Python 2 - Answer Key

Contents

Answer Key

Unit 0: Primer Solutions

- ### Python Syntax Solutions

Do It Yourself

```
for i in range(9):
    print(i)
```

- ### Flow Control Solutions

Inside the Computer

```
forward()
forward()
```

More Memory Chips

```
forward()
forward()
forward()
```

Around the Corner

```
forward()
turn_right()
forward()
```

Scale the Slope

```
stun()
forward()
forward()
forward()
stun()
forward()
forward()
forward()
```

- ## Variables Solutions

Trickier Trap 1

```
L1 = 5 * x
```

Trickier Trap 2

```
L1 = 7 * x ** 5
```

Trickier Trap 3

```
L1 = x ** 2 + 2 * x + 1
```

Trickier Trap 4

```
L1 = (3 * x - 2) / (1 + 2 * x) ** 2
```

Unit 1: Advanced Expressions Solutions

- ## Boolean Logic Solutions

Gymnastics Score

```
print(frontflip*1 + handspring*2 + cartwheel*3 + backflip*4)
```

Pythagorean Theorem

```
print(a*a + b*b == c*c)
```

Not divisible

```
print(a % b != 0)
```

What is the difference between == and is?

Answer: The `==` operator compares values, while the `is` operator compares memory locations.

Valid triangle

```
print((a+b > c) and (a+c > b) and (b+c > a))
```

Is the Number a Teen?

```
print(13 <= number <= 19)
```

Pizza Delivery

```
print("01st street" <= address <= "46th street")
```

Alphabet position

```
letter = "a"

print(ord(letter) - ord("a") + 1)
print(ord(letter) - 96) # alternative solution
```

Print the letter

```
position = 1

print(chr(position + ord("A") - 1))
print(chr(position + 64)) # alternative solution
```

Test It Out

```
n = 22

print(n >= 10 and n < 100 and n % 2 == 0)
```

How Many Digits?

```
print(10 <= n <= 99 and n % 2 == 0 or 100 <= n <= 999 and n % 2 == 1)
```

Programmer Happiness

```
smiling = True
sick = True
coding = True

print(smiling and not sick and coding)
```

Exact Change

```
amount = 100
```

```
given = 100

print(not amount-given)
```

Coupons

```
shipping = 10
if coupon:
    shipping = 0
print(price + shipping)

# Alternative - technically correct but makes less sense
# Since coupon is "" by default, shipping should be 10 be default
shipping = 0
if not coupon:
    shipping = 10
print(price + shipping
```

)

• Advanced Math Operations Solutions

Temperature Conversions

```
C = 35

print(C * 9 / 5 + 32)
```

Finding e

```
n = 3

print((1 + (1/n)) ** n)
```

Divvying Up Candy

```
candy = 50
friends = 7

print(candy % friends)
```

Test It Out

```
n = 12.34567
d = 3
```

```
print(('%.' + str(d) + 'f') % n)
```

New Heights

```
height = 60
print(height)

height += 4
print(height)

height += 6
print(height)

height += 3
print(height)
```

Bacteria Population

```
bacteria = 100

bacteria *= 2      # 1 hour
print(bacteria)
bacteria *= 2      # 2 hours
print(bacteria)
bacteria *= 2      # 3 hours
bacteria *= 2      # 4 hours
bacteria *= 2      # 5 hours
print(bacteria)
```

Sea Levels

```
print("Between 1995 and %d, sea levels rose by %.2f mm." % (2000, 9.83))
print("Between 1995 and %d, sea levels rose by %.2f mm." % (2005, 9.83 +
28.62))
print("Between 1995 and %d, sea levels rose by %.2f mm." % (2010, 9.83 +
28.62 + 45.58))
print("Between 1995 and %d, sea levels rose by %.2f mm." % (2015, 9.83 +
28.62 + 45.58 + 63.09))
```

Fibonacci Sequence

```
n = 7

fib1 = 1
```

```
fib2 = 1
# The first fib3 calculated in the loop is the 3rd number
for _ in range(n-2):
    fib3 = fib2 + fib1
    fib1, fib2 = fib2, fib3
if n > 2:    # If not, fib3 doesn't exist!
    print(fib3)
else:        # prints 1
    print(fib2)
```

Golden ratio

```
n = 50    # Note: if n > 76, Python will calculate fib2/fib1 = 1
fib1 = 1
fib2 = 1
for _ in range(n):
    fib3 = fib2 + fib1
    fib1, fib2 = fib2, fib3
print("The golden ratio is approximately " + str(fib2 / fib1))
```

- ## Advanced String Operations Solutions

GC Content

```
dna = "ATGTACGATTATGATCCGTTATACCAGAGTTGTTCACGTGTCACAGCAAA"

total_nucleotides = len(dna)
gc_nucleotides = dna.count("G") + dna.count("C")
print("The DNA sequence's GC content is " + str(gc_nucleotides /
total_nucleotides))
```

AT/GC Ratio

```
dna = "ATGTACGATTATGATCCGTTATACCAGAGTTGTTCACGTGTCACAGCAAA"

at_nucleotides = dna.count("A") + dna.count("T")
gc_nucleotides = dna.count("G") + dna.count("C")
print("The DNA sequence's AT/GC ratio is " + str(at_nucleotides /
gc_nucleotides))
```

Hair Genes

```
if "TAGCGGT" in mother_dna and "TAGCGGT" in father_dna:
    print("black")
```

```
elif "TAGCGGT" in mother_dna or "TAGCGGT" in father_dna:
    print("brown")
else:
    print("blonde")
```

In One String But Not the Other

```
# Find the characters which are in str1 but not in str2
str1 = "abcdefghijklmnopqrstuvwxyz"
str2 = "abdefhijkmnoprstvwyz"
extras = ""
for c in str1:
    if c not in str2:
        extras += c
print("The characters in str1 not in str2 are " + extras)
```

Counting Replacements

```
necklace = "bbbwbwwwwbwbwbwbwbbwbbwwwwbwbwbwbwbwb"
replacements = 2

for _ in range(replacements):
    necklace = necklace.replace("b", "X")
    necklace = necklace.replace("w", "bw")
    necklace = necklace.replace("X", "w")
print(necklace)
```

In Both Strings

```
a = "abcdefgh"
b = "efghijkl"

count = 0
for letter in a:
    if letter in b:
        count += 1
print(count)
```

Reversing a Name

```
name = "Codey"

reverse = ""
for c in name:
    reverse = c + reverse
print(reverse)
# Alternative way to reverse (and print)
print(name[::-1])
```

Palindromes

```
word = "racecar"

reverse = ""
for c in word:
    reverse = c + reverse
if reverse == word:
    print("palindrome")
else:
    print("not palindrome")
```

Space It Out

```
quote = "hello world"

new_quote = ""
for c in quote:
    new_quote = new_quote + c + " "
print(new_quote)
```

- ## Review and Quiz Solutions

Question 1: A teacher has given a homework assignment a certain `score`. If the homework was `late`, the student's final score is 50% of the graded score. Print the student's final score as an integer.

Answer:

```
score = 80
late = True

print(score * (1 - late * 0.5))
```

Question 2: Which of the following is False?

Answer: `1.0 > 1`

Question 3: Given a letter `grade` that is either "A", "B", "C", or "D", print the minimum score a student must get to achieve that grade. The minimum grade to get an A is 90, B is 80, C is 70, and D is 60.

Answer:

```
grade = "C"

if grade == "A":
    print(90)
elif grade == "B":
    print(80)
elif grade == "C":
    print(70)
else:
    print(60)
```

Question 4: Given an integer `a`, print `True` if it is divisible by `b` but not by `c`, and `False` otherwise.

Answer:

```
a,b,c = 10,5,3

print(a % b == 0 and a % c != 0)
```

Question 5: Given a number `n`, print `True` if it is an odd three-digit number or an even four-digit number. Print `False` otherwise.

Answer:

```
n = 1234

print(100 <= n <= 999 and n % 2 == 1 or 1000 <= n <= 9999 and n % 2 ==
0)
```

Question 6: Given the `balance` in a bank account and an amount to be removed in a `withdrawal`, print the total amount of money in the bank account after the withdrawal.

Answer:

```
balance = 1000
withdrawal = 900

print("${:,.2f}".format(balance - withdrawal))
```

Question 7: What was the output when you exceeded the maximum floating point number?

An overflow error

Question 8: Given the side length of a cube in the variable `length`, print the difference between the cube's volume and its surface area. Hint: The volume of a cube is length^3, and the surface area is the sum of the areas of every face. The area of a face of the cube is length^2, and a cube has six faces.

Answer:

```
length = 12

print(length**3 - 6*length**2)
```

Question 9: Given the `distance` to a destination and the `speed_limit` of the road leading there, print how long it would take to reach the destination in hours. Remember: distance = speed * time. Make sure to print your output as a float.

Answer:

```
distance = 100
speed_limit = 20

print("{:.2f} hours".format(distance / speed_limit))
```

Question 10: Given an integer `jackpot`, print the amount formatted with a dollar sign and commas separating digits.

Answer:

```
jackpot = 10000000

print("${:,}".format(jackpot))
```

- ## Unit Project Solutions

```
def input_money(str):
    money = 0
    while True :
        money = int(input(str))
        if(money < 0):
            print("Please enter a valid number.")
        else:
            break
    return money

watson = input_money("Welcome to Final Jeopardy.  How much money does
Watson currently have? ")
player_a = input_money("How much money does Player A have? ")
player_b = input_money("How much money does Player B have? ")

output = ""

if watson > player_a and watson > player_b :
    output += "Watson is in first place."
    second_place = max(player_a, player_b)
    if 2*second_place > watson:
        output += " Watson should wager " + str(second_place + 1)
    else:
        output += " Watson should wager " + str(0)
elif watson < player_a and watson < player_b :
    output += "Watson is in third place."
    output += " Watson should wager " + str(watson)
else:
    output += "Watson is in second place."
    first_place = max(player_a, player_b)
    third_place = min(player_a, player_b)
    if first_place - watson < watson :
        output += " Watson should wager " + str(first_place - watson +
1)
    else:
        output += " Watson should wager " + str(watson - third_place +
1)

print(output)
```

Unit 2: Functions Solutions

- ## Parameters Solutions

Multiply

```
def multiply(a, b):
    print(a*b)

multiply(3, 4)
```

Greet Three

```
def greet_three(name1, name2, name3):
    print("Hello {0}, {1}, and {2}!".format(name1, name2, name3))

greet_three("Codey", "Bodie", "Jodie")
```

- ## Returning Values Solutions

Squared

```
def squared(num):
    return num**2

print(squared(1))
print(squared(3))
print(squared(7))
```

Double Squared Plus One

```
def squared(num):
    return num**2

def double_squared_plus_one(num):
    return 2 * squared(num) + 1
```

- ## Review and Quiz Solutions

Question 1: Which of the following is a valid function header?

Answer:

```
def countdown():
```

Question 2: What is the output of the following code?

Answer:

```
yummy
yummy
yummy
```

Question 3: Which function has correctly defined parameters?

Answer:

```
def function4(num1, num2):
  print(num1 + num2)
```

Question 4: Parameters follow the same naming rules as variables.

Answer: True

Question 5: The `return` keyword may be used outside of a function.

Answer: False

Question 6: What is the output of the following?

Answer:

```
9
```

Question 7: If a variable is defined in the body of a function, which of the following regions can see the variable? Select all that apply.

Answer:
Code within the function
Code within a function defined within the first function

Question 8: Write a function `better_string()` that takes one string argument and returns the same string with "better" prepended to the string (at the beginning).

Answer:

```
def better_string(input_string):
    return "better " + input_string

print(better_string("ice cream"))
```

Question 9: Write a function `triple_operation()` that takes one number argument and performs the following operation three times: add 3, multiply by 5, then subtract 2. You should use an inner function.

Answer:

```
def triple_operation(num):
    def operation(num):
        return (num+3)*5 - 2
    return operation(operation(operation(num)))

print(triple_operation(3))
```

Question 10: Write a function `my_status()` that takes four string arguments: `time`, `mood`, `hunger`, `sleepy` and prints a message describing how you are feeling at the current time. `mood`, `hunger`, and `sleepy` should have a default argument such as "fine".

Answer:

```
def my_status(time, mood="fine", hunger="fine", sleepy="fine"):
    print("The current time is {0} and I feel:".format(time))
    print("Mood: {0}".format(mood))
    print("Hunger: {0}".format(hunger))
    print("Sleepiness: {0}".format(sleepy))

my_status("2:35 PM", sleepy="very")
```

- ## Unit 2 Project Solution

```
import turtle
screen = turtle.Turtle()
screen.getscreen().setup(750,750)
screen.hideturtle()

sky = turtle.Turtle()
sky.penup()
sky.speed(0)
sky.pencolor("blue")
sky.fillcolor("blue")
sky.hideturtle()
sky.goto(-315,100)
sky.pendown()
sky.begin_fill()
sky.goto(-315,400)
sky.goto(15,400)
sky.goto(15,100)
sky.goto(-315,100)
sky.end_fill()


def cloud(x,y):
    cloud = turtle.Turtle()
    cloud.hideturtle()
    cloud.speed(0)
    cloud.penup()
    cloud.pencolor("gray")
    cloud.goto(x,y)
    cloud.pendown()
    cloud.fillcolor("gray")
    cloud.begin_fill()
    cloud.left(55)
    for i in range(15):
        cloud.forward(2)
        cloud.right(4)
    cloud.left(80)
    for i in range(45):
        cloud.forward(2)
        cloud.right(4)
    cloud.goto(x,y)
    cloud.end_fill()

def tree(x,y):
    tree = turtle.Turtle()
    tree.speed(0)
    tree.penup()
    tree.hideturtle()
    tree.color("brown")
    tree.shape("rect")
    tree.goto(x-1,y)
    tree.pendown()
    tree.forward(2)
    tree.right(90)
    tree.forward(15)
    tree.right(90)
    tree.forward(2)
    tree.right(90)
    tree.forward(15)
    tree.right(90)

    tree.pencolor("green")
    tree.left(90)
    tree.goto(x,y)
    tree.shape("triangle")
    tree.pensize(50)
    tree.fillcolor("green")
    tree.stamp()

def mountain(x,y):
    mtn = turtle.Turtle()
    mtn.hideturtle()
    mtn.pencolor("white")
    mtn.speed(0)
    mtn.penup()
    mtn.goto(x,y)
    mtn.fillcolor("gray")
    mtn.begin_fill()
    mtn.pendown()
    mtn.goto(x-100,y)
    mtn.goto(x,y+150)
    mtn.goto(x+100,y)
    mtn.goto(x,y)
    mtn.end_fill()

mountain(-150,125)
mountain(-200,100)
mountain(-100,100)
for i in range(15):
    for j in range(10):
        tree(-300 + i*20 + j*11, 100-j*15)

for i in range(4):
    cloud(-300 + 75 * i , 300 + (i%2)*50 )

turtle.done()
```

Unit 3: Lists Solutions

- ## Lists, Mutability, and Indexes Solutions

Do It Yourself

```
fruits = ['apple', 'banana', 'orange']
print(fruits)

fruits[2] = 'grape'
print(fruits)
```

Three Favorite Foods

```
foods = ['hot dogs', 'minestrone', 'ice cream']
print(foods)
```

Multiple Types of Items in a List

```
varied_list = [3, 'cat', ['a, 'b']]
print(varied_list)
```

Index Ranges

Question: Given a list called l, with elements [5, 9, 4, 3, 7, 2, 6, 8], how would you return [4, 3, 7, 2]?

```
l[2:6]
```

No Start Index

Question: Given a list called l, with elements [5, 9, 4, 3, 7, 2, 6, 8], how would you return [5, 9, 4, 3, 7, 2, 6]?

```
l[:7]
```

No End Index

Question: Given a list called l, with elements [5, 9, 4, 3, 7, 2, 6, 8], how would you return [7, 2, 6, 8]?

```
l[4:]
```

No Start or End Index

Question: What is the output of the following code snippet? Try answering this question without using a code editor from above.

```
[2, 51, 42, 98, 34, 6, 47]
[23, 99, 42, 98, 34, 6, 47]
[23, 99, 42, 400, 34, 6, 47]
```

Middle to Zero

```
numbers[int(len(numbers)/2)] = 0
print(numbers)
```

Coordinate Practice

```
coordinates[0], coordinates[1] = -coordinates[0], -coordinates[1]
print(coordinates)
```

● Lists and Loops Solutions

Print the Squares

```
# Use i as an iterator
i = 0
while i < len(numbers):
    print(numbers[i] * numbers[i])
    i = i + 1
```

Prime Length

```
n = len(numbers)
print(n)
factor = 2
hasFactor = False

# For every number from 1 to n
while factor < n:
    # If the number is divisible by the factor, set hasFactor to True
    if n % factor == 0:
        hasFactor = True
    # Go to the next factor
    factor = factor + 1

if hasFactor:
    print("not prime")
else:
    print("prime")
```

- ## Item 'in' List Solutions

Primes or No Primes?

```
if 2 in numbers or 3 in numbers or 5 in numbers or 7 in numbers:
    print("has primes")
else:
    print("no primes")
```

Give Your Introductions!

```
if len(names) == 1:
    print("Introducing " + names[0] + "!")
if len(names) == 2:
    print("Introducing " + names[0] + " and " + names[1] + "!")
if len(names) == 3:
    print("Introducing " + names[0] + ", " + names[1] + ", and " +
names[2] + "!")
```

Matching Ends

```
print(numbers[0] == numbers[-1])
```

List Operations Solutions

How Many Apples and Oranges

Solutions located in chapter.

```
apples = fruits.count("apple")
oranges = fruits.count("orange")

if apples == 1 and oranges == 1:
    print("I have 1 apple and 1 orange")
elif apples == 1:
    print("I have 1 apple and " + str(oranges) + " oranges")
elif oranges == 1:
    print("I have " + str(apples) + " apples and 1 orange")
else:
    print("I have " + str(apples) + " apples and " + str(oranges) + "
oranges")

# Alternative
apples = fruits.count("apple")
oranges = fruits.count("orange")

output = "I have " + str(apples) + " apple"
if apples != 1:
    output = output + "s"
output = output + " and " + str(oranges) + " orange"
if oranges != 1:
    output = output + "s"
print(output)
```

Where's n?

```
numbers_copy = numbers[:]
while n in numbers_copy:
    index = numbers_copy.index(n)
    print(index)
    numbers_copy[index] += 1
```

Add Nothing (0) to the End

```
while len(numbers) < 10:
    numbers.append(0)
print(numbers)
```

Add Nothing (0) to the Beginning

```
while len(numbers) < 10:
    numbers.insert(0, 0)
print(numbers)
```

Append But Don't Extend

```
for number in b:
    a.append(number)
print(a)
```

Remove All of Nothing (0)

```
while 0 in numbers:
    numbers.remove(0)
print(numbers)
```

Unpad Right

```
while len(numbers) > 0 and numbers[-1] == 0:
    numbers.pop()
print(numbers)
```

Unpad Left

```
while len(numbers) > 0 and numbers[0] == 0:
    numbers.pop(0)
print(numbers)
```

Snack Time

```
snacks = fruit + chips + drinks
print(snacks)
```

- ## Reverse and Sort Solutions

Print Increasing Pairs

```
i = 0
```

```
while i < len(numbers) - 1:    # Can also use for i in
range(len(numbers) - 1)
    if numbers[i+1] > numbers[i]:
        print("{0} {1}".format(numbers[i], numbers[i+1]))
    i += 1
```

Alphabetized?

```
alph = True
for i in range(len(names) - 1):
    if names[i+1] < names[i]:
        alph = False
        break
if alph:
    print("alphabetized")
else:
    print("not alphabetized")
```

Bubble Sort Exercise

```
for i in range(len(numbers)-1,0,-1):
    for j in range(i):
        if numbers[j] > numbers[j+1]:
            numbers[j], numbers[j+1] = numbers[j+1] , numbers[j]
print(numbers)
```

Palindrome or Emordnilap?

```
copy = numbers[:]
copy.reverse()
palindrome = True
for i in range(len(numbers)):
    if numbers[i] != copy[i]:
        palindrome = False
        break
if palindrome:
    print("palindrome")
else:
    print("not palindrome")
```

Who's the Smallest?

```
# Students should not use min()
smallest = numbers[0]
for n in numbers:
    if n < smallest:
        smallest = n
print(smallest)
```

Print the Product

```
product = 1
for n in numbers:
    product *= n
print(product)
```

Print the Average

```
list_sum = 0
for n in numbers:
    list_sum += n
avg = list_sum / len(numbers)
print(avg)
```

Print the Variance

```
list_sum = 0
for x in numbers:
    list_sum += x
n = len(numbers)
avg = list_sum / n

variance_sum = 0
for x in numbers:
    variance_sum += (x - avg) ** 2
variance = variance_sum / n
print(variance)
```

Print the SD Difference

```
import math

def sd(numbers):
    list_sum = 0
    for x in numbers:
        list_sum += x
    n = len(numbers)
    avg = list_sum / n

    variance_sum = 0
    for x in numbers:
        variance_sum += (x - avg) ** 2
    variance = variance_sum / n
    return math.sqrt(variance)
    # Alternatively, return variance ** 0.5

print(sd(a) - sd(b))
```

Sublist Length

```
# Ending index for the sublist
index = 0

# While the sublist is all true values (nonzero numbers)
while index < len(numbers) and all(numbers[ : index + 1]):
    index = index + 1

# Print the index (also number of elements in sublist)
print(index)
```

Map Exercise

```
print(max(map(max, listOfLists)))
```

Review and Quiz Solutions

Question 1: Given the list `numbers = ["1", "2", "3", "4", "5"]`, which of the following are valid methods of accessing the element `"3"`? Mark all correct answers.

Answer:

```
numbers[-3]
numbers[2]
```

Question 2: Given a list of integers `numbers`, print out all elements in the list that are divisible by 4.

Answer:

```
numbers = [0,2,3,5,8,12]
for n in numbers:
    if n % 4 == 0:
        print(n)
```

Question 3: Given a list of integers `numbers`, print out the third-smallest element. You may assume the list contains at least three elements.

Answer:

```
numbers = [3,5,7,2,1]
numbers_copy = numbers[:]
numbers_copy.sort()
print(numbers_copy[2])
```

Question 4: Given a list of integers `numbers`, change the value of the middle element(s) to 0, then print the list.

Answer:

```
numbers = [1,3,4,5,9]
numbers[int(len(numbers) / 2)] = 0
if len(numbers) % 2 == 0:
    numbers[int(len(numbers) / 2) - 1] = 0
print(numbers)
```

Question 5: Print the number of double-digit numbers in a given list of integers `numbers`.

Answer:

```
numbers = [1,3,5,6,7,10,19]
count = 0
for n in numbers:
    if 10 <= n <= 99:
        count += 1
print(count)
```

Question 6: Given a list of integers `numbers`, print `sum could be 100` if removing any one element from `numbers` would result in the sum of the remaining numbers becoming 100. Print `sum could not be 100` otherwise.

Answer:

```
numbers = [85, 14, 8]
```

```
target = sum(numbers) - 100
if target in numbers:
    print("sum could be 100")
else:
    print("sum could not be 100")
```

Unit 4: Dictionaries Solutions

- ## Dictionaries Solutions

Language Dictionary

Students should make a dictionary of their own according to the example provided. Answers may vary.

```
{
    "student": "mwanafunzi",
    "teacher": "mwalimu",
    "computer": "kompyuta"
}
```

Do It Yourself

```
favorites = {}
favorites['fruit'] = 'apple'
favorites['season'] = 'winter'
favorites['color'] = 'purple'
print(favorites)
favorites['color'] = 'lime green'
print(favorites)
```

Technologies and Inventors

```
for invention in computer:
    print("The " + invention + " was invented by " +
computer[invention])
```

Factorials

```
def factorialDict(limit):
    n=1
    factorial = 1
    factorials = {}
    while factorial < limit:
        factorials[n] = factorial
        n=n+1
        factorial = factorial * n
    print(factorials)

factorialDict(200)
```

Print the Winner

```
highestScore = 0
winner = ""

for person in score:
    if score[person] > highestScore:
        winner = person
        highestScore = score[person]
print(winner)

# Another possible solution:
for person in score:
    if not winner or score[person] > score[winner]:
        winner = person
print(winner)
```

Opponents

```
names = matchup.keys() + matchup.values()
names.sort()
for name in names:
    print(name)

# Alternate solution:
for name in sorted(matchup.keys() + matchup.values()):
    print(name)
```

Average Score

```
scores = score.values()
print(float(sum(scores)) / len(scores))
```

- ## More About Dictionaries Solutions

DIY 1: Add a Key-Value Pair

```
roster['Jonathan'] = 8
```

DIY 2: Create A Dictionary

```
birth_months = {
    'Grant' : 'July',
    'Daniel' : 'June',
    'Jessica' : 'October',
    'Rita' : 'February'
}
target = input('Enter a name: ')
if target in birth_months:
    print("{0}'s birthday is in {1}".format(target,
birth_months[target]))
else:
    print("That name is not in the dictionary.")
```

DIY 3: Letter Frequency Table

```
phrase = input("Input a phrase: ")
frequency = {}  # this makes an empty dictionary
for letter in phrase:
    if letter in frequency:
        old_value = frequency.get(letter)
        frequency[letter] = old_value + 1
    else:
        frequency[letter] = 1

# add a line of code to print out the dictionary
print(frequency)
```

Dictionary in Action: Storing Computers in a Network

```
import turtle
locations = {}
locations['A'] = (-50, 50)
locations['B'] = (125, -75)
locations['C'] = (0, -50)
locations['D'] = (-150, -150)
locations['E'] = (125, 150)
# add one more
locations['F'] = (50, 50)

print(locations)
```

```
screen = turtle.Screen()
screen.bgcolor("yellow")
stevie = turtle.Turtle()
stevie.penup()

for letter in locations.keys():
    stevie.goto(locations.get(letter))
    stevie.dot(25, "red")
    stevie.write(letter, align = 'center', font = ('Arial', 24, 'bold'))
```

Dictionary in Action: Drawing Paths in a Network

```
# Answers may vary, but the argument to the first speed() call should be
# less than the argument to the second speed() call
# The argument for speed() should be an integer from 0 to 10
# 0 is instant, 1 is slowest, 10 is fastest
import turtle
locations = {}
locations['A'] = (-50, 50)
locations['B'] = (125, -75)
locations['C'] = (0, -50)
locations['D'] = (-150, -150)
locations['E'] = (125, 150)
print(locations)

screen = turtle.Screen()
screen.bgcolor("yellow")
stevie = turtle.Turtle()
stevie.speed(3)
stevie.penup()

for letter in locations.keys():
    stevie.goto(locations.get(letter))
    stevie.dot(25, "red")
    stevie.write(letter, align = 'center', font = ('Arial', 24, 'bold'))

paths = {}
paths['A'] = ['D','C']
paths['D'] = ['B']
paths['C'] = ['B', 'E']
print(paths)

stevie.speed(9)
for start in paths.keys():
    start_location = locations.get(start)
    neighbors = paths.get(start)
    for finish in neighbors:
        finish_location = locations.get(finish)
        stevie.goto(start_location)
        stevie.pendown()
        stevie.goto(finish_location)
        stevie.penup()
```

- ## Review and Quiz Solutions

Question 1: Given a list of what a group of people `ordered` and the `price` of each item as a dictionary, print the group's total bill.

Answer:

```
total = 0
for item in ordered:
    total += price[item]
print(total)
```

Question 2: Create a dictionary `letters` where the keys are all upper and lower case letters, and values are their ASCII values. Print the `letters` dictionary after you create it.

Answer:

```
letters = {}
for char in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
    letters[char] = ord(char)
print(letters)
```

Question 3: Given a letter, use the `ord()` function to print its ASCII value. Given an ASCII value, use `chr()` to convert it to a character and print the character.

Answer:

```
print(ord(letter))
print(chr(value))
```

Question 4: Given a list of student's `'homework'`, `'quiz'`, and `'test'` scores in the given `grade` dictionary as percentages, print the student's weighted average where 40% is their test score, 35% is their quiz score, and 25% is their homework score.

Answer:

```
print(grade['test']*0.4 + grade['quiz']*0.35 + grade['homework']*0.25)
```

- Tetris Solution

```
import turtle, random

screen = turtle.Screen()
screen.bgcolor("black")
screen.delay(0)

sprite = turtle.Turtle()
sprite.penup()
sprite.speed(0)
sprite.color('white')
sprite.ht()

#Shapes Informations
shapes = {}
blocks = {}
blocks['j'] = ("blue", [(0,0),(-20,-20), (-20,0), (20,0)])
blocks['o'] = ("red", [(0,0),(0,-20), (-20,-20), (-20,0)])
blocks['z'] = ("green", [(0,0),(0,-20), (-20,0), (-20,20)])
blocks['l'] = ("yellow", [(0,0),(20,0), (-20,0), (-20,20)])
blocks['i'] = ("brown", [(0,0),(-20,0), (20,0),(40,0)])
blocks['s'] = ("violet", [(0,0),(0,20), (-20,0), (-20,-20)])
blocks['t'] = ("purple", [(0,0),(20,0), (0,-20), (0,20)])

shapes['p'] = ((-10,-10), (-10,10), (10,10), (10,-10))
shapes['j'] = ((-30,-30), (-30,10), (30,10), (30,-10),
(-10,-10),(-10,-30))
shapes['o'] = ((-30,-30), (-30,10), (10,10), (10,-30))
shapes['z'] = ((-30,-10), (-30,30), (-10,30), (-10,10),
(10,10),(10,-30),(-10,-30),(-10,-10))
shapes['l'] = ((-30,-10), (-30,30), (-10,30), (-10,10),
(30,10),(30,-10))
shapes['i'] = ((-30,-10), (-30,10), (50,10),(50,-10))
shapes['s'] = ((-30,-30), (-30,10), (-10,10), (-10,30),
(10,30),(10,-10),(-10,-10),(-10,-30))
```

```
shapes['t'] = ((-10,-30), (-10,30), (10,30), (10,10),
(30,10),(30,-10),(10,-10),(10,-30))

for i,j in shapes.items():
    screen.register_shape(i,j)

sprite.shape('p')

#Game Set Up
block = sprite.clone()
block.st()
letter = ""
shape = ""

Left = -100
Right = 80
Bottom = -160
Top = 140
size = 20

pixels = {}

for i in range(Bottom,Top+size,size):
    pixels[i] = {}
    for j in range(Left,Right+size,size):
        pixels[i][j] = sprite.clone()
        pixels[i][j].goto(j,i)
        pixels[i][j].stamp()

#Game Loop
def reset():
    global letter, shape
    letter = "jozlist"[random.randint(0,6)]
    shape_info = blocks[letter]
    block.seth(0)
    block.goto(0,Top)
    block.shape(letter)
    block.color(shape_info[0])
    shape = shape_info[1][:]
    drop()

def drop():
    if(check_bound(0,-size,shape)):
        block.goto(block.xcor(),block.ycor()-size)
        screen.ontimer(drop, 800)
    else:
        if(block.ycor() != Top):
            add_to_bound()
            check_row()
            reset()

#Bounding Area
bound = {}
def check_bound(dx,dy,offsets):
    x = block.xcor()
    y = block.ycor()
    for i in offsets:
        ty = y+i[0]+dy
        tx = x+i[1]+dx
        if(ty < Bottom or tx < Left or tx > Right or (ty in bound.keys()
and tx in bound[ty])):
            return False

    return True

def add_to_bound():
    x = block.xcor()
    y = block.ycor()
    for i in shape:
        ty = y+i[0]
        tx = x+i[1]
        if(ty not in bound.keys()):
            bound[ty] = set()

        bound[ty].add(tx)
        pixel = pixels[ty][tx]
        pixel.color(block.fillcolor())
        pixel.st()

#Keyboard Controls
def move_by(dx,dy):
    if(check_bound(dx,dy,shape)):
        block.goto(block.xcor()+dx,block.ycor()+dy)

def left():
    move_by(-size,0)

def right():
    move_by(size,0)

def down():
    move_by(0,-size)

def rotate_left():
    global shape
    t = []
    for i in shape:
        t.append((i[1],-i[0]))

    if(check_bound(0,0,t)):
        shape = t
        block.left(90)

def rotate_right():
    global shape
    t = []
    for i in shape:
        t.append((-i[1],i[0]))

    if(check_bound(0,0,t)):
        shape = t
        block.right(90)

screen.onkey(left, "Left")
screen.onkey(right, "Right")
screen.onkey(down, "Down")
screen.onkey(rotate_left, "A")
screen.onkey(rotate_right, "D")
screen.listen()

#Collapsing Rows
def check_row():
    full = (Right-Left+size)/size
    up=0
    t = sorted(bound.keys())
    for i in t:
        while(i+up in bound.keys() and len(bound[i+up]) == full):
            up+=size

        if(up>0):
            if(i+up in bound.keys()):
                bound[i] = set(bound[i+up])
                for k in pixels[i].keys():
                    if(pixels[i+up][k].isvisible()):
                        pixels[i][k].color(pixels[i+up][k].fillcolor())
                        pixels[i][k].st()
                    else:
                        pixels[i][k].ht()
            else:
                del bound[i]
                for v in pixels[i].values():
                    v.ht()

reset()
```

- Tetris Follow Up Solution

```
import turtle, random

screen = turtle.Screen()
screen.bgcolor("black")
screen.delay(0)

sprite = turtle.Turtle()
sprite.penup()
sprite.speed(0)
sprite.color('white')
sprite.ht()

#Shapes Informations
shapes = {}
blocks = {}
blocks['j'] = ("blue", [(0,0),(-20,-20), (-20,0), (20,0)])
blocks['o'] = ("red", [(0,0),(0,-20), (-20,-20), (-20,0)])
blocks['z'] = ("green", [(0,0),(0,-20), (-20,0), (-20,20)])
blocks['l'] = ("yellow", [(0,0),(20,0), (-20,0), (-20,20)])
blocks['i'] = ("brown", [(0,0),(-20,0), (20,0),(40,0)])
blocks['s'] = ("violet", [(0,0),(0,20), (-20,0), (-20,-20)])
blocks['t'] = ("purple", [(0,0),(20,0), (0,-20), (0,20)])

shapes['p'] = ((-10,-10), (-10,10), (10,10), (10,-10))
shapes['j'] = ((-30,-30), (-30,10), (30,10), (30,-10),
(-10,-10),(-10,-30))
shapes['o'] = ((-30,-30), (-30,10), (10,10), (10,-30))
shapes['z'] = ((-30,-10), (-30,30), (-10,30), (-10,10),
(10,10),(10,-30),(-10,-30),(-10,-10))
shapes['l'] = ((-30,-10), (-30,30), (-10,30), (-10,10),
(30,10),(30,-10))
shapes['i'] = ((-30,-10), (-30,10), (50,10),(50,-10))
shapes['s'] = ((-30,-30), (-30,10), (-10,10), (-10,30),
(10,30),(10,-10),(-10,-10),(-10,-30))
shapes['t'] = ((-10,-30), (-10,30), (10,30), (10,10),
(30,10),(30,-10),(10,-10),(10,-30))

for i,j in shapes.items():
    screen.register_shape(i,j)

sprite.shape('p')

#Game Set Up
block = sprite.clone()
```

```
block.st()
letter = ""
shape = ""

Left = -100
Right = 80
Bottom = -160
Top = 140
size = 20

pixels = {}

for i in range(Bottom,Top+size,size):
    pixels[i] = {}
    for j in range(Left,Right+size,size):
        pixels[i][j] = sprite.clone()
        pixels[i][j].goto(j,i)
        pixels[i][j].stamp()

scoreTurtle = sprite.clone()
scoreTurtle.goto(110, 110)
scoreTurtle.color("green")
scoreTurtle.hideturtle()
score = 0
scoreTurtle.write(str(score))

def update_score(amount):
    global score
    score += amount
    scoreTurtle.clear()
    # clear previous writing
    scoreTurtle.write("score : " + str(score))

#Game Loop
def reset():
    global letter, shape
    letter = "jozlist"[random.randint(0,6)]
    shape_info = blocks[letter]
    block.seth(0)
    block.goto(0,Top)
    block.shape(letter)
    block.color(shape_info[0])
    shape = shape_info[1][:]
    drop()

def drop():
    update_score(1)
    if(check_bound(0,-size,shape)):
        block.goto(block.xcor(),block.ycor()-size)
        screen.ontimer(drop, 800)
    else:
        if(block.ycor() != Top):
            update_score(10)
            add_to_bound()
            check_row()
            reset()

#Bounding Area
bound = {}
def check_bound(dx,dy,offsets):
    x = block.xcor()
    y = block.ycor()
    for i in offsets:
        ty = y+i[0]+dy
        tx = x+i[1]+dx
        if(ty < Bottom or tx < Left or tx > Right or (ty in bound.keys()
and tx in bound[ty])):
            return False

    return True

def add_to_bound():
    x = block.xcor()
    y = block.ycor()
    for i in shape:
        ty = y+i[0]
        tx = x+i[1]
        if(ty not in bound.keys()):
            bound[ty] = set()

        bound[ty].add(tx)
        pixel = pixels[ty][tx]
        pixel.color(block.fillcolor())
        pixel.st()

#Keyboard Controls
def move_by(dx,dy):
    if(check_bound(dx,dy,shape)):
        block.goto(block.xcor()+dx,block.ycor()+dy)

def left():
    move_by(-size,0)

def right():
    move_by(size,0)
```

```
def down():
    move_by(0,-size)
    update_score(1)

def hard_down():
    screen.onkey(None, "Space")
    while (check_bound(0, -size, shape)):
        update_score(2)
        block.goto(block.xcor() + 0, block.ycor() - size)
    screen.onkey(hard_down, "Space")


def rotate_left():
    global shape
    t = []
    for i in shape:
        t.append((i[1],-i[0]))

    if(check_bound(0,0,t)):
        shape = t
        block.left(90)

def rotate_right():
    global shape
    t = []
    for i in shape:
        t.append((-i[1],i[0]))

    if(check_bound(0,0,t)):
        shape = t
        block.right(90)

screen.onkey(left, "Left")
screen.onkey(right, "Right")
screen.onkey(down, "Down")
screen.onkey(rotate_left, "A")
screen.onkey(rotate_right, "D")
screen.onkey(hard_down, 'Space')
screen.listen()

#Collapsing Rows
def check_row():
    full = (Right-Left+size)/size
    up=0
    t = sorted(bound.keys())
    for i in t:
        while(i+up in bound.keys() and len(bound[i+up]) == full):
            up+=size
        if(up>0):
            if(i+up in bound.keys()):
                bound[i] = set(bound[i+up])
                for k in pixels[i].keys():
                    if(pixels[i+up][k].isvisible()):
                        pixels[i][k].color(pixels[i+up][k].fillcolor())
                        pixels[i][k].st()
                    else:
                        pixels[i][k].ht()
            else:
                del bound[i]
                update_score(100)
                for v in pixels[i].values():
                    v.ht()

reset()
```

Unit 5: PyGal and Charting

• Get Started with Data Solutions

How would you go about finding the "most common" price for hamburgers, given this data? Brainstorm some ideas in the text box below.

Answers may vary. Potential answers: calculate the average price, find the price that occurs the most often.

• CSV Files Solutions

All coding questions in this chapter build off of the previous questions. For each editor, the contents of the previous editor are copied over as starting code. For brevity, only the new code written in each module is included below.

Processing One Line of the File

```
import csv

file = open("electricity_report.csv")
csv_reader = csv.reader(file)
electricity_list = list(csv_reader)

# your code here
def sum_one_hour(single_row):
    sum = 0
    keep = single_row[2:]
    print(keep)
    for num in keep:
        sum = sum + int(num)
    return sum

print(sum_one_hour(electricity_list[1]))
```

Processing One Column

```
# Sum wind energy column
wind_energy = [row[2] for row in electricity_list]
print(wind_energy)
```

Summing a Column

```
# remove the first value, then sum up the remaining
def sum_one_column(single_col):
    sum = 0
    # Write your code below
    keep = single_col[1:]
    for num in keep:
        sum += int(num)
    return sum
print(sum_one_column(wind_energy))
```

1. Write code that computes the total amount of electricity produced by coal during the timeframe specified in the CSV file. Remember that you must first open the file, make a CSV reader, and use the reader to make a list of lists before you can manipulate the data.

Answer:

```
import csv

file = open("electricity_report.csv")
csv_reader = csv.reader(file)
electricity_list = list(csv_reader)

coal_energy = [int(row[-2]) for row in electricity_list[1:]]
print(sum(coal_energy))    # Should print 14355169
```

2. Write code that sums up all the rows of megawatt-hour data, which would show the total number of megawatts-hours of electricity generated in the 48 contiguous US states by all listed energy sources.

Answer:

```
import csv

file = open("electricity_report.csv")
csv_reader = csv.reader(file)
electricity_list = list(csv_reader)

total_energy = [row[2:] for row in electricity_list[1:]]
# Convert all entries to integers
total_energy = [[int(col) for col in row] for row in total_energy]
print(sum([sum(row) for row in total_energy]))    # Should print
74132236
```

3. Explain how the program you wrote in this lesson allowed you to gain insight and knowledge from the Electricity data set.

Answers may vary. Sample answer: the program allowed us to calculate and compare the total amounts of energy generated by different sources.

- ## Making Plots Solutions

Making a Pie Chart

```
import pygal
pie_chart = pygal.Pie()
pie_chart.title = 'Students in CS Principles 4th hour'
pie_chart.add('freshmen', 5)
pie_chart.add('sophomores', 12)
pie_chart.add('juniors', 16)
pie_chart.add('seniors', 3)
pie_chart.render()
```

Making a Bar Chart

```
import pygal
bar_chart = pygal.Bar()
bar_chart.title = 'Number of AP exams taken by year'
bar_chart.add('AP CSP', [44330, 72187, 96105, 102500])
bar_chart.add('AP CSA', [60519, 65133, 69685, 72659])
bar_chart.x_labels = map(str, range(2017, 2021))
bar_chart.render()
```

Making a Pie Chart for the Energy Data

```
import csv, pygal

file = open("electricity_report.csv")
csv_reader = csv.reader(file)
electricity_list = list(csv_reader)

row15 = electricity_list[15]  # entire row 15
megaWatt_list = row15[2:]   # numbers in row 15 without the first 2
values
header = electricity_list[0]
energy_types = header[2:]
pie_chart = pygal.Pie()
pie_chart.title = 'Energy Generated in U.S. Lower 48 by Source'
for i in range(0,len(energy_types)):
    pie_chart.add(energy_types[i], int(megaWatt_list[i]))
pie_chart.render()
```

Making a Plot for Solar Data

```
import csv, pygal

file = open("electricity_report.csv")
csv_reader = csv.reader(file)
electricity_list = list(csv_reader)

solar_energy = [row[3] for row in electricity_list]
solar_energy = solar_energy[1:]
line_chart = pygal.Line(show_dots=False, stroke_style={'width':5})
line_chart.title = 'Solar Energy Generated in U.S. Lower 48'
line_chart.x_title = 'Seven day period'
line_chart.y_title = 'MegaWatts per hour'
line_chart.add(None, solar_energy)
line_chart.render()
```

On Your Own

Answers may vary. This editor provides a "sandbox" for students to create any graphs they want to make.

- ## Thinking About Data Sets Solutions

The School District

Question: Which of the following questions could be answered using only the data in the dataset?
Answer: Are students who live in a certain area of the city more absent than the average student?

The Podcast Service

Question: Which of the following things could NOT be determined from the data?
Answer: How long does an average download take?

The Biologist

Question: Which of the following things could NOT be determined from the data?
Answer: How many coyotes travel together?

The University

Question: What research question could NOT be investigated from this data?
Answer: Do students use the system more on smartphones than on laptops?

The Soccer Clubs

Question: Two different soccer clubs maintain information about their players. The clubs are about to merge and they want to form a single data set. What will be the biggest difficulty in merging the two data files?
Answer: The data files may store different categories of information.

The E-Bike Company

Question: Think of 3 different interesting things you could find out from the data. Then, share your ideas with a partner or with the class.
Suggested Answers:
Which store has the highest average prices?
What color bike does the company have the most of in stock?
What is the average size of a bike at each location?

Your Own Question

Answers may vary.

- Review and Quiz Solutions

Question 1: If a CSV file is stored as a list of rows called data_rows, then which expression represents an entire row?

Answer:

```
data_rows[3]
```

Question 2: If you wanted to keep all but the first 3 things in a list called one_row, what code would you use?

Answer:

```
keep = one_row[3:]
```

Question 3: What module do you need to import in order to make plots?

Answer: `pygal`

Question 4: What information cannot be determined from this data?

Answer: **The leading scorer**

Question 5a:. Write code that stores all of the information about Maria into row_list.

```
row_list = results_list[3]
```

Question 5b: Write code that prints out 'Maria wore bib number 2252 and finished in place 3 out of 265 runners.

Answer:

```
print(str(row_list[2]) + ' wore bib number ' + str(row_list[1]) + ' and
finished in place ' + str(row_list[0])+ ' out of ' +
str(len(results_list)-1) + ' runners.')
# Better solution:
```

```
print("{0} wore bib number {1} and finished in place {2} out of {3}
runners.".format(row_list[2], row_list[1], row_list[0],
len(results_list)-1))
```

Question 5c: Write code that collects all the data from the age column and places them into a list called age_list.

Answer:

```
age_list = []
for row in results_list:
    age_list.append(row[3])
# Better solution:
age_list = [row[3] for row in results_list]
```

Question 5d: Write code that removes the first element from age_list but keeps the rest of the data.

Answer:

```
age_list = age_list[1:]
```

- ## Data Project: Baseball Stats Solutions

```
import statistics, pygal, csv

file = open("batting-averages.csv")
csv_reader = csv.reader(file)
avg = list(csv_reader)

avg = [float(row[1]) for row in avg[1:]]
print("The mean AVG is {0}".format(statistics.mean(avg)))
print("The mode AVG is {0}".format(statistics.mode(avg)))
print("The median AVG is {0}".format(statistics.median(avg)))
print("The maximum AVG is {0}".format(max(avg)))
print("The minimum AVG is {0}".format(min(avg)))
print("The range is {0}".format(max(avg) - min(avg)))
```

Minimum: 0.108
Maximum: 0.374
Range: 0.266
Mean: 0.2426

Median: 0.2445
Mode: 0.25

Unit 6: Classes Solutions

- ## Classes and Objects Solutions

Create Objects

```
# Write your Restaurant class here
class Restaurant:
    def createPizza(self):
        print("Making pizza ... done")
    def deliverPizza(self):
        print("Delivering pizza ... done")

# Create a Restaurant object and call both methods
restaurant = Restaurant()
restaurant.createPizza()
restaurant.deliverPizza()
```

- ## Attributes Solutions

Methods as Attributes

Student answers may vary. Students should declare a new Human object with their name and age in the same way as the examples in the coding exercise.

The getattr Function

Exercise 1

```
class Human:
    def introduce(self):
        print("My name is " + self.name)
        print("I am " + str(self.age) + " years old")

unknown = Human()
unknown.name = "Jane"
unknown.age = 13
unknown.introduce()
```

Favorite Food

```
class Human:
    # Write your constructor here
    def __init__(self, name, age, food):
        self.name = name
        self.age = age
        self.food = food

    def introduce(self):
        print("My name is " + self.name + " and I am " + str(self.age) +
" years old. My favorite food is " + self.food + ".")

jane = Human("Jane", 13, "pizza")
jane.introduce()
```

Vehicle Registration

```
class Vehicle:
    def __init__(self, make, wheels):
        self.make = make
        self.wheels = wheels
        if (wheels == 2):
            self.type = 'Motorcycle'
        elif (wheels == 4):
            self.type = 'Car'
        else:
            self.type = 'Truck'

    def describe(self):
        print(self.make + ', ' + self.type)

vehicle_one = Vehicle('Corolla', 4)
vehicle_one.describe()

vehicle_one = Vehicle('Scrambler', 2)
vehicle_one.describe()

vehicle_one = Vehicle('Peterbilt', 18)
vehicle_one.describe()
```

Using Multiple Classes

```
class Card:
    def __init__(self, number, suit):
        self.number = number
        self.suit = suit
```

```
class Deck:
    def __init__(self, numberOfSuits, cardsPerSuit):
        self.deck = []

        for suit in range(numberOfSuits):
            for number in range(1, cardsPerSuit + 1):
                self.deck.append(Card(number, suit))

    def printDeck(self):
        for card in self.deck:
            suit = ''
            if (card.suit == 0):
                suit = 'Hearts'
            elif (card.suit == 1):
                suit = 'Diamonds'
            elif (card.suit == 2):
                suit = 'Clubs'
            elif (card.suit == 3):
                suit = 'Spades'
            print(str(card.number) + " of " + suit)

cards = Deck(4, 13)
cards.printDeck()
```

- ## Inheritance Solutions

Class Inheritance

Question 1: What is the superclass of Reptile?
Answer: Animal

Question 2: What are the subclasses of Mammal?
Answer: Dog and Cat

Question 3: How many classes in the diagram inherit from Animal?
Answer: 5

Create a Minivan

```
# Write your Minivan class here
class Minivan(Car):
    def __init__(self, maxPassengers=7):
        super().__init__(maxPassengers)

    def pickUp(self):
        if self.maxPassengers - self.currentPassengers < 3:
            print("The car is full!")
        else:
            self.currentPassengers += 3
            print("Picked up three passengers. There are " +
str(self.currentPassengers) + " people in the car.")

    def dropOff(self):
        if self.currentPassengers == 0:
            print("There is no one to drop off!")
        else:
            self.currentPassengers -= 2
            print("Dropped off two passengers. There are " +
str(self.currentPassengers) + " people in the car.")

    def unload(self):
        self.currentPassengers = 0
        print("Dropped off all passengers. There are no people in the
car.")

minivan = Minivan()
minivan.pickUp()
minivan.pickUp()
minivan.dropOff()
minivan.dropOff()
minivan.unload()
```

- ## Review and Quiz Solutions

Question 1: What will the following code print?
Answer: 10, 5, 10

Question 2: For a Fruit class, what the name of the class method that is called when a new Fruit object is created?
Answer:

```
__init__()
```

Question 3: What will the following code print?
Answer:
red
yellow
green

Question 4: What is the difference between a function and a method?
Answer: Methods are functions that belong to an object.

Question 5: Define a Library class that has:

- a constructor that takes a list of strings books that contains all books owned by the library. It should create a dictionary attribute bookStatus, where bookStatus[book] returns the name of the patron who checked out the book. If the book is not checked out, bookStatus[book] returns the string "At library".
- a method checkStatus(book) which uses bookStatus to print the current status of book.
- a method borrow(book, patron) that checks a book out to a patron by updating the book's value in bookStatus to be patron, if the book is currently at the library.
- a method return(book) that resets the book's value in bookStatus to the string "At library".
- a method add(book) that adds a new book to bookStatus. New books are stored at the library.

Answer:

```
# Write your Library class here.
class Library:
    def __init__(self, books):
        self.bookStatus = {}
        for book in books:
            self.bookStatus[book] = "At library"

    def checkStatus(self, book):
        print(self.bookStatus[book])

    def borrow(self, book, patron):
        self.bookStatus[book] = patron

    def returnBook(self, book):
        self.bookStatus[book] = "At library"

    def add(self, book):
        self.bookStatus[book] = "At library"
```

- ## Project Solution

```
class Supers:
    def __init__(self, super_name, secret_id, age, affiliations = ""):
        self.my_name = super_name
        self.affiliations = affiliations
        self.secret_id = secret_id
        self.powers = []
        self.weaknesses = []
        self.age = age
        self.powers_active = True
    def addPowers(self, power):
        if (type(power) == list):
            for p in power:
                self.powers.append(p)
        elif (type(power) == str):
            self.powers.append(power)

    def addWeakness(self, weakness):
        self.weaknesses.append(weakness);

    def makeInvincible(self, power):
        self.strength = float('inf')
        self.speed = float('inf')
        self.healing_factor = float('inf')
    def introduce(self):
        intro = "My name is " + str(self.my_name) + " (although my
friends call me " + str(self.secret_id) + "). "
        if (len(self.powers) > 0 and self.powers_active):
            intro += "My powers are: "
            for i in range(0, len(self.powers)):
                if (i != len(self.powers)-1):
                    intro+= self.powers[i] + ", "
                else:
                    intro+= "and " + self.powers[i] + "."
        else:
            intro += "I dont have any powers, yet..."
        if (self.affiliations != ""):
            intro += " I work with the " + str(self.affiliations) + "."
        print(intro)

class Mutant(Supers):
    def __init__(self, super_name, secret_id, age, affiliations = ""):
        super().__init__(super_name, secret_id, age, affiliations)
        self.species = "Human"
        self.powers_active = False
    def activateMutantGene(self):
        self.powers = true

class Kryptonian(Supers):
    def __init__(self, super_name, secret_id, age, affiliations = ""):
        super().__init__(super_name, secret_id, age, affiliations)
        self.species = "Kryptonian"
    def nearKryptonite(is_near):
        if (is_near):
            print("Need to get out of here!")
        else:
            print("Whew! Good for now")
```

```
class Amazon(Supers):
    def __init__(self, super_name, secret_id, age, affiliations = ""):
        super().__init__(super_name, secret_id, age, affiliations)
        self.species = "Amazonian"

Superman = Kryptonian("Superman", "Clark Kent", 30, "Justice League")
Superman.addPowers(["super strength", "flight", "laser eyes"])
Superman.strength = float('inf')

Batman = Supers("Batman", "Bruce Wayne", 35, "Justice League")
Batman.intellignce = 5000
Batman.strength = 1000

WonderWoman = Amazon("Wonder Woman", "Diana Prince", 5000, "Justice
League")
WonderWoman.addPowers(["super strength", "flight", "enhanced senses",
"resistance to sorcery"])
WonderWoman.strength = 100000000

Justice_League = [Superman, Batman, WonderWoman]

Wolverine = Mutant("Wolverine", "Logan", 189, "X-Men")

Superman.introduce()
WonderWoman.introduce()
Batman.introduce()
Wolverine.introduce()
```

Unit 7: Recursion Solutions

- ### Merge Sort Solutions

Implementing Merge Sort

```
def merge(elements, startIndex, midIndex, endIndex):
    left = startIndex;
    right = midIndex;
    temp = [0] * (endIndex - startIndex)
    t = 0

    # Check the first element of each list to see which is smaller
    # and update the temporary list
    while (left < midIndex and right < endIndex):
        if (elements[left] < elements[right]):
            temp[t] = elements[left]
            left += 1
        else:
            temp[t] = elements[right]
            right += 1
        t += 1

    # If elements remain in left list, fill in the rest of the original
list with the elements
    if (left < midIndex):
        temp[t:] = elements[left:midIndex]

    # If elements remain in right list, fill in the rest of the original
list with the elements
    if (right < endIndex):
        temp[t:] = elements[right:endIndex]

    elements[startIndex:endIndex] = temp

def mergeSort(elements, startIndex, endIndex):
    # Implement merge sort here
    if endIndex - startIndex < 2:
        return

    midIndex = int((endIndex+startIndex)/2)
    mergeSort(elements, startIndex, midIndex)
    mergeSort(elements, midIndex, endIndex)
    merge(elements, startIndex, midIndex, endIndex)

# Main program
values = [7, 3, 2, 9, 10, 11, 4, 6]

# Call mergesort() on values
mergeSort(values, 0, len(values))

print(values)
```

- ## Fractals Solutions

Coding the Sides

```
####################################
##  Exercise 1: Coding the Sides  ##
####################################

import turtle
```

```
t = turtle.Turtle()
t.penup()
t.goto(-150, 0)
t.pendown()

def draw_side(side_length, level):
    global recursion_depth
    print ((recursion_depth-level) * "  " + "starting snowflake with 
side_length: " + str(side_length) + " and level: " + str(level))
    if level == 0:
        t.fd(side_length)
    else:
        draw_side(side_length/3, level-1)
        t.right(60)
        draw_side(side_length/3, level-1)
        t.left(120)
        draw_side(side_length/3, level-1)
        t.right(60)
        draw_side(side_length/3, level-1)


recursion_depth = 2

draw_side(300, recursion_depth)
```

Making the Snowflake

```
########################################
##  Exercise 2: Making the Snowflake  ##
########################################

import turtle

t = turtle.Turtle()
t.penup()
t.goto(-150, 0)
t.pendown()

def draw_side(side_length, level):
    global recursion_depth
    print ((recursion_depth-level) * "  " + "starting snowflake with 
side_length: " + str(side_length) + " and level: " + str(level))
    if level == 0:
        t.fd(side_length)
    else:
        draw_side(side_length/3, level-1)
        t.right(60)
        draw_side(side_length/3, level-1)
        t.left(120)
        draw_side(side_length/3, level-1)
        t.right(60)
        draw_side(side_length/3, level-1)

def draw_snowflake(side_length, total_levels):
    draw_side(side_length, recursion_depth)
    t.left(120)
    draw_side(side_length, recursion_depth)
    t.left(120)
    draw_side(side_length, recursion_depth)

recursion_depth = 2

draw_snowflake(100, recursion_depth)
```

- ## Review and Quiz Solutions

Question 1: What will the following code print?
Answer: 10

Question 2: What will be the size of the stack when the base case is reached?
Answer: 3

Question 3: Let's look at another sequence similar to the fibonacci code. What will be printed?
Answer: 5

Question 4: Take a look at another implementation of coin_combination. What will be printed?
Answer: 3, 3, 4, 4

Question 5: What is the purpose of the helper function merge for mergeSort?
Answer: It combines 2 sorted lists together into a sorted list.

Question 6: How are fractals related to recursion?
Answer: All of the above

- ## Unit 7 Project Solution

```
import turtle

t = turtle.Turtle()
```

```
t.pencolor("blue")
t.speed(0)
t.penup()
t.goto(-50,100)
t.pendown()

def dragon(level, rule):

    if level <= 0:
        t.forward(75/levels)
        return

    if rule == 0:
        dragon(level-1, 0)
        t.right(90)
        dragon(level-1, 1)

    elif rule == 1:
        dragon(level-1, 0)
        t.left(90)
        dragon(level-1, 1)


levels = 10
dragon(levels, 0)

turtle.done()
```