Starting Out with Python, 4th Edition, Global Edition

Answers to Review Questions

Chapter 1

Multiple Choice

1. b
2. a
3. d
4. b
5. c
6. a
7. c
8. b
9. a
10. a
11. d
12. b
13. c
14. b
15. c
16. a
17. b
18. d
19. b
20. b
21 c
22. a
23. d
24. a
25. b

True or False

1. False
2. True
3. True
4. False
5. True
6. False
7. True
8. False
9. False
10. False

Short Answer

1. A computer's main memory (random access memory) temporarily stores programs and data that are currently in use, while the computer's secondary storage retains programs and data in an ongoing manner, for long periods of time.

2. A bit that is turned on represents 1, and a bit that is turned off represents 0.

3. 128 different character codes. The Unicode character set addresses this limitation, as it is able to represent many more characters.

4. A statement

5. Mnemonics

6. A compiler is a program that translates a high-level language program into a separate machine language program. The machine language program can then be executed any time it is needed. An interpreter is a program that both translates and executes the instructions in a high-level language program. As the interpreter reads each individual instruction in the program, it converts it to a machine language instruction and then immediately executes it. Because interpreters combine translation and execution, they typically do not create separate machine language programs.

7. Operating system

Exercises

1. *No solution -- This is a hands-on exercise to help you learn how to work with the Python interpreter in interactive mode.*

2. *No solution -- This is a hands-on exercise to help you learn how to work with the IDLE programming environment.*

3.

Decimal	Binary
14	1110
87	01010111
128	10000000
170	10101010

4.

Binary	Decimal
101	5
1111	15
110010	50

5. The first printable character (a space) has a code of 32, the “A” character has a code of 65, and the “a” character has a code of 97.

6.

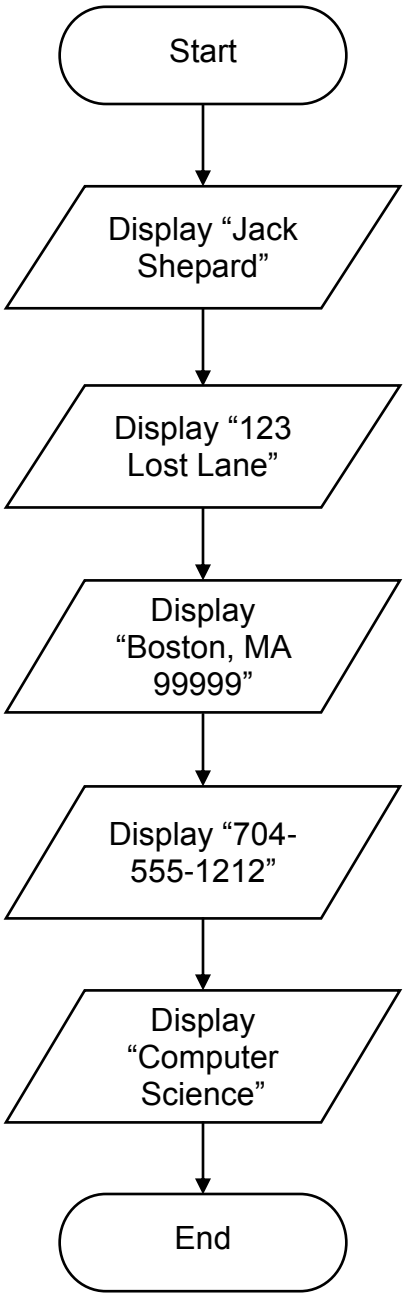
- Guido van Rossum is the creator of the Python programming language, and his title of “Benevolent Dictator for Life” (BFDL) means that despite Python being a community-developed project, he has final say in any issues that the community cannot reach an agreement on.
- A collection of aphorisms that summarise the core guiding principles and design goals of Python. One can print them in IDLE by typing “`import this`”.
- The first version of Python 3 (version 3.0) was released in 2008, and the final version of Python 2 (version 2.7) was released in 2010.]

CHAPTER 2 TO 13

Programming Exrecise answers samples

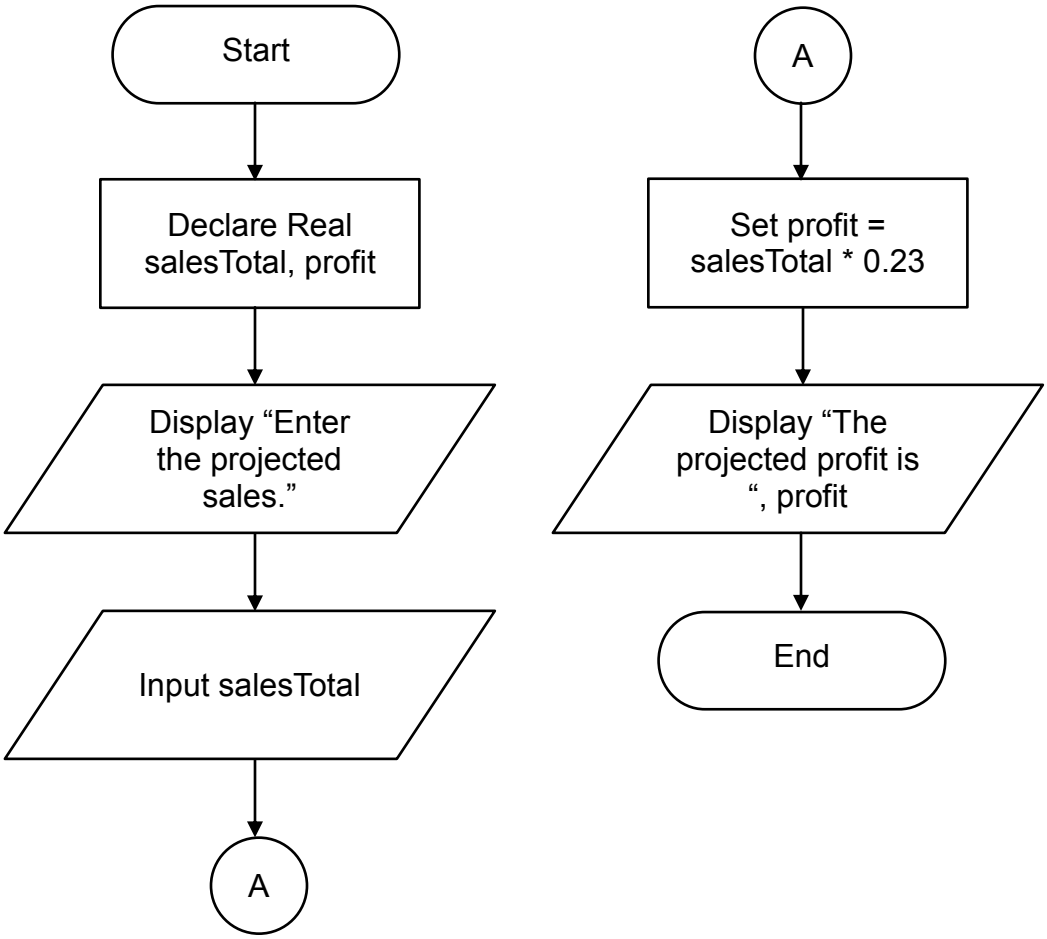
Programming Exercise 2-1

```
print ("Jack Shepard")
print ("123 Lost Lane")
print ("Boston, MA  99999")
print ("704-555-1212")
print ("Computer Science")
```



Programming Exercise 2-2

```
# Variables to hold the sales total and the profit
salesTotal = 0.0
profit = 0.0

# Get the amount of projected sales.
salesTotal = float(input("Enter the projected sales: "))

# Calculate the projected profit.
profit = salesTotal * 0.23

# Print the projected profit.
print ("The projected profit is ", format(profit, '.2f'))
```



Programming Exercise 3-1

```
# Get the number.
number = int(input('Enter an integer: '))

# Determine if the number is positive, negative or zero.
if number > 0:
    print('Positive')
elif number < 0:
    print('Negative')
else:
    print('Zero')

# Determine if the number is even or odd.
if number % 2 == 0:
    print('Even')
else:
    print('Odd')
```

Programming Exercise 3-2

```
# Local variables
lengthA = 0.0
widthA = 0.0
areaA = 0.0
lengthB = 0.0
widthB = 0.0
areaB = 0.0

# Get length A
lengthA = float(input('Enter length A: '))

# Get width A
widthA = float(input('Enter width A: '))

# Get length B
lengthB = float(input('Enter length B: '))

# Get width B
widthB = float(input('Enter width B: '))

# Calculate area A
areaA = lengthA * widthA

# Calculate area B
areaB = lengthB * widthB

# Print area comparison
print ('Area A:', format(areaA, '.2f'))
print ('Area B:', format(areaB, '.2f'))
if  areaA > areaB:
    print ('Area A is greater than Area B.')
elif  areaA < areaB:
    print ('Area B is greater than Area A.')
else:
    print ('Area A is equal to Area B.')
```

Programming Exercise 4-1

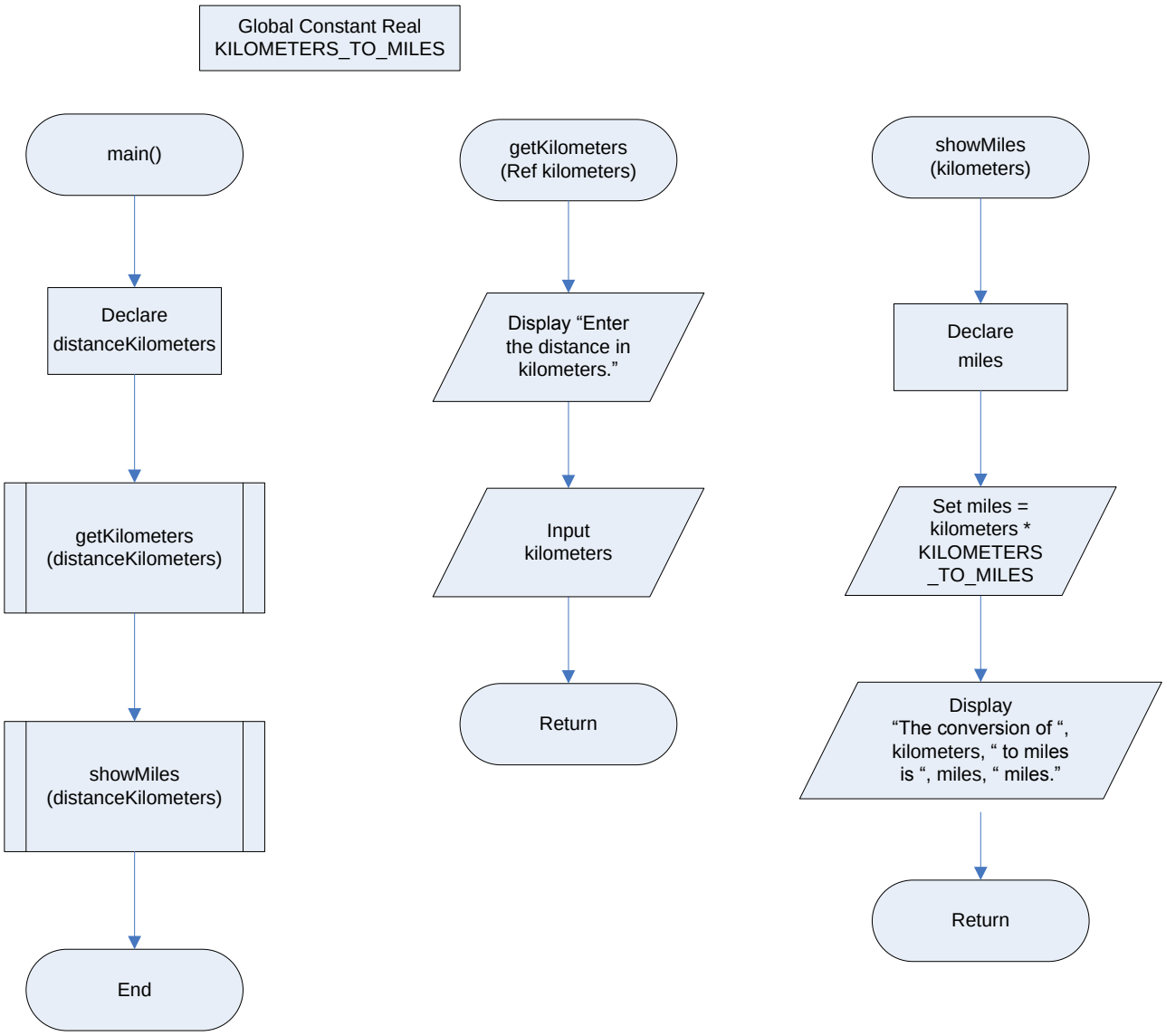
```
# Initialize variables for bugs and
# total number of bugs collected.
bugs = 0
total = 0

# Get number of bugs collected each day
for day in range(5):
    bugs = int(input('Enter the number of bugs collected today: '))
    total += bugs

# Display the total number of bugs collected.
print ('Total bugs collected: ', total)
```

Programming Exercise 4-2

```
# Declare and initialize a variable
# for the calories burned per minute.
caloriesPerMinute = 4.2

# Declare variables for the number of calories burned,
# and the number of minutes.
caloriesBurned = 0.0
minutes = 0

print ('Minutes\t\tCalories Burned')
print ('-------------------------------')

# Execute the for loop to display calories burned.
for minutes in range(10, 31, 5):
    caloriesBurned = caloriesPerMinute * minutes
    print (minutes, "\t\t", caloriesBurned)
```

Programming Exercise 5-1

```
# Global constant for conversion
KILOMETERS_TO_MILES = 0.6214

# main def
def main():
    # Local variables
    mykilometers = 0.0   # Variable to hold the distance in kilometers

    # Get distance in kilometers
    mykilometers = float(input("Enter the distance in kilometers: "))

    # Print miles
    showMiles(mykilometers)

# The showMiles function accepts kilometers as an argument
# and prints the equivalent miles.
def showMiles(kilometers):
    #Declare local variables
    miles = 0.0

    miles = kilometers * KILOMETERS_TO_MILES
    print ("The conversion of", format(kilometers, '.2f'), "kilometers")
    print ("to miles is", format(miles, '.2f'), "miles.")

# Call the main function.
main()
```



Programming Exercise 5-2

```
# main function
def main():
    # Call the function and print the result for testing.
    print(repeat('Hi', 3))


# The repeat function accepts text and multiplier
# and returns a string of text repeated multiplier times.
def repeat(text, multiplier):

    # Initialize output to an empty string.
    output = ''

    # Repeat multiplier times.
    for i in range(multiplier):

        # Concatenate text to the end of output.
        output = output + text

    # Return output.
    return output


# Call the main function.
main()
```

Programming Exercise 6-1

```
def main():
    # Declare local variables
    contents = ''

    # Open numbers.txt file for reading
    infile = open('numbers.txt', 'r')

    # Read in data and store in content
    contents = infile.read()

    # Close file
    infile.close()

    # Print contents
    print(contents)

# Call the main function.
main()
```

Programming Exercise 6-2

```
def main():
    # Declare variables
    line = ''
    counter = 0

    # Prompt for file name
    fileName = input('Enter the name of the file: ')

    # Open the specified file for reading
    infile = open(fileName, 'r')

    # Priming read
    line = infile.readline()
    counter = 1

    # Read in and display first five lines
    while line != '' and counter <= 5:
     # Strip '\n'
        line = line.rstrip('\n')
        print(line)
        line = infile.readline()
        # Update counter when line is read
        counter +=1

    # Close file
    infile.close()

# Call the main function.
main()
```

Programming Exercise 7-1

```
# The main function.
def main():
    # Declare local variables.
    numbers = [74, 19, 105, 20, -2, 67, 77, 124, -45, 38]
    total = 0

    # Start with an empty list.
    valid_numbers = []

    # Loop through the numbers list and append values
    # between 0 and 100 to the valid_numbers list.
    # Also add the number to the total if it is valid.
    for num in numbers:
        if num >= 0 and num <= 100:
            valid_numbers.append(num)
            total += num

    # Calculate the average.
    average = total / len(valid_numbers)

    # Display the results.
    print('Total of valid numbers:', total)
    print('Average of valid numbers:', \
          format(average, '.4f'))

# Call the main function.
main()
```

Programming Exercise 7-2

```
import random

def main():
    # Initialize list of numbers.
    number_list = [0, 0, 0, 0, 0, 0, 0]

    # Assign random numbers to list.
    for i in range(7):
        number_list[i] = random.randint(0, 9)

    # Display numbers in a single line.
    for i in range(7):
        print (number_list[i], end='')

        # Separate current number from next number.
        if i < 6:
            print(', ', end='')

# Call the main function.
main()
```

Programming Exercise 8-1

```
# The main function.
def main():
    # Receive user input.
    first_name = input ('Enter your first name: ')
    last_name = input ('Enter your last name: ')

    # Display the results.
    print('\nInitials: ', first_name[0].upper(), '.', \
          last_name[0].upper(), '.', sep='')

    print('Email Name: ', first_name[0].upper(), \
          first_name[1:], ' ', last_name.upper(), sep='')

    print('Username: ', first_name[0].lower(), \
          last_name.lower(), sep='')


# Call the main function.
main()
```

Programming Exercise 8-2

```
def main():
    # Get a string of numbers as input from the user.
    number_string = input('Enter a sequence of digits ' \
                          'with nothing separating them: ')

    # Call string_total method, and store the total.
    total = string_total(number_string)

    # Display the total.
    print('The total of the digits in the ' \
          'string you entered is', total)


# The string_total method receives a string and returns
# the total of all the digits contained in the string.
# The method assumes that the string does not contain
# non-digit characters
def string_total(string):
    # Local variables
    total = 0
    number = 0

    # Step through each character in the string.
    for i in range(len(string)):
        # Convert the character to an integer.
        number = int(string[i])
        # Add the value to the running total.
        total += number

    # Return the total.
    return total

# Call the main function.
main()
```

Programming Exercise 9-1

```
# The main function.
def main():
    # Initialize dictionaries.
    mean_radius = {'io':1821.6, 'europa':1560.8, \
                   'ganymede':2634.1, 'callisto':2410.3}

    surface_gravity = {'io':1.796, 'europa':1.314, \
                       'ganymede':1.428, 'callisto':1.235}


    orbital_period = {'io':1.769, 'europa':3.551, \
                      'ganymede':7.154, 'callisto':16.689}



    # Get input from user.
    moon = input('Enter name of Galilean moon of Jupiter: ')

    # Convert to lowercase for reliable matching in dictionaries.
    moon = moon.lower()

    # Show error message if name does not exist.
    # Otherwise, display details of specified moon.
    if moon not in mean_radius:
        print(moon.title(), 'is an invalid moon name.')
    else:
        print('Details of', moon.title(), 'are:')
        print('Mean Radius:', mean_radius[moon], 'km')
        print('Surface Gravity:', surface_gravity[moon], 'm/s²')
        print('Orbital Period:', orbital_period[moon], 'days')


# Call the main function.
main()
```

Programming Exercise 9-2

```
import random

def main():
    # Initialize dictionary
    capitals = {'Alabama':'Montgomery', 'Alaska':'Juneau',
                'Arizona':'Phoenix', 'Arkansas':'Little Rock',
                'California':'Sacramento', 'Colorado':'Denver',
                'Connecticut':'Hartford', 'Delaware':'Dover',
                'Florida':'Tallahassee', 'Georgia':'Atlanta',
                'Hawaii':'Honolulu', 'Idaho':'Boise',
                'Illinois':'Springfield', 'Indiana':'Indianapolis',
                'Iowa':'Des Moines', 'Kansas':'Topeka',
                'Kentucky':'Frankfort', 'Louisiana':'Baton Rouge',
                'Maine':'Augusta', 'Maryland':'Annapolis',
                'Massachusetts':'Boston', 'Michigan':'Lansing',
                'Minnesota':'Saint Paul', 'Mississippi':'Jackson',
                'Missouri':'Jefferson City', 'Montana':'Helena',
                'Nebraska':'Lincoln', 'Nevada':'Carson City',
                'New Hampshire':'Concord', 'New Jersey':'Trenton',
                'New Mexico':'Santa Fe', 'New York':'Albany',
                'North Carolina':'Raleigh', 'North Dakota':'Bismarck',
                'Ohio':'Columbus', 'Oklahoma':'Oklahoma City',
                'Oregon':'Salem', 'Pennsylvania':'Harrisburg',
                'Rhode Island':'Providence', 'South Carolina':'Columbia',
                'South Dakota':'Pierre', 'Tennessee':'Nashville',
                'Texas':'Austin', 'Utah':'Salt Lake City',
                'Vermont':'Montpelier', 'Virginia':'Richmond',
                'Washington':'Olympia', 'West Virginia':'Charleston',
                'Wisconsin':'Madison', 'Wyoming':'Cheyenne'}

    # Local variables
    correct = 0
    wrong = 0
    next_question = True
    index = 0
    user_solution = ''

    # Continue until user quits the game.
    while next_question:

        # Get access to the list of state names.
        state_iterator = iter(capitals)

        # Get a random state name for the question.
        index = (random.randint(1,50) - 1)
        for i in range (index-1):
            temp = state_iterator.__next__()
        current_state = str(state_iterator.__next__())

        # Get user solution.
        user_solution = input('What is the capital of ' + \
                              current_state + \
                              '? (or enter 0 to quit): ')
        # User wants to quit the game.
        if user_solution == '0':
```

```
            next_question = False
            print('You had', correct, 'correct responses and',\
                  wrong, 'incorrect responses.')
        # User solution is correct.
        elif user_solution == capitals[current_state]:
            correct = correct + 1
            print('That is correct.')
        # User solution is incorrect.
        else:
            wrong = wrong + 1
            print('That is incorrect.')

# Call the main function.
main()
```

Programming Exercise 10-1

```
# pet.py
class Pet:
    def __init__(self, name, animal_type, age):
        self.__name = name;
        self.__animal_type = animal_type
        self.__age = age

    def set_name(self, name):
        self.__name = name

    def set_animal_type(self, animal_type):
        self.__animal_type = animal_type

    def set_age(self, age):
        self.__age = age

    def get_name(self):
        return self.__name

    def get_animal_type(self):
        return self.__animal_type

    def get_age(self):
        return self.__age


# Exercise 10-1.py
import pet

def main():
    # Local variables
    pet_name = ""
    pet_type = ""
    pet_age = 0

    # Get pet data.
    pet_name = input('Enter the name of the pet: ')
    pet_type = input('Enter the type of animal: ')
    pet_age = int(input('Enter the age of the pet: '))

    # Create an instance of Pet.
    mypet = pet.Pet(pet_name, pet_type, pet_age)

    # Display the data that was entered.
    print ('Here is the data that you entered: ')
    print ('Pet name: ', mypet.get_name())
    print ('Animal type: ', mypet.get_animal_type())
    print ('Age of pet: ', mypet.get_age())

# Call the main function.
main()
```

Programming Exercise 10-2

```
# car.py
class Car:
    def __init__(self, year_model, make):
        self.__year_model = year_model;
        self.__make = make
        self.__speed = 0

    def accelerate(self):
        self.__speed += 5

    def brake(self):
        self.__speed -=5

    def get_speed(self):
        return self.__speed


# Exercise 10-2.py
import car

def main():
    # Create an instance of Car.
    my_car = car.Car('2008', 'Honda Accord')

    # Accelerate 5 times.
    print('car is accelerating: ')
    for i in range(5):
        my_car.accelerate()
        print ('Current speed: ', my_car.get_speed())
    print()

    # Brake 5 times.
    print ('car is braking: ')
    for i in range(5):
        my_car.brake()
        print ('Current speed: ', my_car.get_speed())

# Call the main function.
main()
```

Programming Exercise 11-1

```
class Employee:
    def __init__(self, name, id_number):
        self.__name = name
        self.__id_number = id_number

    def set_name(self, name):
        self.__name = name

    def set_id_number(self, id_number):
        self.__id_number = id_number

    def get_name(self):
        return self.__name

    def get_id_number(self):
        return self.__id_number

class ProductionWorker(Employee):
    def __init__(self, name, id_number, shift_number, pay_rate):
        # Call superclass __init__ method.
        Employee.__init__(self, name, id_number)

        # Initialize the shift_number and pay_rate attributes.
        self.__shift_number = shift_number
        self.__pay_rate = pay_rate

    # Mutator functions for shift_number and pay_rate.
    def set_shift_number(self, shift_number):
        self.__shift_number = shift_number

    def set_pay_rate(self, pay_rate):
        self.__pay_rate = pay_rate

    # Accessor functions for shift_number and pay_rate.
    def get_shift_number(self):
        return self.__shift_number

    def get_pay_rate(self):
        return self.__pay_rate
```

```
# Exercise 11-1.py
import emp

def main():
    # Local variables
    worker_name= ''
    worker_id = ''
    worker_shift = 0
    worker_pay = 0.0

    # Get data attributes
    worker_name = input('Enter the name: ')
    worker_id = input('Enter the ID number: ')
    worker_shift = int(input('Enter the shift number: '))
    worker_pay = float(input('Enter the hourly pay rate: '))

    # Create an instance of ProductionWorker
    worker = emp.ProductionWorker(worker_name, worker_id, \
                                  worker_shift, worker_pay)

    # Display information
    print ('Production worker information:')
    print ('Name:', worker.get_name())
    print ('ID number:', worker.get_id_number())
    print ('Shift:', worker.get_shift_number())
    print ('Hourly Pay Rate: $', \
           format(worker.get_pay_rate(), ',.2f'), sep='')

# Call the main function.
main()
```

Programming Exercise 12-1

```
# The main function.
def main():

    # Receive user input.
    number = int(input('Enter starting number: '))

    # Display the numbers.
    print_num(number)


# The print_num function is a recursive function
# that accepts an integer argument, n, and prints
# every second number from n down to a minimum of 0.
def print_num(n):
    if n >= 0:
        print(n)
        print_num(n - 2)


# Call the main function.
main()
```

Programming Exercise 12-2

```
def main():
    # Local variables
    num1 = 0
    num2 = 0

    # Get the first positive nonzero integer from the
    # user.
    while num1 <= 0:
        num1 = int(input('Enter the first number: '))

    # Get the second positive nonzero integer from the
    # user.
    while num2 <= 0:
        num2 = int(input('Enter the second number: '))

    # Call the multipy function, and display the product.
    print(num1, 'times', num2, 'is', multiply(num1, num2))

# The multiply function is a recursive function that
# accepts two arguments into the parameters x and y,
# and returns the value of x times y.
# The function assumes that x and y will always hold
# positive nonzero integers.
def multiply(x, y):
    if x == 0 or y == 0:
        return 0
    else:
        return x + multiply(x, y - 1)

# Call the main function.
main()
```

Programming Exercise 13-1

```
import tkinter

class ShowInfoGUI:
    def __init__(self):
        # Create the main window
        self.main_window = tkinter.Tk()

        # Create two frames
        self.top_frame = tkinter.Frame(self.main_window)
        self.bottom_frame = tkinter.Frame(self.main_window)

        # Create a blank label in the top frame
        self.value = tkinter.StringVar()
        self.address_label = tkinter.Label(self.top_frame, \
                    textvariable= self.value)

        # Create the two buttons in the bottom frame
        self.address_button = tkinter.Button(self.bottom_frame, \
                text = 'Show Info', command = self.show_info)
        self.quit_button = tkinter.Button(self.bottom_frame, \
                text = 'Quit', command = self.main_window.destroy)

        # Pack the label
        self.address_label.pack()
        #Pack the buttons
        self.address_button.pack(side = 'left')
        self.quit_button.pack(side = 'left')

        # Pack the frames
        self.top_frame.pack()
        self.bottom_frame.pack()

        # Enter the tkinter main loop
        tkinter.mainloop()

    # Define the show_info function
    def show_info(self):
        self.value.set('Steven Marcus\n274 Baily Drive\n' \
                       'Waynesville, NC 27999')

# Create an instance of ShowInfoGUI
show_info = ShowInfoGUI()
```

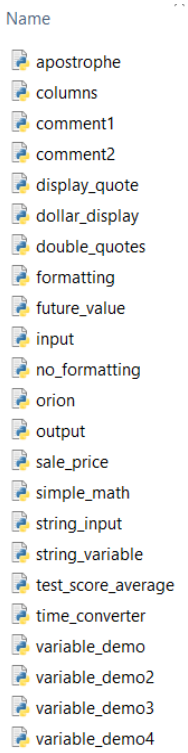
Programming Exercise 13-2

```
import tkinter

class LatinTranslatorGUI:
    def __init__(self):
        # Create the main window
        self.main_window = tkinter.Tk()

        # Create two frames
        self.top_frame = tkinter.Frame(self.main_window)
        self.bottom_frame = tkinter.Frame(self.main_window)

        # Create a blank label in the top frame
        self.value = tkinter.StringVar()
        self.word_label = tkinter.Label(self.top_frame, \
                                        textvariable= self.value)

        # Create the buttons in the bottom frame
        self.sinister_button = tkinter.Button(self.bottom_frame, \
                                              text = 'sinister', \
                                              command = self.show_word1)
        self.dexter_button = tkinter.Button(self.bottom_frame, \
                                            text = 'dexter', \
                                            command = self.show_word2)
        self.medium_button = tkinter.Button(self.bottom_frame,
                                            text = 'medium', \
                                            command = self.show_word3)

        # Pack the label
        self.word_label.pack()

        # Pack the buttons
        self.sinister_button.pack(side = 'left')
        self.dexter_button.pack(side = 'left')
        self.medium_button.pack(side = 'left')

        # Pack the frames
        self.top_frame.pack()
        self.bottom_frame.pack()

        # Enter the tkinter main loop
        tkinter.mainloop()

    # Define the show_word functions
    def show_word1(self):
        self.value.set('left')
    def show_word2(self):
        self.value.set('right')
    def show_word3(self):
        self.value.set('center')

# Create an instance of LatinTranslatorGUI
latin_translator = LatinTranslatorGUI()
```

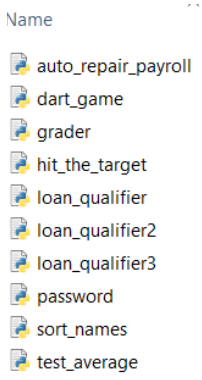
Chapter 2 to 13

Source Code in Python Format

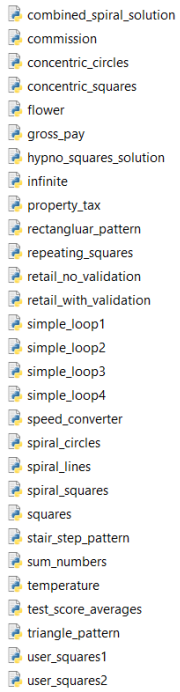
Chap02



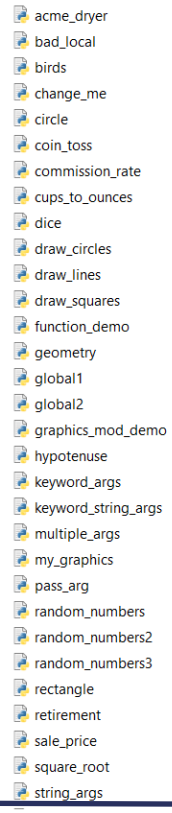
Chap03



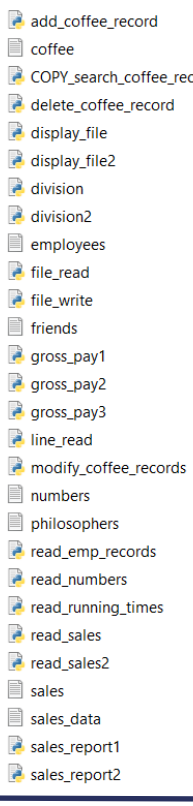
Chap04

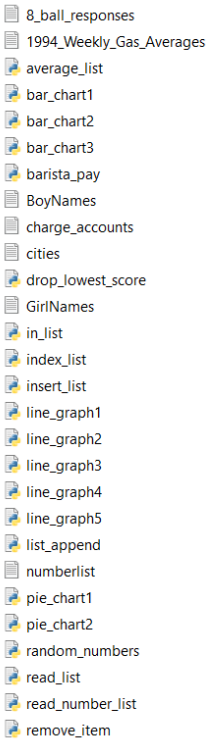


Chap05

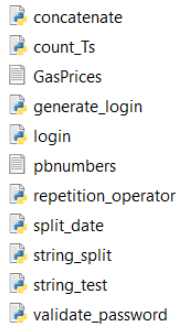


Chap06

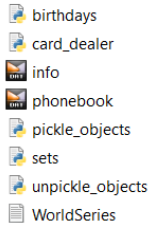




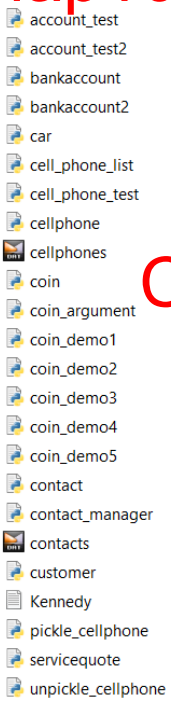
Chap08



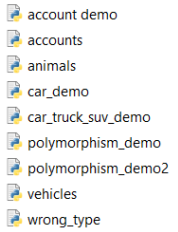
Chap09



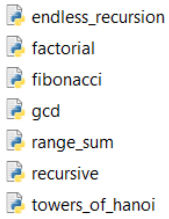
Chap10



Chap11



Chap12



Chap13