Answers for Debugging Exercises: Chapter 12

Find the Output

1.
```
>>> raise NameError('var')
```

Ans. NameError: var

2.
```
try:
      raise TypeError('int Expected')
except TypeError:
      raise
```

Ans. TypeError: int Expected

3.
```
try:
   file = open("File.txt", "r")
   file.write("Hello World")
except IOError:
   print("Error writing to file.......")
else:
      print("Write Operation Successful...........")
```

Ans.
```
Error writing to file.......
```

4.
```
try:
   file = open("File", "r")
try:
   file.write("This is my test file for exception handling!!")
finally:
```

```
    print("Closing the file.....")
    file.close()
except IOError:
    print("Error: file not found .......")
```

Ans. Error: file not found ....

5.

```
def convert(var):
   try:
      return int(var)
   except ValueError as e:
      print(e.args)
convert("xyz")
```

Ans. ("invalid literal for int() with base 10: 'xyz'",)

6.

```
List = ['a', 0, 2]
for i in List:
    try:
        print(i),
        r = 1/int(i)
        break
    except:
        print("Error ....")
```

Ans.

a Error ....

0 Error ....

7.

```
>>> raise MemoryError("Problem dealing with memory....")
```

Ans. MemoryError: Problem dealing with memory....

8.
```
while 1:
    try:
        n = int(input("Enter an integer: "))
        break
    except ValueError:
        print("Enter again ...")
    else:
       print("Congratulations... number accepted....")
```

Ans.

```
Enter an integer: a
Enter again ...
Enter an integer: 10
```

9.
```
try:
 file = open('Integers.txt')
 num = int(file.readline())
except (IOError, ValueError):
    print("I/O error or a ValueError occurred")
except:
    print("An unexpected error occurred")
    raise
```

Ans. I/O error or a ValueError occurred

10.
```
 def func(i):
   List = [1,2,3]
   try:
      assert i >= 1
      return l[i]
```

```
        except TypeError,e:
            print("Dealing with TypeError")
        except IndexError, e:
            print("Dealing with IndexError")
        except:
            print("Any other error...")
        finally:
            print("Terminating the program .....")
    func(-1)
```

Ans.

```
Any other error...
Terminating the program .....
```

11.

```
error = Exception("Raising my error...")
raise error
```

Ans. Exception: Raising my error...

12.

```
def listen(name):
    raise Exception(name + " you have generated an error...")
    listen("Henry")
```

Ans. Exception: Henry you have generated an error...

13.

```
try:
    var = 10
    print(var)
    raise NameError("Hello")
except NameError as e:
```

```
        print("Error occurred......")
        print(e)
```

Ans.

```
10
Error occurred......
Hello
```

14.

```
class Error(Exception):
    def __init__(self, num):
        self.num = num
    def __str__(self):
        return repr(self.num)
try:
    raise Error(420)
except Error as e:
    print("Received error:", e.num)
```

Ans. Received error: 420

15.

```
str="123"
raise NameError("String please…!")
```

Ans. NameError: String please…!

Find the Error

1.

```
try:
    file = open('File1.txt')
    str = f.readline()
    print(str)
except ValueError:
```

```
        print("Error occurred ..... Program Terminating...")
    else:
        print("Program Terminating Successfully.....")
```

Ans. NameError: name 'f' is not defined

2.
```
try:
   raise KeyboardInterrupt
finally:
   print('Good Morning')
```

Ans. KeyboardInterrupt

3.
```
def divide(x, y):
   try:
      result = x / y
   except ZeroDivisionError:
      print("Division by zero!")
   else:
      print("result is", result)
   finally:
      print("executing finally clause")
divide('x', 1)
```

Ans. TypeError: unsupported operand type(s) for /: 'str' and 'int'

4.
```
def KelvinToFahrenheit(Temp):
   assert (Temp >= 0),"Freezing"
   return ((Temp -273)*1.8)+32
print(KelvinToFahrenheit(-5))
```

Ans. AssertionError: Freezing

5.
```
try:
   file = open("File.txt", "r")
```

```
        file.write("Hello World")
    finally:
        print("Error writing to file.......")
```

Ans. IOError: File not open for writing

6.
```
try:
    x = float(input("Enter the number: "))
    inverse = 1.0 / x
finally:
    print("Thank you ....")
print("The inverse: ", inverse)
```

Ans.

```
Enter the number: 0
Thank you ....
ZeroDivisionError: float division by zero
```

7.
```
try:
    x = float(input("Enter the number: "))
    inverse = 1.0 / x
except ValueError:
    print("Number means an int or a float")
except ZeroDivisionError:
    print("Infinity.......")
finally:
    print("Thank you ....")
print("The inverse: ", inverse)
```

Ans.

```
Enter the number: 0
Infinity.......
Thank you ....
The inverse:
NameError: name 'inverse' is not defined
```

8. `>>> print(var)`

Ans. NameError: name 'var' is not defined

9. `>>> 10 + 'a'`

Ans. TypeError: unsupported operand type(s) for +: 'int' and 'str'

10.
```
Dict = {"One":1, "Two":2}
print(Dict["Three"])
```

Ans. KeyError: 'Three'

11.
```
List = [1,2,3,4,5]
print(List[5])
```

Ans. IndexError: list index out of range

12.
```
List = [1,2,3,4,5]
print(List.join(100))
```

Ans. AttributeError: 'list' object has no attribute 'join'

13.
```
List = [1,2,3,4,5]
print(List['one'])
```

Ans. TypeError: list indices must be integers, not str

14.
```
Tup = ('abc', 'def', 'xyz', 'jkl')
Tup[2] = 'ghi'
```

Ans. TypeError: 'tuple' object does not support item assignment

15.
```
def func1(i):
```

```
    return i / 0
def func2():
    raise Exception("Raising Exception ......")
def func3():
    try:
        func1(5)
    except Exception as e:
        print(e)
        raise
    try:
        func2()
    except Exception as e:
        print(e)
func3()
```

Ans. ZeroDivisionError: integer division or modulo by zero