Chapter 9, section 10 {exercises}

شایسته گیوه ای

1. What is the difference between a class and an object?

>> Class is a blueprint for objects and objects instances of a class .

2. What are some other names for the term instance variable?

>> Attributes and fields .

3. What is another name for the term method?

>> Oparations.

4. What symbol associates an object with a method invocation?

>> To call a method in a object we use "dots" ,so class- name .method (parametr list)

5. How does a method differ from a function?

>> A method is ultimately a function, but it is defined and exists inside an object.

6. What method from the string class returns a new string with no leading or trailing whitespace?

>> The strip method,so: string -- obj.strip()

7. What function returns the length of its string argument?

>> The len function,so len (sting-- obj)

8. What type of object does the open function return?

>> Well , it returns a textiowrapper object witch is in the io module , but we normally call it a file object.

9. What does the second parameter of the open function represent?

>> It is the mode we want to open the file with.

there are 3 modes :

"R" for read (just to read the file)

"W" for write (whitch creates a new file / or deletes existing files data and starts fresh)

"A" for append (to append data to the file)

10. Write a program that stores the first 100 integers to a text file named numbers.txt. Each number

should appear on a line all by itself.

```
>> F=open ("numbers .txt ,”w”);
For i in range (100):
F.write (f” {i}n “);
F.close();
```

11. Complete the following function that reads a collection of integers from a text file named numbers.txt.

Each number in the file appears on a line all by itself. The function accepts a single parameter, a string

text file name. The function returns the sum of the integers in the file.

```
def sumfile(filename):
        # Add your code here . . .
```

```
>> Def sumfile (filename);
        F=open (filename,"r");
Sum=o;
For line in f:
Sum+=int(line);
```

Return sum;

F.close();

12. Provide the syntactic sugar for each of the following methods of the Fraction class:

(a)----- sub---- a>>--- sub---(b) is a-b

(b) ----eq----a>>---eq----(b) is a ==b

(c)----- neg -----a>>__neg() is-a

(d)----- gt----a>>---get _(b) is a>b

13. How is using a Turtle object from Python's Turtle graphics module different from using the free functions; for example, t.penup() versus penup()?

>> when using a Turtle object from Python's Turtle.graphics module, you first need to create an instance of the Turtle class by calling its constructor. For example:

import turtle

t = turtle.Turtle()

Once you have created an instance of the Turtle class, you can call its methods to control its behavior. For example:

t.penup()

t.goto(100, 100)

t.pendown()

t.circle(50)

On the other hand, when using free functions like penup(), you don't need to create an instance of any class. You can simply call the function directly. For example:

import turtle

turtle.penup()

turtle.goto(100, 100)

turtle.pendown()

turtle.circle(50)

In summary, using a Turtle object from Python's Turtle.graphics module requires creating an instance of the Turtle class and then calling its methods, while using free functions like penup() can be called directly without creating an instance.

14. For each of the drawings below write a program that draws the shape using a Turtle object from

Python’s Turtle graphics module.

triangle

>> From turtle import *

T=turtle();

T.pensize(5)

```
For i in range (3):
    T.forward(200);
    T.left (120)
T.hideturtle();
Exitonclick();
```

Star

```
 >> From turtle import *
T=turtle();
 T.pensize(5)
t.left(36)
t.forward(300);

For i in range (4):
    T.forward(300);
    T.left (144)
T.hideturtle();
Exitonclick();
```

شكل 3

```
>> From turtle import *
T=turtle();
 T.pensize(5)
t.left(75);
```

```
t.forward(150)
right—bool=true;
for I in range (9):
if right—bool:
t.right(150);
else:
t.left(150);
t.forward(150);
right—bool=not right—bool

T.hideturtle();
Exitonclick();
```

شکل مربع تو پر

```
>> From turtle import *
Def create-square(amount):
      For I in range (amount):
          for I in range (4):
              t.forward(20);
              t.right(90);
t.penup();
  t.forward(20);
    t.pendown();
```

```
t=turtle();
t.pensize(5)

t.left(90);
x=0;
  for I in range(5):
       t.penup();
        t.setposition(x,0);
        t.pendown();
         create-square(5)
         x +=20;
T.hideturtle();
Exitonclick();
```

دایره

```
>> From turtle import*
t=turtle();
t.pensize(5)
t.circle(100);

T.hideturtle();
Exitonclick();
```

دایره تو پر

```
>> From turtle import *
Def square():
      For I in range (4):
            t.forward(150);
            t.right(90);
t=turtle();
t.pensize(3)
t.left(90);
  for I in range(36):
square();
t.right();
T.hideturtle();
Exitonclick();
```

15. Does Python permit a programmer to change one symbol in a string object? If so, how?

>> No, Python strings are immutable, which means that once a string is created, it cannot be changed. However, you can create a new string with the desired change by using string concatenation or slicing. For example:

```
string = "hello"
new_string = string[:3] + "p" + string[4:]
```

print(new_string) # "helpo"

In this example, we create a new string by slicing the original string and concatenating the desired change ("p") in the middle.

16. What would be the consequences if a turtle.Turtle object were immutable?

>> If a turtle.Turtle object were immutable, it would not be possible to change its properties or attributes after it has been created. This would have several consequences:

1. Limited functionality: Since the turtle object cannot be modified, it would have limited functionality and would not be able to perform complex tasks.

2. Inability to move: The turtle object is used for drawing and moving around the screen. If it were immutable, it would not be able to move, making it useless for its intended purpose.

3. Limited customization: Customization of the turtle object would be limited since its properties could not be changed.

4. Difficulty in debugging: If there were any errors in the code related to the turtle object, it would be difficult to debug since the object cannot be modified.

Overall, making a turtle.Turtle object immutable would severely limit its usefulness and functionality in programming.

17. In the context of programming, what is garbage?

>> In programming, garbage refers to data or memory that is no longer needed or used by a program but still occupies space in the computer's memory. This can happen when a program creates objects or variables that are not properly disposed of or released after they are no longer needed, leading to memory leaks and potentially slowing down the program or causing it to crash. Garbage collection is the process of automatically identifying and removing unused data from a program's memory to improve performance and prevent errors.

18. What is garbage collection, and how does it work in Python?

>> Garbage collection is the process of automatically freeing up memory that is no longer being used by a program. In Python, garbage collection is handled by the Python interpreter's built-in garbage collector.

The garbage collector works by periodically scanning the memory used by a Python program to identify objects that are no longer being referenced or used. Once these objects are identified, the garbage collector frees up the memory they were using so that it can be used for other purposes.

Python's garbage collector uses a combination of reference counting and cycle detection to identify and free up unused memory. Reference counting involves keeping track of how many references there are to each object in memory, and freeing up an object's memory when its reference count drops to zero. Cycle detection involves identifying groups of objects that reference each other in a circular pattern, which can prevent them from being freed up by reference counting alone.

Overall, Python's garbage collector helps to ensure that programs don't run out of memory due to unused objects taking up space in memory.

19. Consider the following code:

```
a = "ABC"
b = a
c = b
a = "XYZ"
```

(a) At the end of this code’s execution what is the reference count for the string object "ABC"?

(b) At the end of this code’s execution is b an alias of a?

(c) At the end of this code’s execution is b an alias of c?

>> (a) The reference count for the string object "ABC" is 0 at the end of this code's execution, as there are no variables or objects referencing it.

(b) No, b is not an alias of a at the end of this code's execution. Initially, a and b are separate variables pointing to the same string object "ABC". However, when c is assigned to b, b now points to a new string object "XYZ", while a still points to the original "ABC" object.

(c) Yes, b is an alias of c at the end of this code's execution. After c is assigned to b, they both point to the same string object "ABC". When a is then assigned to "XYZ", both b and c still point to the original "ABC" object.