Ques 1: Why are functions advantageous to have in your programs?

Ans : Whenever, the programmer need to `re-write the same code` or make a `duplicate` a copy of that code using `Ctrl + C` and `Ctrl + V` it become `unfeasible or considered not a good practice to do` in large code bank or coding projects Hence, functions are use to `reduce the work of compiler and programmer` for the `compilation of`, and `executional space and time` taken by the duplicate code.

```
def functionName(parameters):
  """doc string"""
  1. statements (function body)
  2. return statement
```

```
# Example : For Custom or User-defined Function!
name = input("Write a name :  ")
message = input("Your Message :  ")

def greet(name, message):
  """This fuction is used to greet various people using their name and message."""
  return f"Good Day, {name}!\n{message}"
```

```
Write a name :  Arpit
Your Message :  Hope, you are doing great :)
```

```
print(greet(name, message))
```

```
Good Day, Arpit!
Hope, you are doing great :)
```

```
print(greet("Shreya", message)) # Anonymous name or Some random Parameter passing
```

```
Good Day, Shreya!
Hope, you are doing great :)
```

```
print(greet(name="Shudhansu Kumar", message="Hope, you are doing great\nYou and krish are pro
```

```
Good Day, Shudhansu Kumar!
Hope, you are doing great
You and krish are providing amazing..
Revolutionary educational stuffs to ineuron students
Thankyou :)
```

Observations :

As we can see in the above example, fuctions are very useful feature

Function helps:

1. In **reduction of code lines**.
2. In **reduction of effort** for writing multiple lines.
3. In **reduction of time consume** to write multi-line code.
4. It provide **Dynamic insertions or modification** (modification become easy)
5. It provide **Code re-usability** as a feature.

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

Ans : Whenever we **called** that function and passed the required parameter, if it the demand of that fuction than the **fuction start it's execution**.

When the **fuction specified**, at that time we only define our ** presentation, persistence and business logics** in our program but when we** called respective functions** than these three logic start there execution.

```
# Example Specification of functionsm using simple Gpay logic on spent money.

print("Please enter you account 4 digit pin : ", end='')
pin = int(input())

def presentationLogic(key, pin):

  # Authentication
  if(key==pin):
      clientValue=10000
      print("Amount of money you had spend today  : ", end='')
      spendedMoney=int(input())
      persistenceLogic(clientValue, spendedMoney)

def businessLogic(gpayBalance, spendedMoney):
  #Amount Deduction - business logic
  gpayBalance = gpayBalance - spendedMoney
  ##Successfull Transaction message
  print("Your Transaction is Successful ...")
  return gpayBalance

def persistenceLogic(clientValue, spendedMoney):
  database = clientValue # Variable as small Storage unit
  #After Udating the store value
  database = businessLogic(database, spendedMoney)
  gpayWallet(database)

def gpayWallet(balance):
```

```
  #To print the balance
  print(f"Your Current GPay account balance is : {balance}")

def login(pin):
  key = 3256 # Saved GPay Pin, inside app known by only user of the account
  if(key==pin):
    presentationLogic(key, pin)
  else:
    print("You have entered a wrong Pin")
    print("Try again ... ")
```

```
 Please enter you account 4 digit pin : 3256
```

Observations :

1. As we define all the functionalities from **Presentation Logic**, **Persistence Logic**, to **Presentation Logic.**
2. But, not even a single fuctionality works because we are **not calling** any of them. let's call the **login()** fuctionality

```
login(pin)
```

```
Amount of money you had spend today  : 3500
Your Transaction is Successful ...
Your Current GPay account balance is : 6500
```

3. Now, we can see all the fuctionalities working when we call login() function.
4. After calling the execution of **presentation Logic, Persistence Logic and Business Logic** works.

Ques 3: What statement creates a function?

Ans : `def` keyword is used to define the function in python.

Syntax of function in python is like:

```
def funtionName(parameter_1, parameter_2, .., parameter_n):
   """doc string, it gives description about the fuction.
    We can called it by using functionName.__doc___ """

    --- function body start ---
    Statement_1
```

```
    Statement_2
    .
    .
    .
    Statement_n
    --- function body end ---

  return value
```

This how we can create a function in python. Every language has their own Syntax to create a function.

Ques 4: What is the difference between a function and a function call?

Ans : From the above example we can easily understant the difference b/w function and function call.

When I defined the presentation logic, persistence logic and business logic I just define the fuctionality or the procedures or steps to accomplished a particular task.

Example: Let take the example of **businessLogic(gpayBalance, spendedMoney)** function

```
def businessLogic(gpayBalance, spendedMoney):
  gpayBalance = gpayBalance - spendedMoney
  print("Your Transaction is Successful ...")
  return gpayBalance
```

Here, we are giving gpayBalance and spendedMoney as **parameter** to businnessLogic() function.

inside, the function we update the gpayBalance by **substracting** the spendedMoney and than return the gpayBalance.

but it **won't work automatically**, we **pass our pin** first from **login() function** than it **authenticate** the value in **presentationLogic()** and that presentationLogic() function **calls** the **persistenceLogic()** and persistenceLogic() calls ** **bold text**businessLogic() **for the calculation and than the **values update** in the persistenceLogic() and than again **persistenceLogic()** function **calls gpayWallet()** function for printing.

Conclusion:

Fuction are the steps/procedure to accomplished certain task or to achieve the expected output/result.

Function calls are used for the execution of these steps/procedures or functions for

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

Ans : There is **only one global scope** in python and whenever we **call a function** a **local scope has been created**.

```
# Example
#Global variable
marksObtained=80

# function without parameter calling global variable from inside the function marksObtained()
def myMarks():
  print(marksObtained)
myMarks()

#printing global variable marksObtained
print(marksObtained)
```

```
80
80
```

Observation :

1. Hence, we can see 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**.

```
# Example
def myfunc():
  marksObtained = 95 # Local variable
  print(marksObtained)

myfunc() # gives 95
print(marksObtained) # print 80
```

```
95
80
```

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

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

Ans : local scope has been **destroyed** whenever the **function call returns** or the **return statement** of a function is **executed**. All the variable has been **destroyed** or forgotten or **cleared** from the memory.

```
#Example:

def screenLockCode():
  actualPin=1442
  return actualPin

def unlockMobile(pin):
  screenPin = pin
  actualPin = screenLockCode() - 10 # return value used in an expression
  print(f"location of actual pin is : {id(actualPin)}")
  if(actualPin == screenPin):
    print("Mobile Phone unlocked successfully ...")
  else:
    print("Invalid Pin has been entered !!")

unlockMobile(1432)
```

```
location of actual pin is : 140619031452592
Mobile Phone unlocked successfully ...
```

```
print(f"location of actual pin is : {id(actualPin)}") #Not available after function call
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-5646fce7d462> in <module>()
----> 1 print(f"location of actual pin is : {id(actualPin)}") #Not available
after function call

NameError: name 'actualPin' is not defined

 SEARCH STACK OVERFLOW
```

Observation :

Hence actualPin local variable has been destroyed after the function call and function returned.

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

Ans : A return value is the value that a function call evaluates to. Like any other value, a return value can be used as part of an expression.

In the above example, I used the return value in an expression to match the pin to actual pin for unlocking the mobile screen.

```
def screenLockCode():
  actualPin=1442
  return actualPin

def unlockMobile(pin):
  screenPin = pin
  actualPin = screenLockCode() - 10 # return value used in an expression
  print(f"location of actual pin is : {id(actualPin)}")
  if(actualPin == screenPin):
    print("Mobile Phone unlocked successfully ...")
  else:
    print("Invalid Pin has been entered !!")

unlockMobile(1432)
```

In the unlockMobile() fuction pin entered is actally 10 less than the return value of screenLockCode() function but the logic is written in such a way that, it open the mobile phone if actualPin is 10 less than the returned actualPin.

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

Ans : If there is no return value in a function than it's return value is `None` .

```
def fuctionReturningNone():
  pass



def fuctionReturningNone():
  pass #

print(fuctionReturningNone())
```

```
None
```

Observation :

1. When there is no return value and no pass keyword it gives error.

2. When we use pass keyword the fuction doesn't required any return value or any statement.
3. when we print the function the return value is `None` .

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

Ans : A global statement will force a variable in a function to refer to the global variable. If you want to refer to a global variable in a function, you can use the `global` keyword to declare which variables are global.

```
#Example
marksObtained = 90 #global variable

def reportCard():
   global marksObtained # local variable declared as global variable using "global" keyword a
   marksObtained += 5
reportCard()

print(marksObtained) # we can print it outside the fuction without function call
```

```
95
```

Ques 10: What is the data type of None?

Ans : Let's find out the datatype of None using **type()** function.

```
type(None)
```

```
NoneType
```

Observation :

Hence, the data type of None is `NoneType` .

Ques 11: What does the sentence `import areallyourpetsnamederic` do?

Ans : The `import keyword` is use to **import the module or file** inside our python project. Hence if this module or file is existed than it will import that module otherwise shows an error.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
import sys
sys.path.insert(0,'/content/drive/MyDrive')

import areallyourpetsnamederic


areallyourpetsnamederic.__dict__
```

```
        'ascii': <function ascii>,
        'bin': <function bin>,
        'bool': bool,
        'breakpoint': <function breakpoint>,
        'bytearray': bytearray,
        'bytes': bytes,
        'callable': <function callable>,
        'chr': <function chr>,
        'classmethod': classmethod,
        'compile': <function compile>,
        'complex': complex,
        'copyright': Copyright (c) 2001-2022 Python Software Foundation.
        All Rights Reserved.

        Copyright (c) 2000 BeOpen.com.
        All Rights Reserved.

        Copyright (c) 1995-2001 Corporation for National Research Initiatives.
        All Rights Reserved.


        Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
        All Rights Reserved.,
        'credits':     Thanks to CWI, CNRI, BeOpen.com, Zope Corporation and a cast of thou
            for supporting Python development.  See www.python.org for more information.,
        'delattr': <function delattr>,
        'dict': dict,
        'dir': <function dir>,
        'display': <function IPython.core.display.display>,
        'divmod': <function divmod>,
        'dreload': <function IPython.lib.deepreload._dreload>,
        'enumerate': enumerate,
        'eval': <function eval>,
        'exec': <function exec>,
        'execfile': <function _pydev_imps._pydev_execfile.execfile>,
        'filter': filter,
        'float': float,
        'format': <function format>,
        'frozenset': frozenset,
        'get_ipython': <bound method InteractiveShell.get_ipython of <google.colab._shell.S
        'getattr': <function getattr>,
        'globals': <function globals>,
        'hasattr': <function hasattr>,
        'hash': <function hash>,
        'help': Type help() for interactive help, or help(object) for help about object.,
        'hex': <function hex>,
        'id': <function id>,
        'input': <bound method Kernel.raw_input of <google.colab._kernel.Kernel object at 0
```

```
'int': int,
'isinstance': <function isinstance>,
'issubclass': <function issubclass>,
'iter': <function iter>,
'len': <function len>,
'license': Type license() to see the full license text,
'list': list,
'locals': <function locals>,
'map': map,
'max': <function max>,
```

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

Ans : After import the spam module or any module we can call it's function using:

```
import moduleName

moduleName.moduleFunctionName()
```

So, here we call the bacon() as **spam.bacon()**

```
import spam

spam.bacon()
```

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

Ans : Whatever line of code that might create an error or might cause any exception we handled that error using try and except clause.

try clause throw that error object to except block and except performed the programmer defined code.

So the error never gonna happened hence we save the program from crashing.

```
#Example without try & except clause
def div(num1, num2):
        res = num1 // num2
        print(f"Answer :  {res}")

div(3, 0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-17-7033ade216ee> in <module>()
      4                 print(f"Answer :  {res}")
      5
----> 6 div(3, 0)

<ipython-input-17-7033ade216ee> in div(num1, num2)
      1 #Example without try & except clause
      2 def div(num1, num2):
----> 3                 res = num1 // num2
      4                 print(f"Answer :  {res}")
      5

ZeroDivisionError: integer division or modulo by zero
```

```
#Example with try & except clause
def div(num1, num2):
    try:
        res = num1 // num2
        print(f"Answer :  {res}")
    except ZeroDivisionError:
        print("Invalid inputs we cannot divide number by zero ")

div(3, 0)
```

```
Invalid inputs we cannot divide number by zero
```

Ques 14: What is the purpose of the try clause? What is the purpose of the except clause?

Ans : `try clause` helps to identify the line of code which is potentially can cause an error . code inside the `try clause` executes and whenever there is an error happens then `except clause` catches that error object and execute the handling code or programmer code for handling that particular error.

✓ 0s completed at 4:09 PM