```
1  # The entry point for a Python program is the first line of the file.
2  # Each line is read and executed by the computer, one at a time.
3
4  # Anything that follows the # symbol on a line is called a "comment". Python
5  # ignores comments when executing a program.
6
7  # Another name for a line of a program is a "statement". Statements
8  # change the *state* of a program.
9  print("Hello")
10
11 # Define: state
12
13
14
15 # Define: statement
16
17
18
19
20 # Contrast with an expression, which does not change the state.
21 11 + 2
22
23 # Define: expression
24
25
26
27
28
29 # print is a "function": a command to Python to do something in particular, in
30 # this case print text to the screen. We say we are "calling" the function print.
31 # (Synonyms: invoke, run.)
32
33 # An open parenthesis always follows a function's name. An "argument list"
34 # follows the parenthesis, consisting of one or more values to "give" to the
35 # function. Each is called an argument (sometimes called a "parameter"). On line
36 # 4, "Hello" is the argument to print.
37
38 # Any value surrounded in quotation marks is called a string. A string consists of
39 # 0 or more characters, which are invidividual symbols that can be printed to the
40 # screen. "Hello" consists of five characters.
41
42
43 # A Python statement is always a single line of code. Sometimes you want a
44 # statement to span multiple lines.
45 print('Some people choose to see the ugliness in this world, the disarray. '\
46       'I choose to see the beauty. To believe there is an order to our days.'\
47       'A purpose.')
48
49
50 # We can also print non-strings.
51 print(5)
52 # 5 is an integer, or an "int". Any positive or negative whole number is an int.
```

```
53 print(5.0)
54 # 5.0 is a floating-point number, or a "float". Floats are used for real numbers
55 # (the math definition of real number).
56 print("5")
57 # "5" is a string containing the character whose symbol is "5". It is not the same
58 # as the integer 5, but that is for another lesson.
59
60
61 # 5, 5.0, and "5" are examples of objects.
62
63 # Define: "object", "instance", and "type"
64
65
66
67
68
69
70 # Lessons:
71 #   print() - prints a value to the console
72 #   Strings, integers, floating-point numbers - what are they?
73 #   Multiline statements
74 #   Objects, types, values, and operations.
75
76
```

```python
# A "variable" is a named object that has a value of some type.

my_name = "Neal"
# This creates a variable named "my_name", whose value is "Neal".
# I can now use the s name in my code as a placeholder for its value.

print(my_name) # prints "Neal"

# Giving a variable a value is called assignment. I can assign a new value later.
my_name = "Jaclyn"
print(my_name) # now prints "Jaclyn"

# Draw a model of what's happening in the computer's memory...









# Every object in Python has a type. We have seen several types already:
# int
# float
# str

# but there are more:
# complex (complex numbers)
# bool (Boolean values of True and False)
# list (an ordered sequence of values)
# tuple (a group of values)
# and many more.

# We can examine the type of a value by calling the "type" function, with either a literal value or a variable.
print(type(5)) # prints "int"
print(type(my_name)) # prints "str"


# We can apply many operators to values and variables to manipulate them.
# What do each of these statements print?
print(my_name + " Terrell") # concatenation
print(5 + 10) # addition
print(5.0 * 10.0) # multiplication
print(5 - 10) # subtraction
print (5 / 2) # division with remainder (decimal division)
print (5 // 2) # division without remainder (round down / truncate)
print (5 % 2) # modulo (remainder)

# Things to remember:
#   Variables - how to create? How to assign? How to print?
```

```
52  #   Types - int, float, str, bool... what are these?
53  #   type() - gives the type of a value
54  #   Operators
55  #       +, -, *, /, //, % on numbers
56  #       + on strings
57
```

```python
# In this program we will capture user input into a variable in order
# to calculate something interesting.

import math
# import lets us use "modules" with useful functions in them. The "math"
# module has a variable called math.pi that we will use later.


# The input() function will cause the program to pause until the user types
# something with the keyboard and presses Enter. Once they do, whatever they
# typed will be given back to you and you can save that input in a variable.

# If you give a string argument to input(), it will print that string as a
# "prompt" to the user.
name = input("What is your name? ")

age = input("What is your age? ")
print("Oh. I am {0} years old.".format(age * 5))

# It's important to remember: input() always gives back a STRING. If you
# want to do math on the input, you must convert it to an appropriate type.
# What type should someone's age be: string, int, or float?

age = int(input("Let's try that again. What is your age? "))
# Wrapping the input function call inside int() converts the string input
# into an integer. Now when we multiply, we will get an integer.


diameter = float(input("What is the diameter of your car's tires in inches? "))
circumference = diameter * math.pi
print("Your tires have a circumference of {0} inches.".format(circumference))

# Things to remember:
#   input() - prompts the user to enter some value, given as a string.
#   int() - converts some other type (like a string) to an int
#   float() - likewise, for a float
#   import - lets you use useful functions from modules like math.

```

```python
# This program solves a quadratic equation in the form ax^2 + bx + c = 0.
import math

a = float(input("Enter A: "))
b = float(input("Enter B: "))
c = float(input("Enter C: "))

discriminant = b * b - 4 * a * c
x1 = (-b + math.sqrt(discriminant)) / (2 * a)
x2 = (-b - math.sqrt(discriminant)) / (2 * a)

print("The two solutions to " + str(a) + "x^2 + " + str(b) + "x + " + str(c) + \
    " are")
print("x1 = " + str(x1))
print("x2 = " + str(x2))

# Lessons:
#   Just more input/output practice.
#   Using math.sqrt
```

```python
# This program determines the fewest number of quarters, dimes, nickels, and
# pennies necessary to make a certain amount of money.

# Sometimes we need to use number literals in our code that might not make sense
# to someone reading our work. In that case, it is better to create a "constant"
# variable with the needed value, and use that variable in place of the number.

# By tradition, constant variables are named with ALL_CAPS
QUARTER_VALUE = 25
DIME_VALUE = 10
NICKEL_VALUE = 5
DOLLAR_VALUE = 100

amount = float(input("How many dollars would you like to make into change? "))
numberOfCents = int(amount * DOLLAR_VALUE)
 # Because amount is a float, multiplying it by 100 will give back a float.
 # Since cents must be a whole number, we convert the result to an int.

# To find the number of quarters needed, we divide by 25 and ignore the
# remainder.
numberOfQuarters = numberOfCents // QUARTER_VALUE

# To find the money amount left over, we need the remainder after dividing by 25.
leftOver = numberOfCents % QUARTER_VALUE

# Repeat for dimes.
numberOfDimes = leftOver // DIME_VALUE
leftOver = leftOver % DIME_VALUE

# Repeat for nickels.
numberOfNickels = leftOver // NICKEL_VALUE
leftOver = leftOver % NICKEL_VALUE

# Whatever is leftover is the number of pennies.
numberOfPennies = leftOver

print("You need {0} quarters, {1} dimes, {2} nickels, and {3} pennies." \
      .format(numberOfQuarters, numberOfDimes, numberOfNickels, numberOfPennies))

# Lessons:
#   Use of constants
#   Using // and % for integer division and remainders

```