Q1. What is an Exception?

An Exception is an event that disrupts the normal flow of the program. An exception that occurs during the execution of a Python **program is an object** that is an instance of a class.

Q2 Name some classes in Python that represent exceptions.

NameError, TypeError, ValueError, IndexError, ZeroDivisionError **are all classes in Python** that represent an exception.

Q3 How can we refer to argument(s) inside an exception object?

We can refer to them by **assigning the object to an identifier using the as keyword**. For example:

```
except TypeError as aName  #aName is the identifier
    print(aName.args)
except NameError as someName #someName is the identifier
   print(someName.args)
```

Q4 What is the type of args?

`args` is of type tuple

Q5 What is the Exception class? (Please note that word Exception here refers to the name of a class)

The Exception class is a base class that enables a programmer to create their own instance of an exception with arguments.Here is an example of how we are using the Exception class to create an instance of an Exception object in the Stack class.

How to raise an Exception in Stack class	How to catch an Exception

`class Stack:`
`    # we are going to use a list`
`    def __init__(self):`
`        # initialize the Stack object`
`        self.items = []`
`    def push(self, item):`
`        # insert item at the end of the list`
`        self.items.append(item) # O(1)`
`    def pop(self):`
`        # remove and return item at the end of the list`
`        try:`
`            return self.items.pop() # O(1)`
`        except:`
`            raise Exception('Stack is Empty!!!!!')`
`    def peek(self):`
`        # return the item located at the last location`
`        return self.items[len(self.items) -1]  # O(1)`
	`from Stack2 import Stack`
	
	`def main():`
	`    aStack = Stack()`
	`    aStack.push('Fred')`
	`    get_name(aStack)`
	`    get_name(aStack)`
	
	`def get_name(bStack):`
	`    try:`
	`        print(bStack.pop())`
	`    except Exception as e:`
	`        print(e.args)`
	`main()`

```
    def isEmpty(self):
        # return True if Stack is empty
        # False otherwise
        return self.items == []
    def size(self):
        # returns the number of items in the Stack object
        return len(self.items)
    def reset(self):
        self.items = []
    def __str__(self):
        # return the string representation of the object
        return str(self.items)
    def show(self):
        # shows the string representation of the Stack
object
        print(self) # calls __str__ automatically
```