber(x):
    return x*x

print(squareNumber(addTwoNum(8, 2)))
```

Function Tracing Problem



```
def hypotenuse(a, b, rightAngle):
    if not rightAngle:
        print("I can't calculate the hypotenuse of a triangle without a right angle!")
    else:
        c = (a**2 + b**2)**(0.5)
        print("The triangle's hypotenuse is", c)

hypotenuse(3, 4, True)

def funkyMath(x, y):
    newX = int(x) + 8
    newY = (y - 2) * 5
    return 12 + newY - newX

funkyMath(3.8, 7)
```

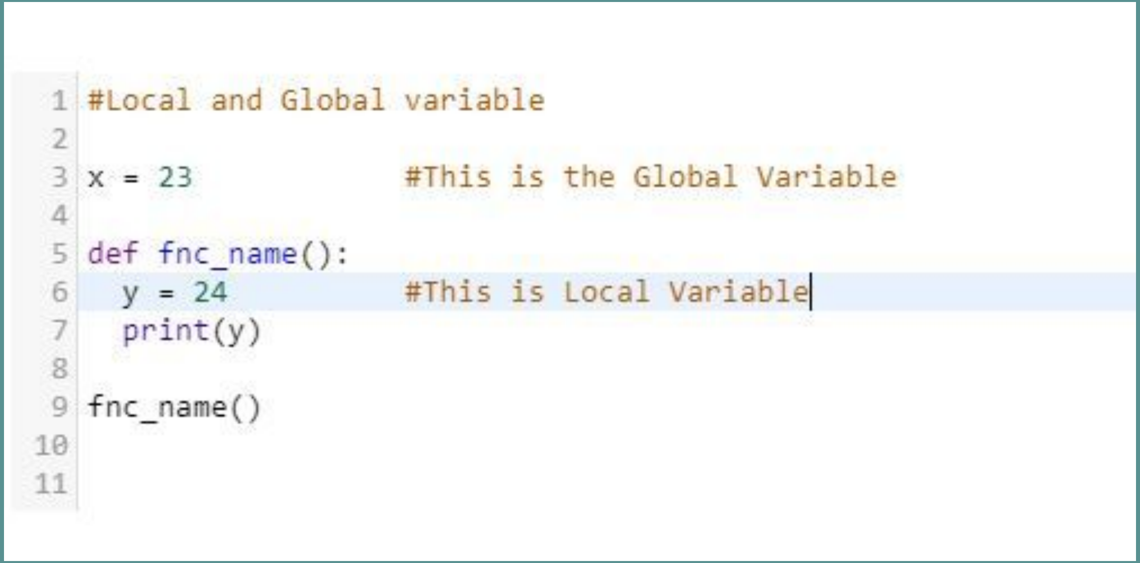


Function	Argument(s)	Returned Value

Variables

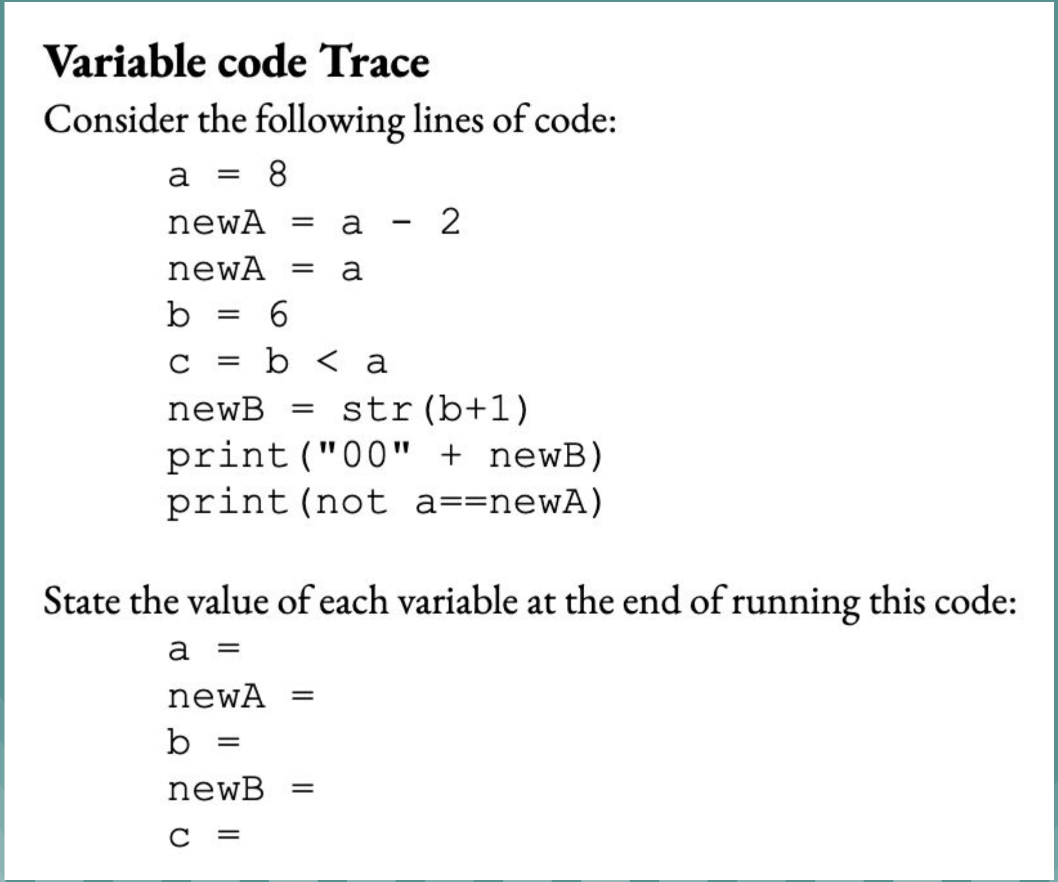
- Variables that are only defined within a function are called Local Variables. They can only be use within that function.

- Variables that are defined on the upper level outside of functions are called Global Variables. These can be used anywhere after they're defined.



```python
#Local and Global variable

x = 23                  #This is the Global Variable

def fnc_name():
  y = 24                #This is Local Variable
  print(y)

fnc_name()


```

Variable tracing example



Variable code Trace

Consider the following lines of code:

```
a = 8
newA = a - 2
newA = a
b = 6
c = b < a
newB = str(b+1)
print("00" + newB)
print(not a==newA)
```

State the value of each variable at the end of running this code:

```
a =
newA =
b =
newB =
c =
```

Error Messages

Syntax Error: Errors caused by a script that does not follow the correct syntax of Python.

ZeroDivisionError: Dividing by 0 has an undefined behavior, Python would call an error message in this instance.

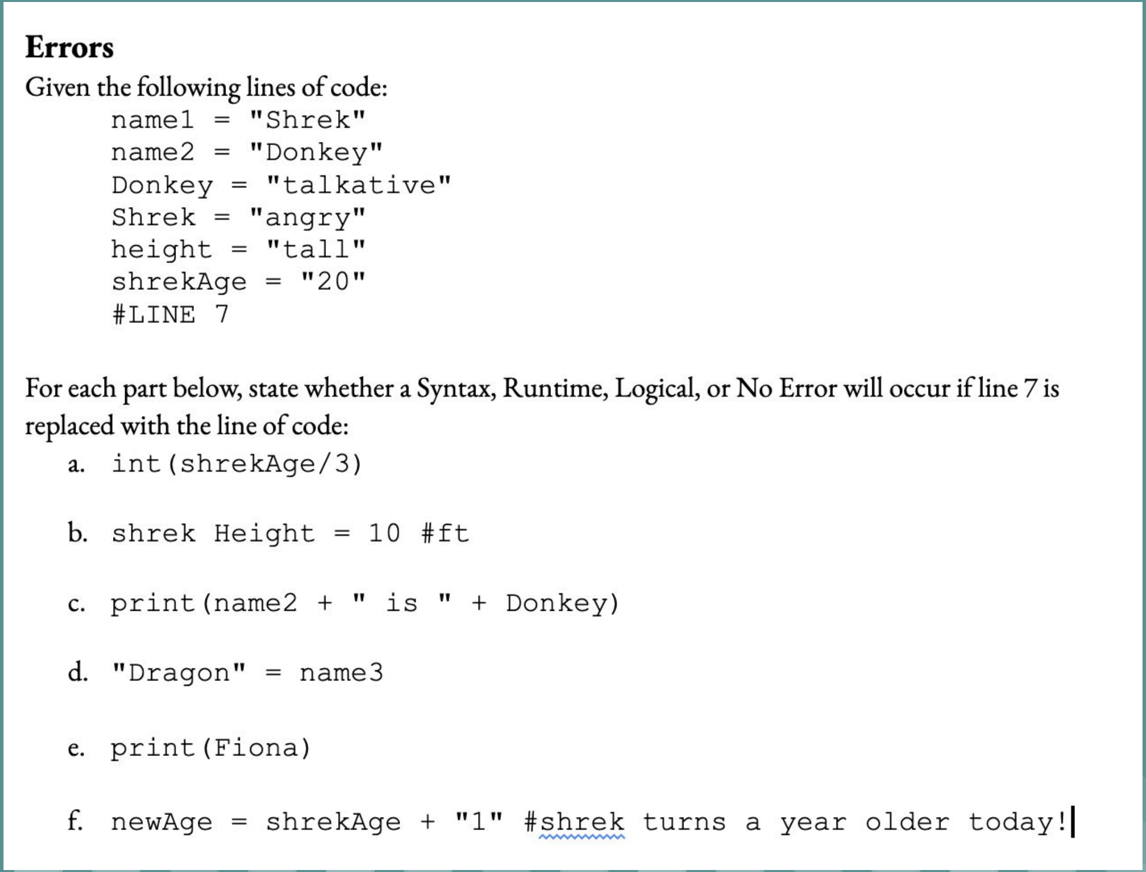
EOL Error: EOL is called an End-Of-Line error. Usually caused when there is a quote or parenthesis that was not closed.

nameError: Error caused when trying to refer to a variable that has not been defined yet.

typeError: When trying to do an operation with elements of the wrong type will result in a type error.

assertionError: This error occurs when a an assertion call was not validated.

Error Messages exercise



Errors

Given the following lines of code:

```
    name1 = "Shrek"
    name2 = "Donkey"
    Donkey = "talkative"
    Shrek = "angry"
    height = "tall"
    shrekAge = "20"
    #LINE 7
```

For each part below, state whether a Syntax, Runtime, Logical, or No Error will occur if line 7 is replaced with the line of code:

a. `int(shrekAge/3)`

b. `shrek Height = 10 #ft`

c. `print(name2 + " is " + Donkey)`

d. `"Dragon" = name3`

e. `print(Fiona)`

f. `newAge = shrekAge + "1" #shrek turns a year older today!`