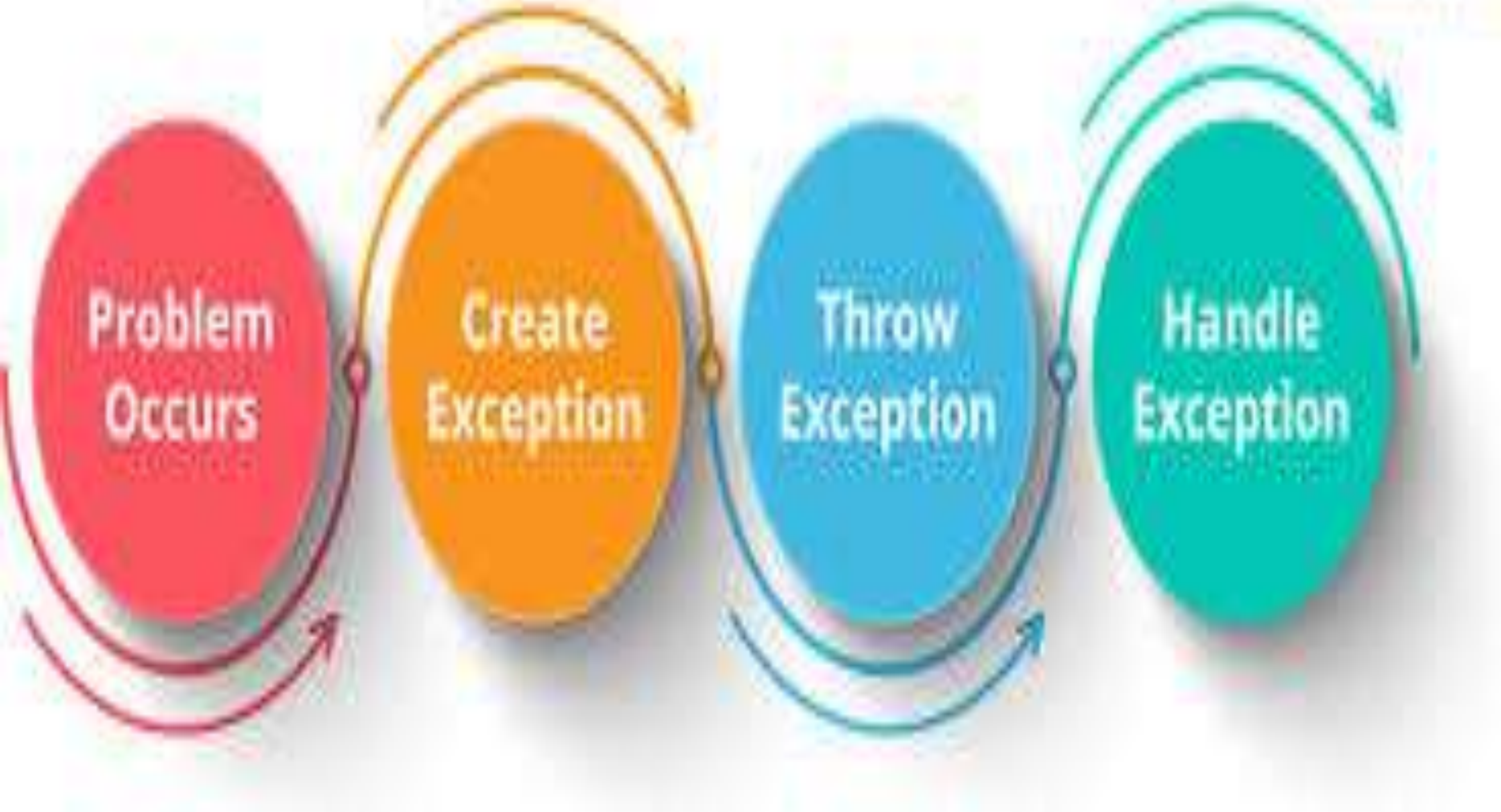
EXCEPTIONAL HANDLING



ERRORS

Error is a abnormal condition whenever it occurs execution of the program is stopped.

An error is a term used to describe any issue that arises unexpectedly that cause a computer to not function properly. Computers can encounter either software errors or hardware errors.

ERRORS

Errors are mainly classified into following types.

1. Syntax Errors
2. Semantic Errors
3. Run Time Errors.
4. Logical Errors.

1. Syntax Errors

Syntax errors refer to formal rules governing the construction of valid statements in a language.

Syntax errors occur when rules of a programming language are misused i.e., when a grammatical rule of the language is violated.

1. Syntax Errors

For instance in the following program segment,

```
def main()
    a=10
    b=3
    print(" Sum is ",a+b
```

: (Colon) Missing

) Missing

2. Semantic Errors

Semantics error occur when statements are not meaningful

Semantics refers to the set of rules which give the meaning of the statement.

For example,

Rama plays Guitar

This statement is syntactically and semantically correct and it has some meaning.

2. Semantic Errors

See the following statement,

Guitar plays Rama

is syntactically correct (syntax is correct) but semantically incorrect. Similarly, there are semantics rules of programming language, violation of which results in semantical errors.

X * Y = Z

will result in semantical error as an expression can not come on the left side of an assignment statement.

3. Run Time Errors.

A Run time error is that occurs during execution of the program. It is caused because of some illegal operation taking place.

For example

1. If a program is trying to open a file which does not exists or it could not be opened(meaning file is corrupted), results into an execution error.

2. An expression is trying to divide a number by zero are RUN TIME ERRORS.

4. Logical Errors.

- A Logical Error is that error which is causes a program to produce incorrect or undesired output.

  for instance,

```
ctr=1;
while(ctr<10):
      print(n *ctr)
```

Exceptions

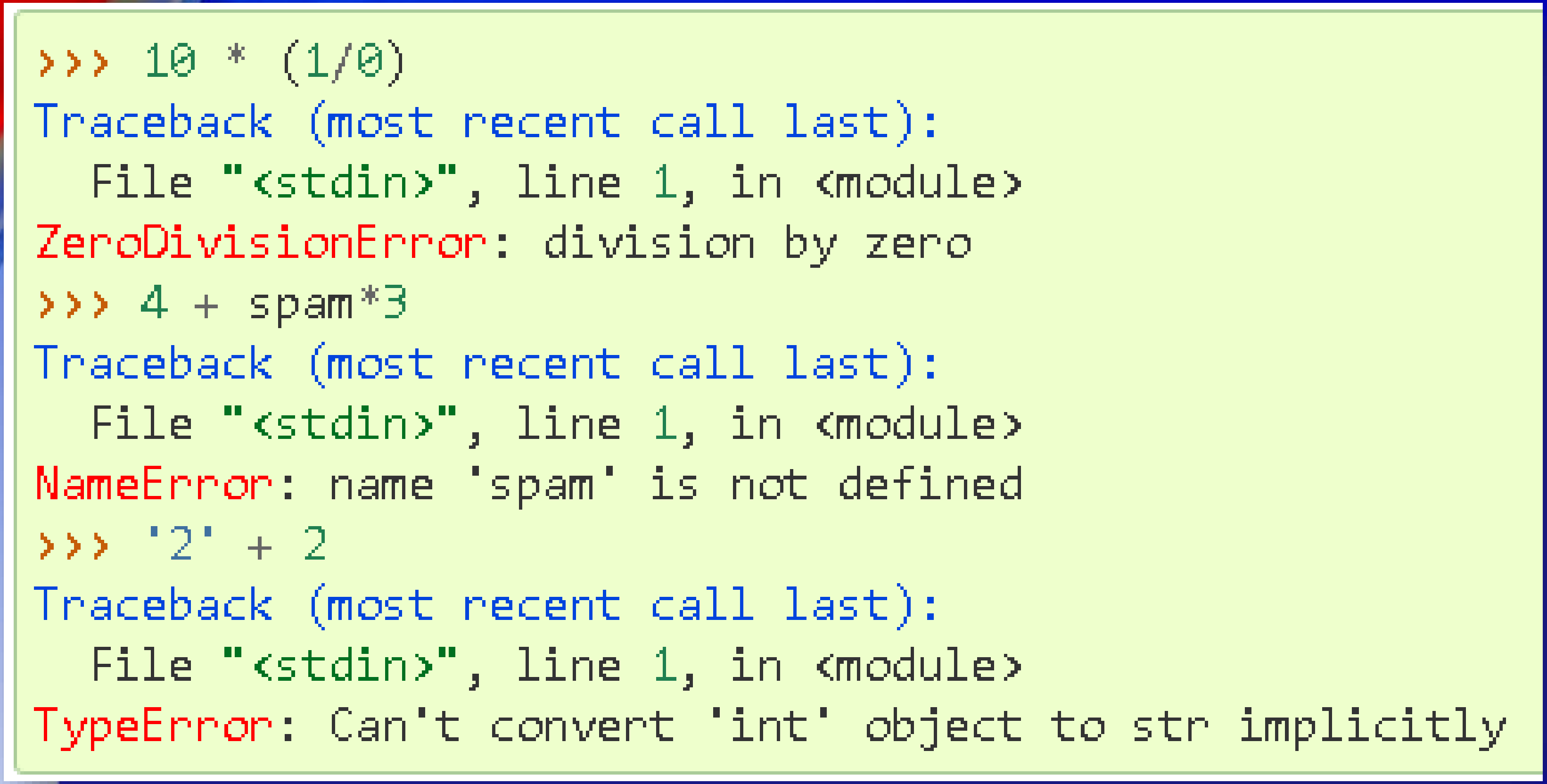
Exceptions

What is an exception?

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called *exceptions*

Exceptions

For Example



```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Handling Exceptions

Handling Exceptions

It is possible to write programs that handle selected exceptions. Look at the following example, which asks the user for input until a valid integer has been entered, but allows the user to interrupt the program (using Control-C or whatever the operating system supports); note that a user-generated interruption is signalled by raising the KeyboardInterrupt exception.

Handling Exceptions

For Example

```
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     except ValueError:
...         print("Oops!  That was no valid number.  Try again...")
...
```

Handling Exceptions

The try statement works as follows.

- ✓ **First, the try clause (the statement(s) between the try and except keywords) is executed.**

- ✓ **If no exception occurs, the except clause is skipped and execution of the try statement is finished.**

Handling Exceptions

The try statement works as follows.

✓If an exception occurs during execution of the try clause, the rest of the clause is skipped. Then if its type matches the exception named after the except keyword, the except clause is executed, and then execution continues after the try statement.

Handling Exceptions

The try statement works as follows.

- ✓ If an exception occurs which does not match the exception named in the except clause, it is passed on to outer try statements; if no handler is found, it is an unhandled exception and execution stops with a message as shown above.

The *except* Clause with No Exceptions

The *except* Clause with No Exceptions

You can also use the except statement with no exceptions defined as follows:

```
try:
    You do your operations here;
except:
    If there is any exception, then
    execute this block.....
else:
    If there is no exception then execute
    this block.
```

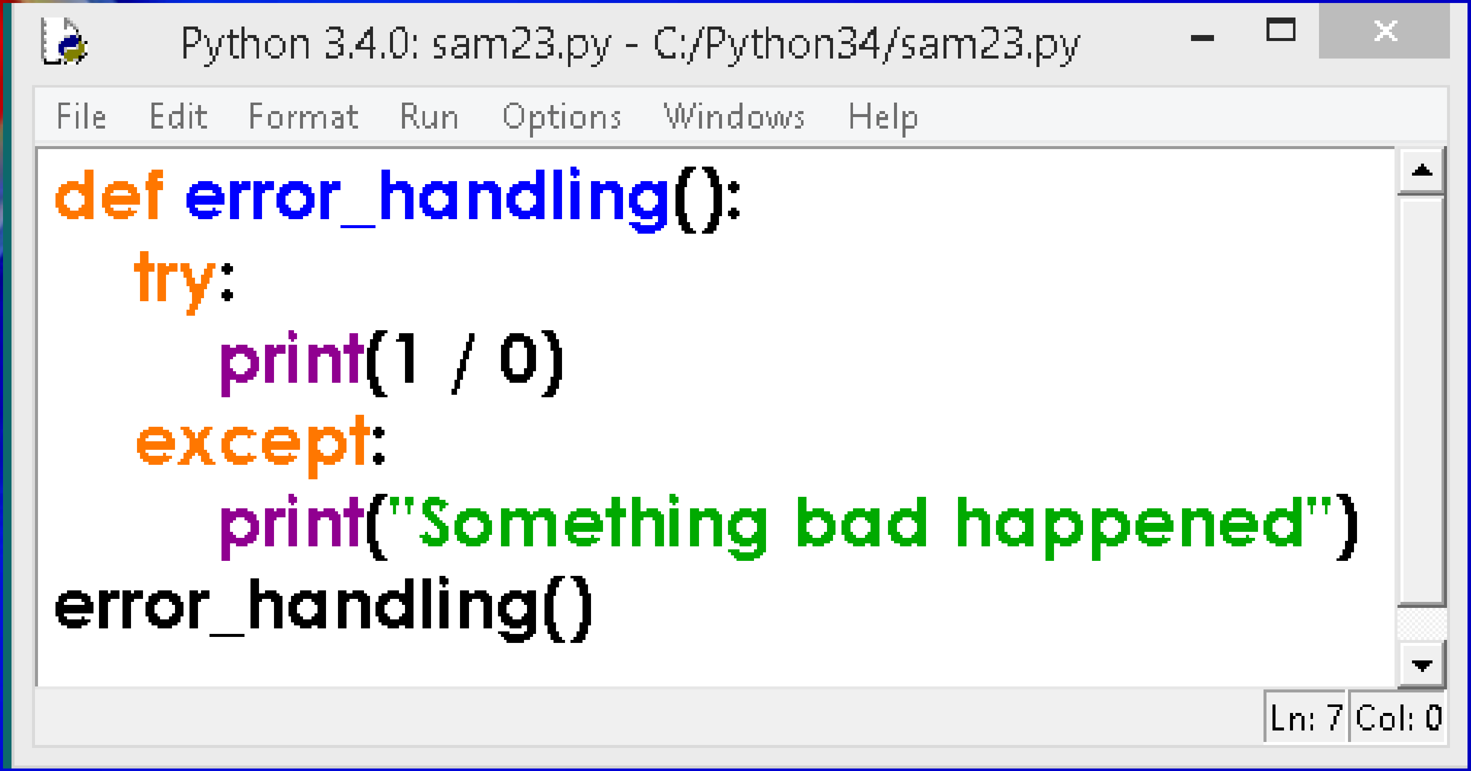
Contd..

The *except* Clause with No Exceptions

This kind of a try-except statement catches all the exceptions that occur. Using this kind of try-except statement is not considered a good programming practice though, because it catches all exceptions but does not make the programmer identify the root cause of the problem that may occur.

Example

Example


Python 3.4.0: sam23.py - C:/Python34/sam23.py
File Edit Format Run Options Windows Help
Ln: 7 Col: 0

```python
def error_handling():
    try:
        print(1 / 0)
    except:
        print("Something bad happened")
error_handling()
```

raise statement

raise statement

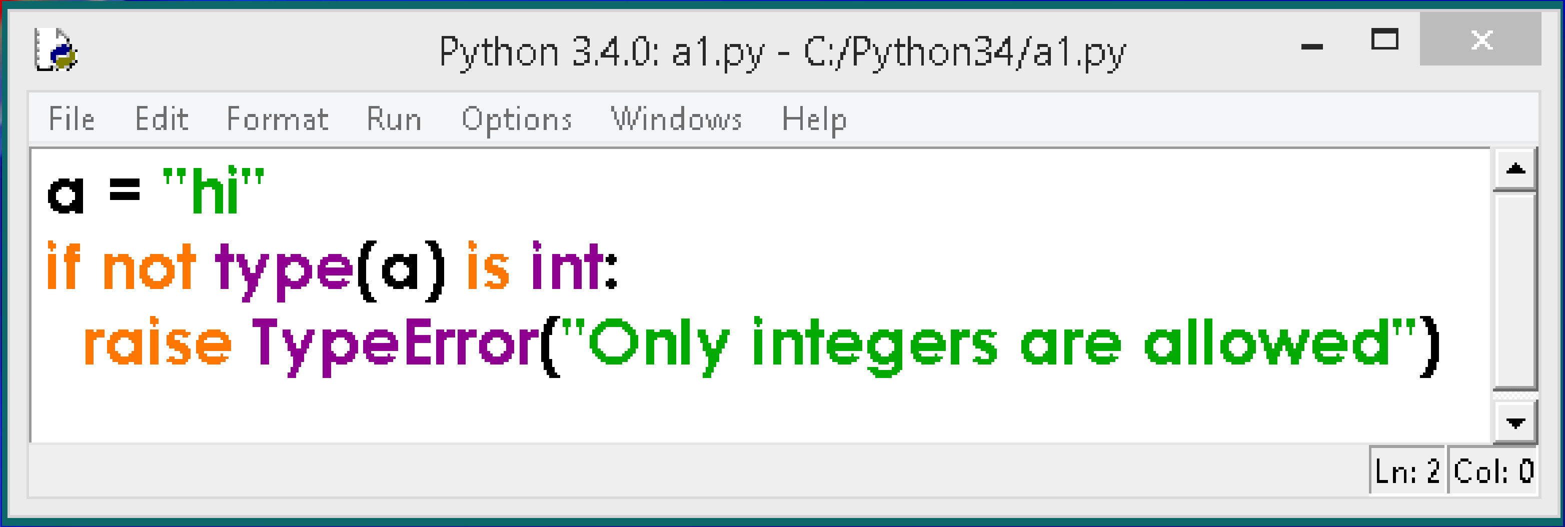
You can raise an exception in your own program by using the raise exception.

Raising an exception breaks current code execution and returns the exception back until it is handled.

Syntax:

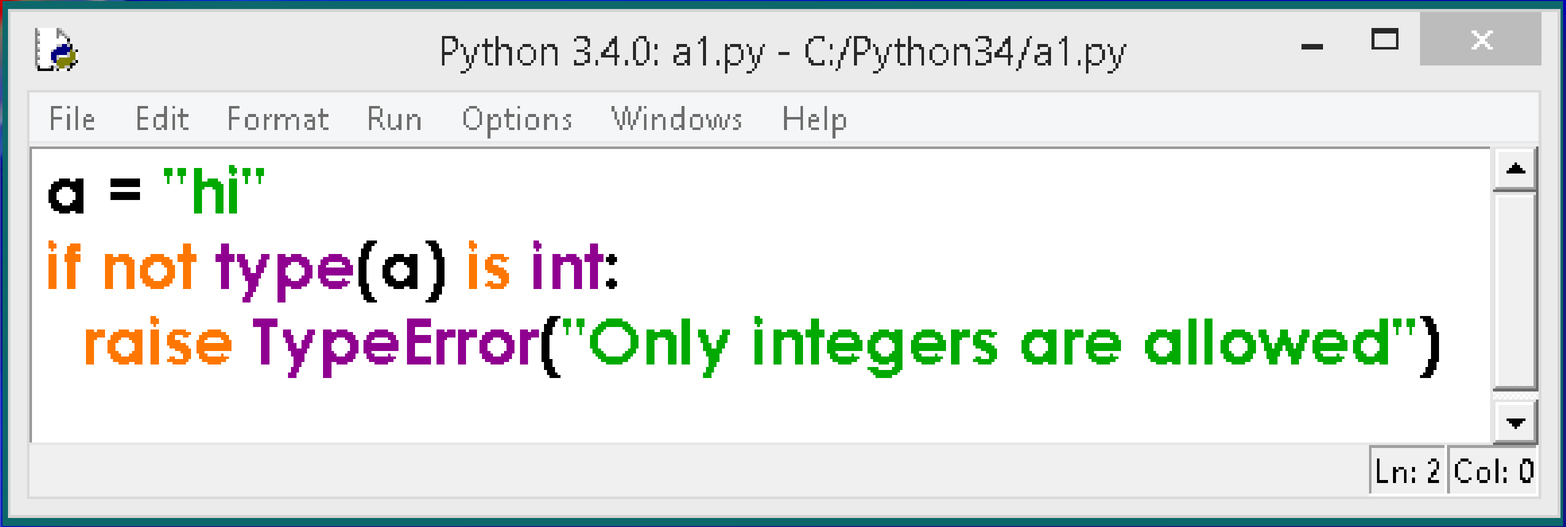
raise [expression1[, expression2]]

raise Example


Python 3.4.0: a1.py - C:/Python34/a1.py
File Edit Format Run Options Windows Help
Ln: 2 Col: 0

```
a = "hi"
if not type(a) is int:
  raise TypeError("Only integers are allowed")
```

raise Example


Python 3.4.0: a1.py - C:/Python34/a1.py
File Edit Format Run Options Windows Help
Ln: 2 Col: 0

```python
a = "hi"
if not type(a) is int:
  raise TypeError("Only integers are allowed")
```

K.V.Sesha Sain [Kasyap]

Corporate Trainer| Technical Trainer|

Company Specific Trainer |Certified Skilled Trainer

M.sc Comp, OCP, JCP, MDSE

Oracle Certified Professional (2) |

Java Certified Professional

16+ Years of Experience

Email:brainsncrowns@gmail.com

Mobile: 6300340023