1. Why are functions advantageous to have in your programs?

The advantages of Python functions are as follows:

With the help of functions, we can avoid rewriting the same logic or code again and again in a program.

In a single Program, we can call Python functions anywhere and also call multiple times.

We can track a large Python program easily when it is divided into multiple functions.

The main achievement of Python functions is its Reusability. However, In a Python program, Function calling is always overhead.

- Reducing duplication of code.
- Improved reusability of code
- Write Clearer Code
- Bigger problems are broken down into smaller problems.

2. When does the code in a function run: when it's specified or when it's called?

The code in a function run only when it is called.

3. What statement creates a function?

A function is defined by using the **def** keyword, followed by a name of your choosing, followed by a set of parentheses which hold any parameters the function will take (they can be empty), and ending with a colon.

```
def hello():
    print("Hello, World!")
```

Our function is now fully defined, but if we run the program at this point, nothing will happen since we didn't call the function.

4. What is the difference between a function and a function call?

A function is a block of code that does a particular operation and returns a result. It usually accepts inputs as parameters and returns a result. The parameters are not mandatory.

A function is a piece of code which enhanced the reusability and modularity of your program. It means that piece of code need not be written again. A function call means invoking or calling that function. Unless a function is called there is no use of that function.

```
def add(a,b):
  return a+b
```

A function call is the code used to pass control to a function.

```
b = add(5,6)
```

b will have value 11

5. How many global scopes are there in a Python program? How many local scopes?

A variable is only available from inside the region it is created. This is called scope.

Local Scope

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

```
# A variable created inside a function is available inside that function:

def myfunc():
  x = 300
  print(x)
```

```
myfunc()
```

Global Scope

A variable created in the main body of the Python code is a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local.

```
# A variable created outside of a function is global and can be used by an

x = 300

def myfunc():
  print(x)

myfunc()

print(x)
```

Enclosing Scope in Python

In this code, ‘b’ has local scope in Python function ‘blue’, and ‘a’ has nonlocal scope in ‘blue’.

Of course, a python variable scope that isn’t global or local is nonlocal. This is also called the enclosing scope.

```
def red():
a=1
    def blue():
       b=2
       print(a)
       print(b)
    blue()
    print(a)
red()
```

Built-in Scope

Finally, we talk about the widest scope. The built-in scope has all the names that are loaded into python variable scope when we start the interpreter.

For example, we never need to import any module to access functions like print() and id().

6. What happens to variables in a local scope when the function call returns?

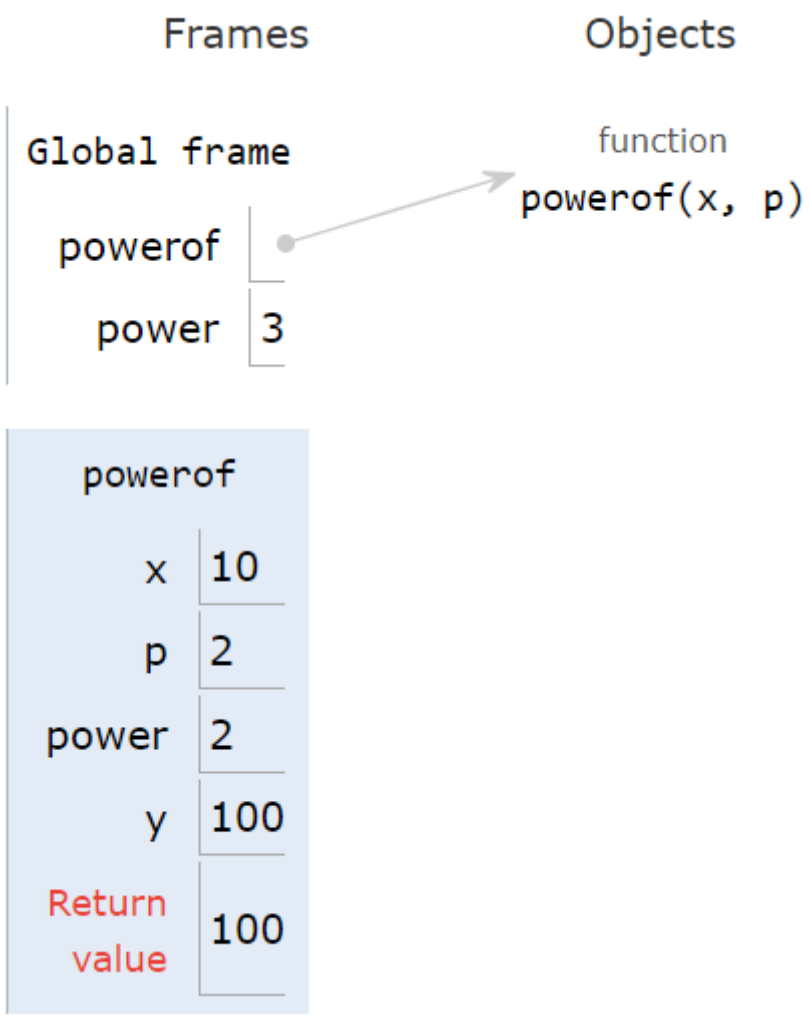
A local variable is declared within the function or program block, and it can be used inside the code block or subroutine in which it's declared. Local variable exists until the code of block execution; once the code executes, it destroys automatically.

Yes they will be deleted. Every time you call a function you put a "stack frame" on top of the stack where all the local variables live in, and when the function is done, the stack frame will be "teared down".

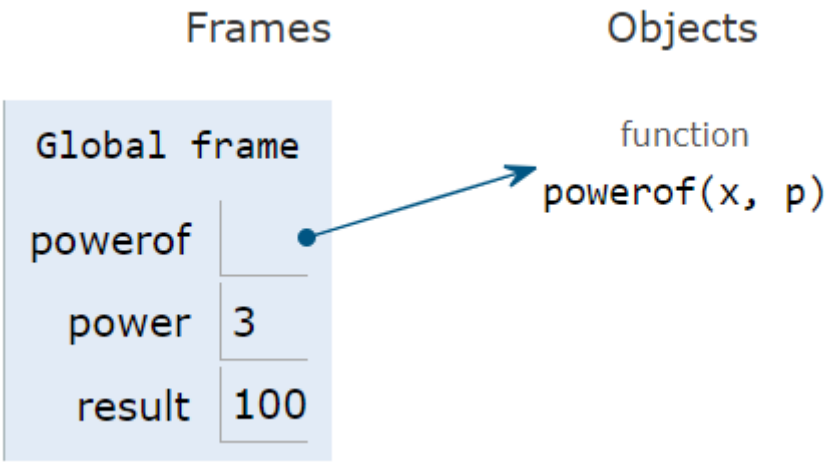
```
def badsquare(x):
    y = x ** power
    return y

power = 2
result = badsquare(10)
print(result)
```

Status of variables after calling the function



Status of variables after termination of function



The frame created for powerof function and its local variables are destroyed automatically while returning from the function

7. What is the concept of a return value? Is it possible to have a return value in an expression?

A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller. The statements after the return statements are not executed. If the return statement is without any expression, then the special value None is returned. A return statement is overall used to invoke a function so that the passed statements can be executed.

Yes, it is possible to have a return value in an expression.

The return() statement, like in other programming languages ends the function call and returns the result to the caller. It is a key component in any function or method in a code which includes the return keyword and the value that is to be returned after that. Some points to remember while using return():

The statements after the return() statement are not executed. return() statement can not be used outside the function. If the return() statement is without any expression, then the NONE value is returned.

```
def fun():
    statements
    .
    .
    return expression
```

8. If a function does not have a return statement, what is the return value of a call to that function?

Any function without an explicit return statement, or one with a return statement without a return value, will return None .

9. How do you make a function variable refer to the global variable?

If you want to modify a global variable inside a function, then you need to explicitly tell Python to use the global variable rather than creating a new local one. To do this, you can use one of the following: The global keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function. To create a global variable inside a function, you can use the global keyword.

If you use the global keyword, the variable belongs to the global scope:

```
def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

10. What is the data type of None?

The None keyword is used to define a null value, or no value at all.

None is not the same as 0, False, or an empty string. None is a data type of its own (NoneType) and only None can be None.

Assign the value None to a variable:

```
x = None

if x:
  print("Do you think None is True?")
elif x is False:
  print ("Do you think None is False?")
else:
  print("None is not True, or False, None is just None...")
```

```
None is not True, or False, None is just None...
```

11. What does the sentence import areallyourpetsnamederic do?

That import statement imports a module named areallyourpetsnamederic .

12. If you had a bacon() feature in a spam module, what would you call it after importing spam?

We can call the bacon() feature in a spam module using

```
spam.bacon()
```

13. What can you do to save a programme from crashing if it encounters an error?

When it encounters an error, the control is passed to the except block, skipping the code in between. We can move our code inside a try and except statement. Try running the program and it should throw an error message instead of crashing the program.

Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

14. What is the purpose of the try clause? What is the purpose of the except clause?

The try block lets you test a block of code for errors.

The except block lets you handle the error.

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

The try block will generate an exception, because x is not defined: