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

   In Python, a class is a blueprint or a template for creating objects, while an object is an instance of a class.
   A class defines the attributes and methods that an object will have, but it does not actually create the object itself. It is like a blueprint that describes how an object should be constructed.
   An object, on the other hand, is a specific instance of a class. It is created from the blueprint defined by the class and has its own unique set of attributes and methods.
   In summary, a class defines the characteristics of objects that can be created from it, while an object is an instance of that class with its own unique properties.

در پایتون، یک کلاس یک طرح اولیه یا یک الگو برای ایجاد اشیا است، در حالی که یک شی نمونه ای از یک کلاس است.

یک کلاس ویژگی ها و متدهایی را که یک شی خواهد داشت تعریف می کند، اما در واقع خود شی را ایجاد نمی کند.مانند طرحی است که نحوه ساخت یک شی را توضیح می دهد.

از طرح تعریف شده توسط کلاس ایجاد می شود و .از سوی دیگر، یک شیء، یک نمونه خاص از یک کلاس است مجموعه ای از ویژگی ها و متدهای منحصر به فرد خود را دارد.

به طور خلاصه، یک کلاس ویژگی های اشیایی را که می توان از آن ایجاد کرد تعریف می کند، در حالی که یک شی نمونه ای از آن کلاس با ویژگی های منحصر به فرد خود است.

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

   Some other names for instance variables in Python are:

   1. Object attributes
   2. Object properties
   3. Instance attributes
   4. Instance properties
   5. Member variables

برخی از نام های دیگر به عنوان مثال متغیرها در پایتون عبارتند از:

1.ویژگی های شی
2.خواص شی
3.صفات نمونه
4.خواص نمونه
5.متغیرهای عضو

3. **What is another name for the term method?**

   Function.

تابع .

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

The dot (.) symbol associates an object with a method invocation in Python.

نماد نقطه (.) یک شی را با فراخوانی متد در پایتون مرتبط می کند .

5. How does a method differ from a function?

a function is a block of code that performs a specific task and returns a value (if necessary). A method, on the other hand, is a function that is associated with an object and can access its data.

Here are some key differences between methods and functions in Python:

1. Syntax: A function is defined using the "def" keyword followed by the function name and parameters (if any). A method is defined inside a class using the same syntax as a function, but it is called on an instance of the class.

2. Scope: Functions are standalone entities that can be called from anywhere in the program. Methods are bound to objects and can only be called on instances of that object.

3. Access to data: Methods have access to the data stored in an object's attributes, while functions do not.

4. Return value: Both functions and methods can return values, but methods often modify the state of an object as well.

In summary, while both functions and methods perform tasks in Python, methods are specific to objects and have access to their data.

از طرف ,یک تابع بلوکی از کد است که وظیفه خاصی را انجام می دهد و مقداری را (در صورت لزوم) برمی گرداند
دیگر یک متد تابعی است که با یک شی مرتبط است و می تواند به داده های آن دسترسی داشته باشد .

.در اینجا چند تفاوت کلیدی بین متدها و توابع در پایتون وجود دارد :

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

The method is called strip() .

متدstrip() نامیده می شود.

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

The len() function returns the length of its string argument.

طول آرگومان رشته خود را برمی گرداند len() تابع .

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

The open function in Python returns a file object.

در پایتون یک شی فایل را برمی گرداند open تابع.

.

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

The second parameter of the open function in Python represents the mode in which the file is opened. It specifies whether the file is being opened for reading, writing, or both. The mode can be specified using a string that contains one or more of the following characters:

- 'r': read mode (default)
- 'w': write mode
- 'x': exclusive creation mode
- 'a': append mode
- 'b': binary mode
- '+': read and write mode

For example, to open a file for writing in binary mode, you would use the following code:

```
file = open('filename', 'wb')
```

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.

Here's the Python code to store the first 100 integers to a text file named numbers.txt:

```
# Open the file for writing
with open("numbers.txt", "w") as f:
    # Write the first 100 integers to the file
    for i in range(1, 101):
        f.write(str(i) + "\n")
```

Explanation:

- We use the open() function to open a file named "numbers.txt" in write mode ("w").
- We use a with statement to ensure that the file is properly closed after we're done writing to it.
- We use a for loop to iterate over the range of integers from 1 to 100 (range(1, 101)).
- For each integer, we convert it to a string using str(i) and append a newline character ("\n") so that each number appears on its own line.
- We use the write() method of the file object (f.write()) to write each number (as a string) to the file.

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):
    total = 0
    with open(filename, 'r') as file:
        for line in file:
            total += int(line)
    return tota
```

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

(a) __sub__ : f.__sub__(g) is equivalent to f - g

(b) __eq__ : f.__eq__(g) is equivalent to f == g

(c) __neg__ : f.__neg__() is equivalent to -f

(d) __gt__ : f.__gt__(g) is equivalent to f > g

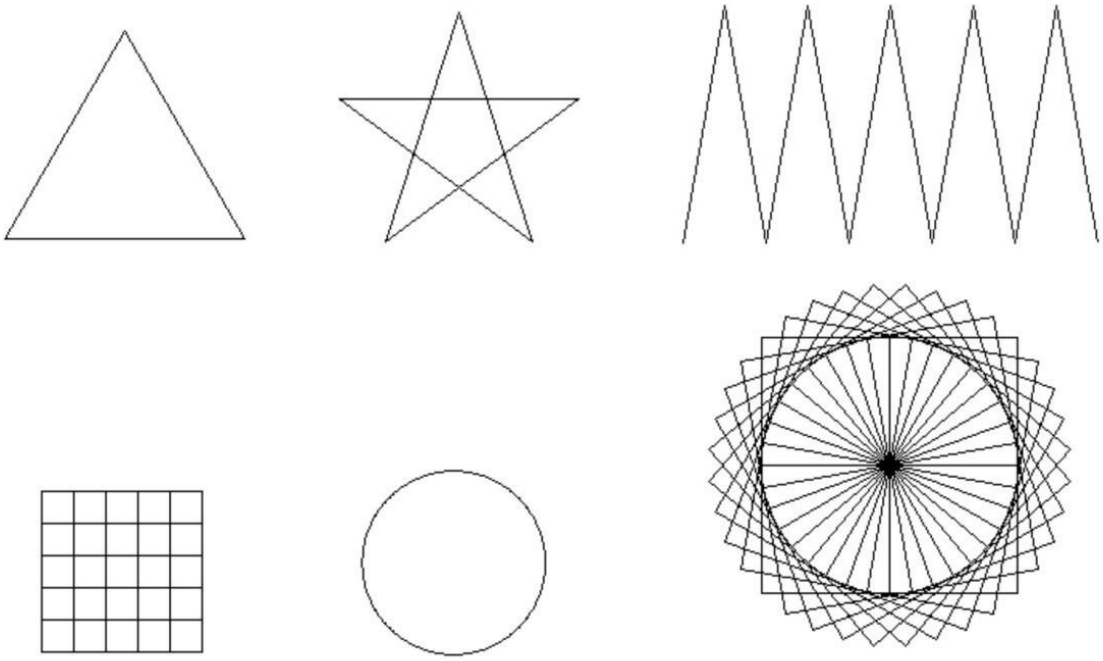
13. How is using a Turtle object from Python's Turtle graphics module different from using the free functions; for example, t.penup() versus penup()?

Using a Turtle object from Python's Turtle graphics module allows you to create and manipulate a specific turtle instance, while using the free .functions operates on the default turtle instance.

For example, if you create a Turtle object named "t" using the code "t = turtle.Turtle()", you can then use methods such as "t.penup()" to lift the pen off the canvas for that specific turtle instance. If you were to use the free function "penup()", it would lift the pen off the canvas for the default turtle instance.

In summary, using a Turtle object allows for more control and specificity in manipulating individual turtles, while using free functions operates on a default turtle 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.



It is impossible to draw the shape here, that's why I write the code

Sample code:

```
import turtle

# Create a Turtle object
t = turtle.Turtle()

# Draw a square
for i in range(4):
    t.forward(100)
    t.right(90)

# Draw a triangle
for i in range(3):
    t.forward(100)
    t.left(120)

# Draw a circle
```

```
t.circle(50)

# Draw a star
for i in range(5):
    t.forward(100)
    t.right(144)

# Hide the Turtle object
t.hideturtle()

# Keep the window open until closed manually
turtle.done()
```

Using the Turtle graphic module, this code can be changed to draw different shapes and patterns

*****

Triangle code :

```
def turtle():
    while True:
        import turtle
        t = turtle.Turtle()
        window = turtle.Screen()
        t.forward(100)
        t.left(120)
        t.forward(100)
        t.left(120)
        t.forward(100)
turtle()
```

*****

circle code:

```
from turtle import*
color('red' , 'yellow')
shape('turtle')
speed('fastest')
circle(100)
```

*****

star code:

```
import turtle
s=turtle.getscreen ()
t=turtle.Turtle ()
for i in range (5):
    t.fd(200)
    t.rt(144)
```

*****

Square code:

```
import turtle
t = turtle.Pen()
t.forward(90)
t.left(90)
t.forward(90)
t.left(90)
t.forward(90)
t.left(90)
t.forward(90)
t.left(90)
```

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 modified. However, you can create a new string by concatenating parts of the original string with the desired change.

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

If a turtle.Turtle object were immutable in Python, it would not be possible to change its properties or attributes after it has been created. This would limit the ability to manipulate the turtle's position, orientation, color, and other characteristics that are essential for creating complex graphics and animations.

For example, if the turtle object were immutable, it would not be possible to move the turtle forward or backward using the forward() or backward() methods. Similarly, it would not be possible to change the turtle's color using the color() method or change its pen size using the pensize() method.

Overall, making a turtle.Turtle object immutable would severely limit its usefulness for creating dynamic and interactive graphics in Python. Therefore, it is important that turtle objects remain mutable so that they can be manipulated and customized as needed.

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

In programming, garbage refers to the memory space that is no longer being used by a program but has not been released back to the operating system. This can occur when a program creates objects or variables in memory and then no longer needs them, but fails to properly delete or release them. Over time, this can lead to memory leaks and cause the program to slow down or crash. Garbage collection is the process of automatically identifying and removing unused memory in a program.

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 itself. Python uses a technique called reference counting to keep track of objects in memory. Cyclic garbage collection works by periodically scanning all objects in memory and identifying any groups of objects that are only referenced by each other (i.e., cyclic references). These groups of objects are then marked as garbage and can be safely deleted.

Overall, Python's garbage collection system allows programmers to focus on writing code without worrying too much about managing memory manually.
The Python interpreter cleans up
garbage through a process called garbage collection. Python uses a reference counting garbage collector
that automatically reclaims the space occupied by abandoned objects.

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"?
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) At the end of this code's execution is b an alias of a?
No, b is not an alias of a at the end of this code's execution. When b was assigned to a, it was pointing to the same object as a. However, when a was reassigned to "XYZ", b still points to the old object "ABC". Therefore, b and a are no longer aliases.

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

Yes, b is an alias of c at the end of this code's execution. When c was assigned to b, it was pointing to the same object as a (which was "ABC" at that time). Later on, when a was reassigned to "XYZ", c still points to the old object "ABC". Therefore, b and c are still aliases.