1. Observe the types of errors the following code can possibly generate. Now handle the errors suitably by enclosing calls to the function inside a try block.

```
def fun(a):
  if a < 4:
    b = a/(a-3)

  print("Value of b = ", b)
```

2. Create a subclass Error of the ErrorType Exception. Now create subclasses ZError and RError of Error to be raised when the value entered by a the user (as in the following statement) is zero and negative respectively:
num = int(input("Enter a number: "))

   Also, in case the value entered cannot be converted to an integer, appropriate handling of the error generated needs to be done.

3. What would be the output of the following lines of code?

```
class MyError(Exception):

  # Constructor or Initializer
  def __init__(self, value):
    self.value = value

  # __str__ is to print() the value
  def __str__(self):
    return(repr(self.value))


try:
  raise(MyError(3*2))

# Value of Exception is stored in error
except MyError as error:
```

```
print('A New Exception occurred: ', error.value)
```

Now comment out the only line of code inside the initializer and replace it with pass. Try to have the value in the following statement printed.

```
print('A New Exception occurred: ', error.value)
```