PYTHON PROGRAMMING by SIVA JASTHI

Course	Python 101
Term	
Week	
Date	
Chapter. Topic	

Errors and Exceptions

Siva R Jasthi

Computer Science and Cybersecurity

Metropolitan State University

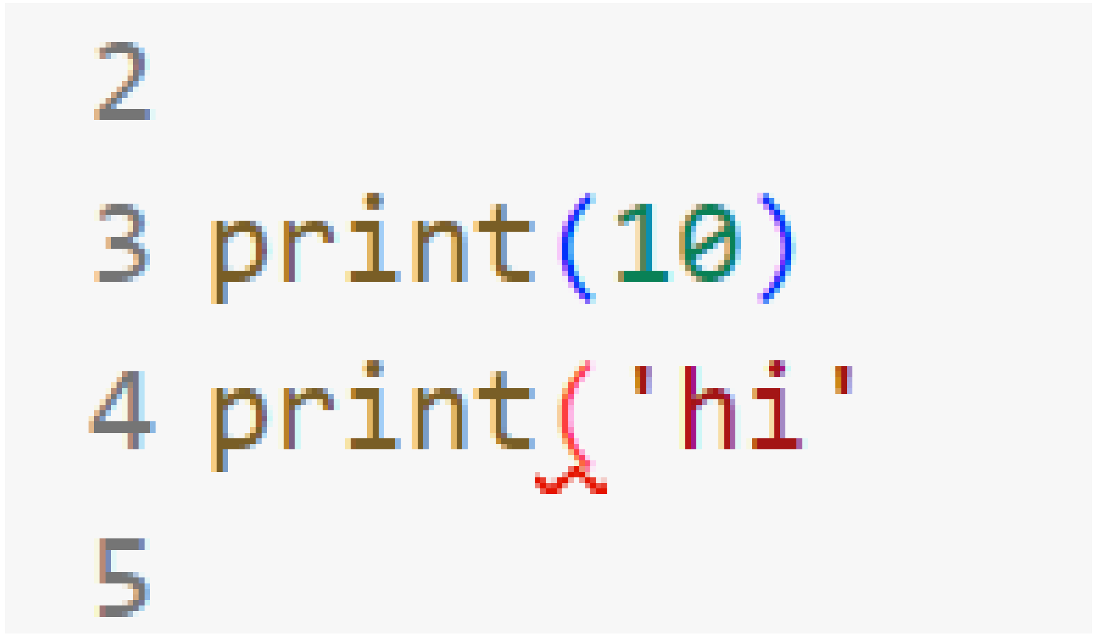
PYTHON PROGRAMMING by SIVA JASTHI

Outline

- Errors
  - Syntax Errors
  - Logical Errors
  - Runtime Errors (Exceptions)

- Exception Handling

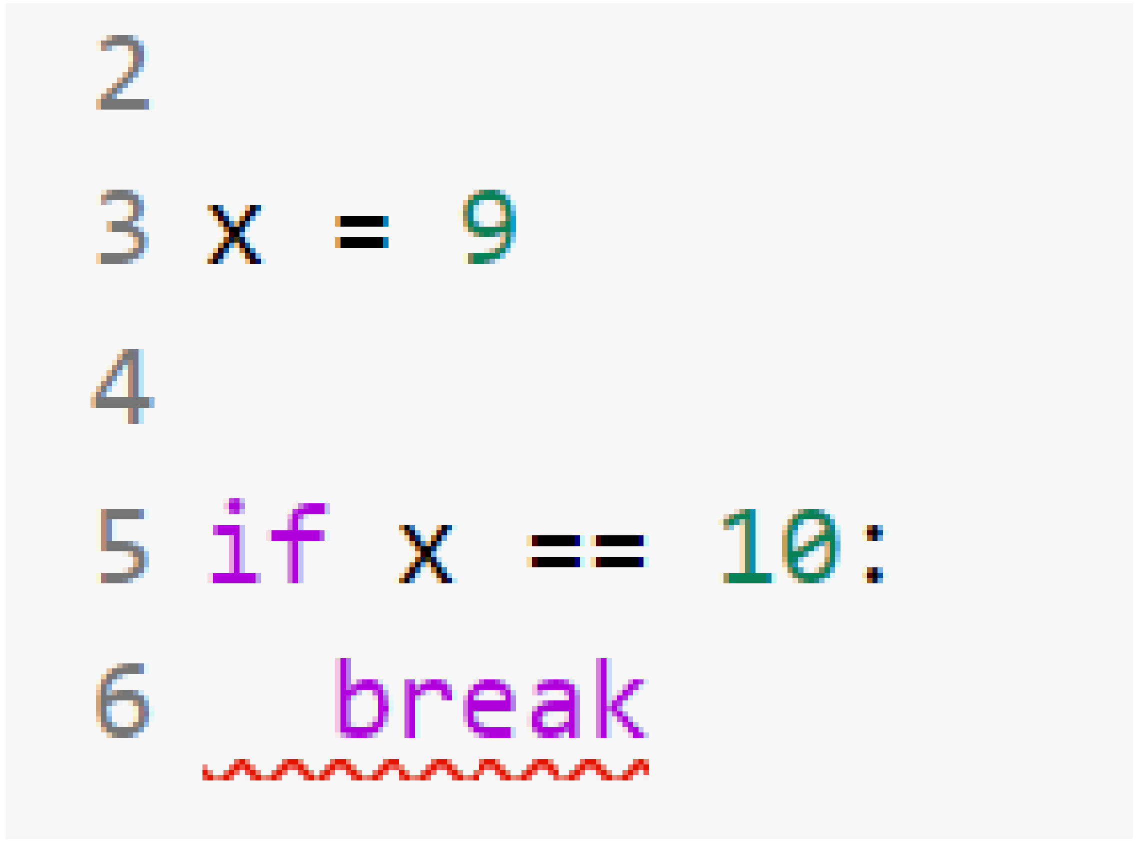
What is wrong with this code?



```python

print(10)
print('hi'

```

Syntax error

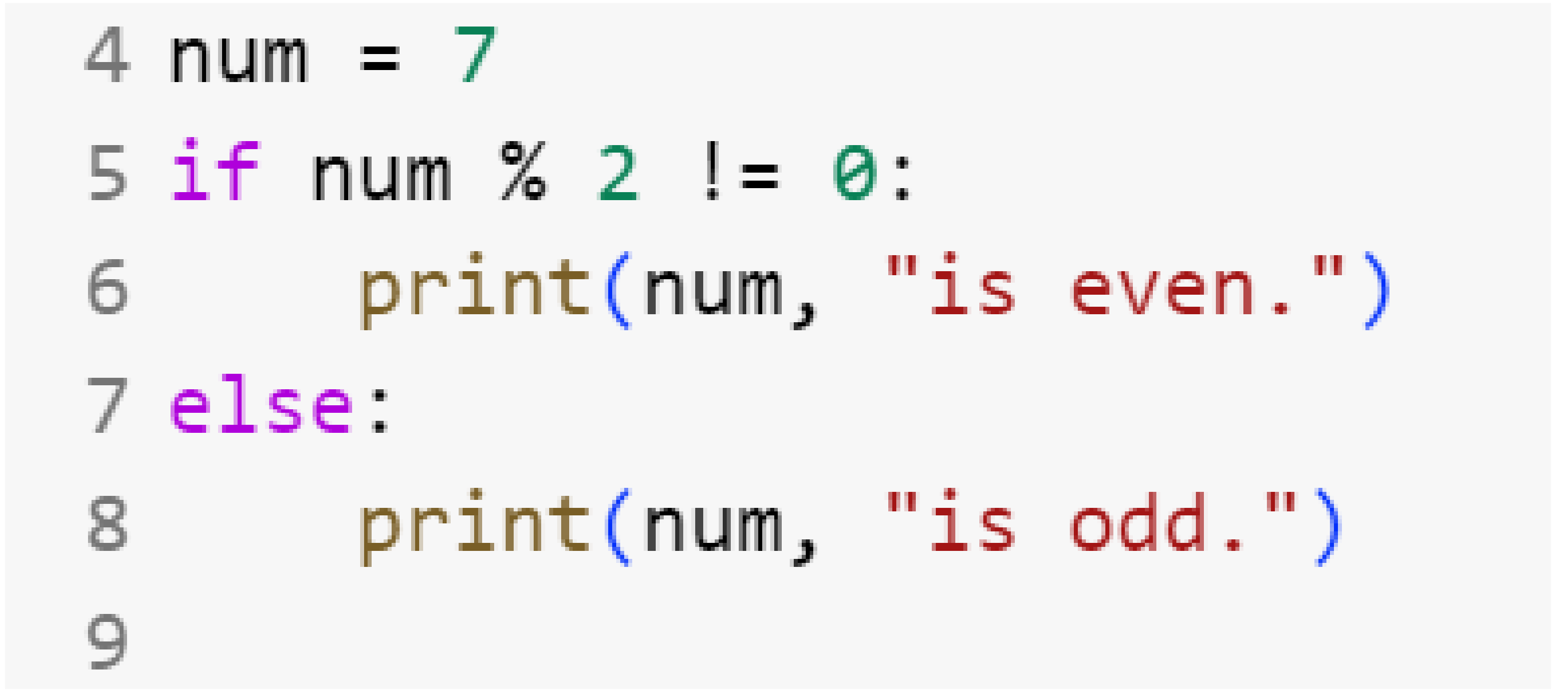
What is wrong with this code?



```

x = 9

if x == 10:
    break
```

Syntax error

What is wrong with this code?

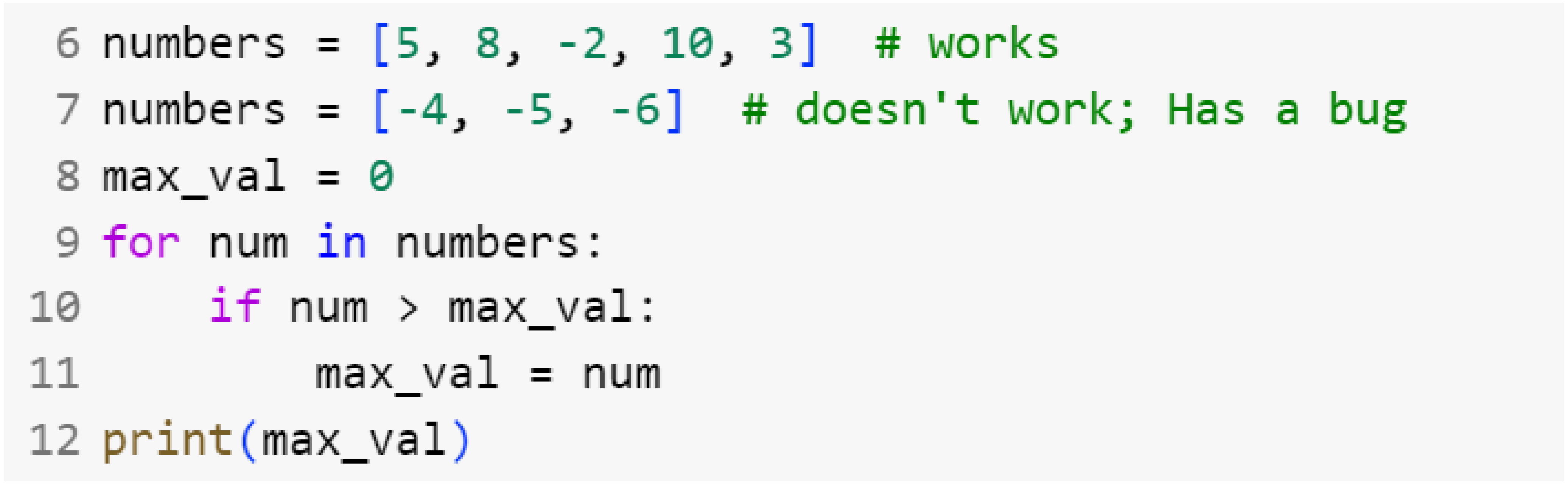


```python
num = 7
if num % 2 != 0:
    print(num, "is even.")
else:
    print(num, "is odd.")

```

```
Num = 7
If num %2 != 0
  print(num, “is even”)

Else:
 print(num, “is odd”)
```

PYTHON PROGRAMMING by SIVA JASTHI

What is wrong with this code?

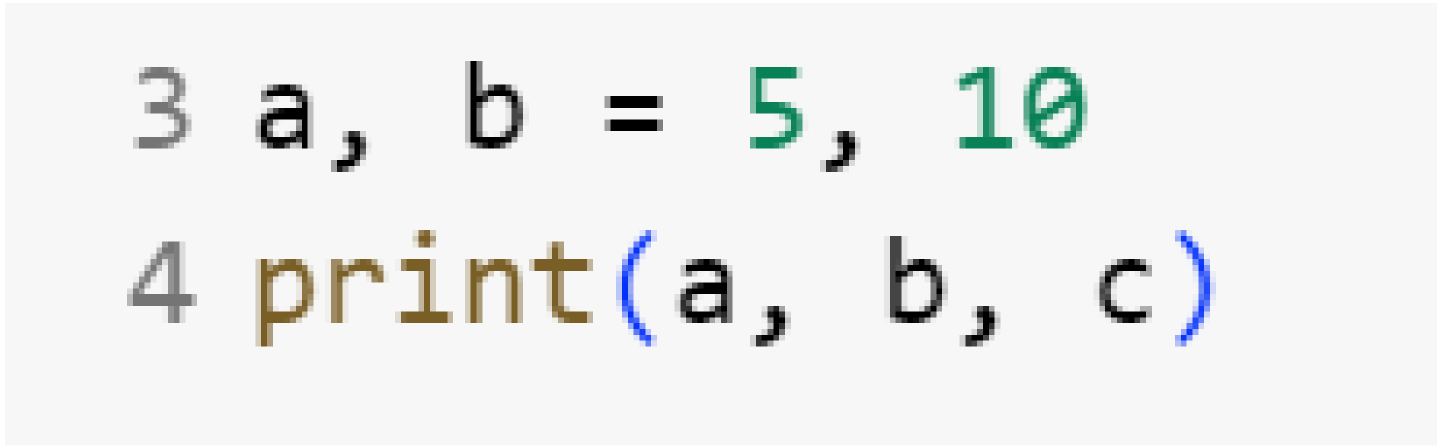


```python
numbers = [5, 8, -2, 10, 3]  # works
numbers = [-4, -5, -6]  # doesn't work; Has a bug
max_val = 0
for num in numbers:
    if num > max_val:
        max_val = num
print(max_val)
```

Version 1

What is wrong with this code?

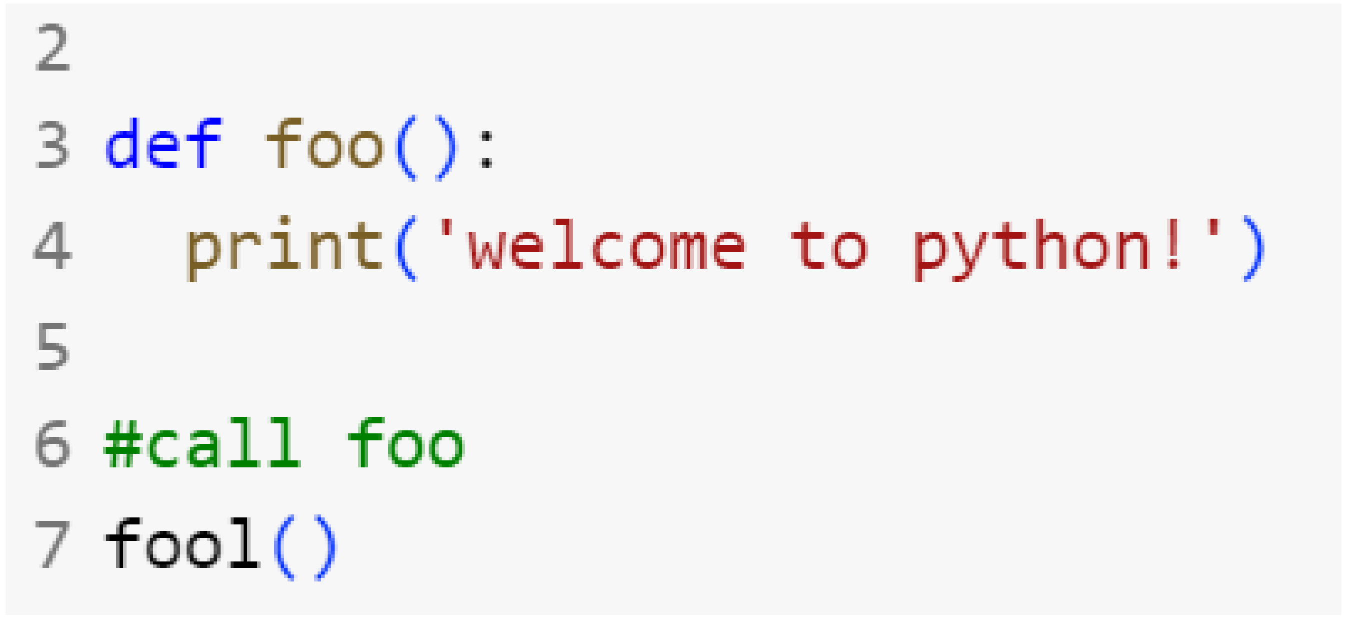
PYTHON PROGRAMMING by SIVA JASTHI



```
a, b = 5, 10
print(a, b, c)
```

Name error

What is wrong with this code?



```python
def foo():
  print('welcome to python!')

#call foo
fool()
```

Name error

Let us switch to google colab for examples

- ch6_2_concepts_Syntax_Logical_Runtime_Errors_n_Exceptions.ipynb

Exceptions: Overview

assert

raise

try	1
except	2
else	3
finally	4

All good in 1: → 1, 3, 4
Something has gone wrong in 1 : -→ 1, 2, 4

Try something 1
If anything goes wrong, do 2
If everything in 1 is good, do 3.
And do 4 all the times.

Exceptions

Exceptional = Excellence = Good!

Exception = Something has gone wrong. It is NOT normal.

What can go wrong when you ask the user for his/her age?

Correct input: 10

However, the following inputs are incorrect. In these cases, our program should not crash. These cases should be gracefully handled. How?

- Enter your age: A (incorrect input)
- Enter your age: 0 (incorrect input; age can not be 0)
- Enter your age: -20 (incorrect input; age can not be negative)
- Enter your age: 140 (incorrect input; no one lives for 140)

Exceptions

PYTHON PROGRAMMING by SIVA JASTHI

Exceptional = Excellence = Good!

Exception = Something has gone wrong. It is NOT normal.

What can go wrong while doing the File I/O (reading/writing)?

Here are some examples:

1. File you are opening doesn’t exist
2. File exists, but it is wrong type (it is not a text file, it is a Power Point file)
3. File exists. It is the correct type. However, it contains incorrect data (1,2,3 instead of names) or (a,b,c instead of numbers)
4. File exists. It is the correct type. It contains the correct data. However, you don’t have the permission to read it.

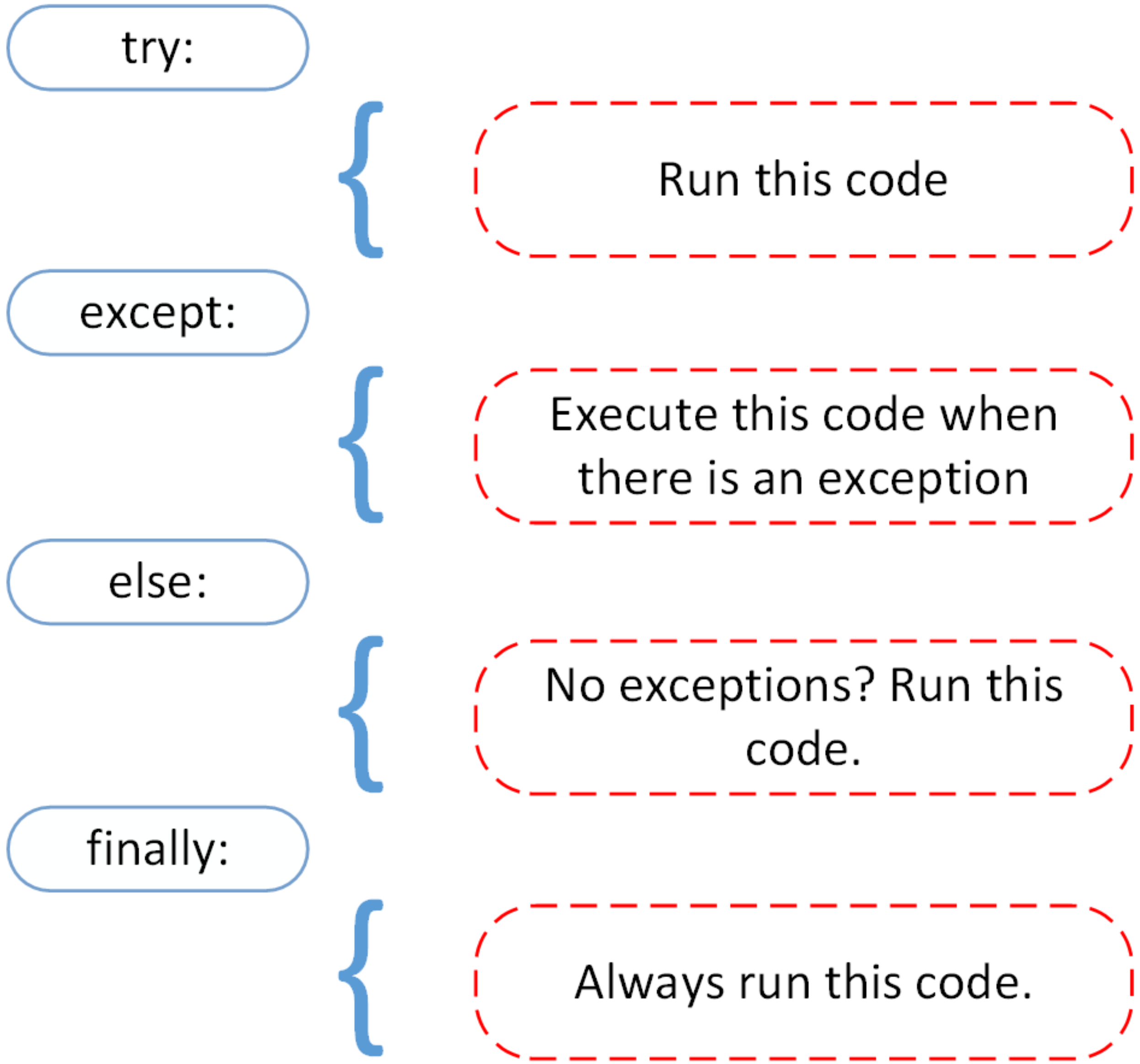
How do we handle those Exceptions?

PYTHON PROGRAMMING by SIVA JASTHI

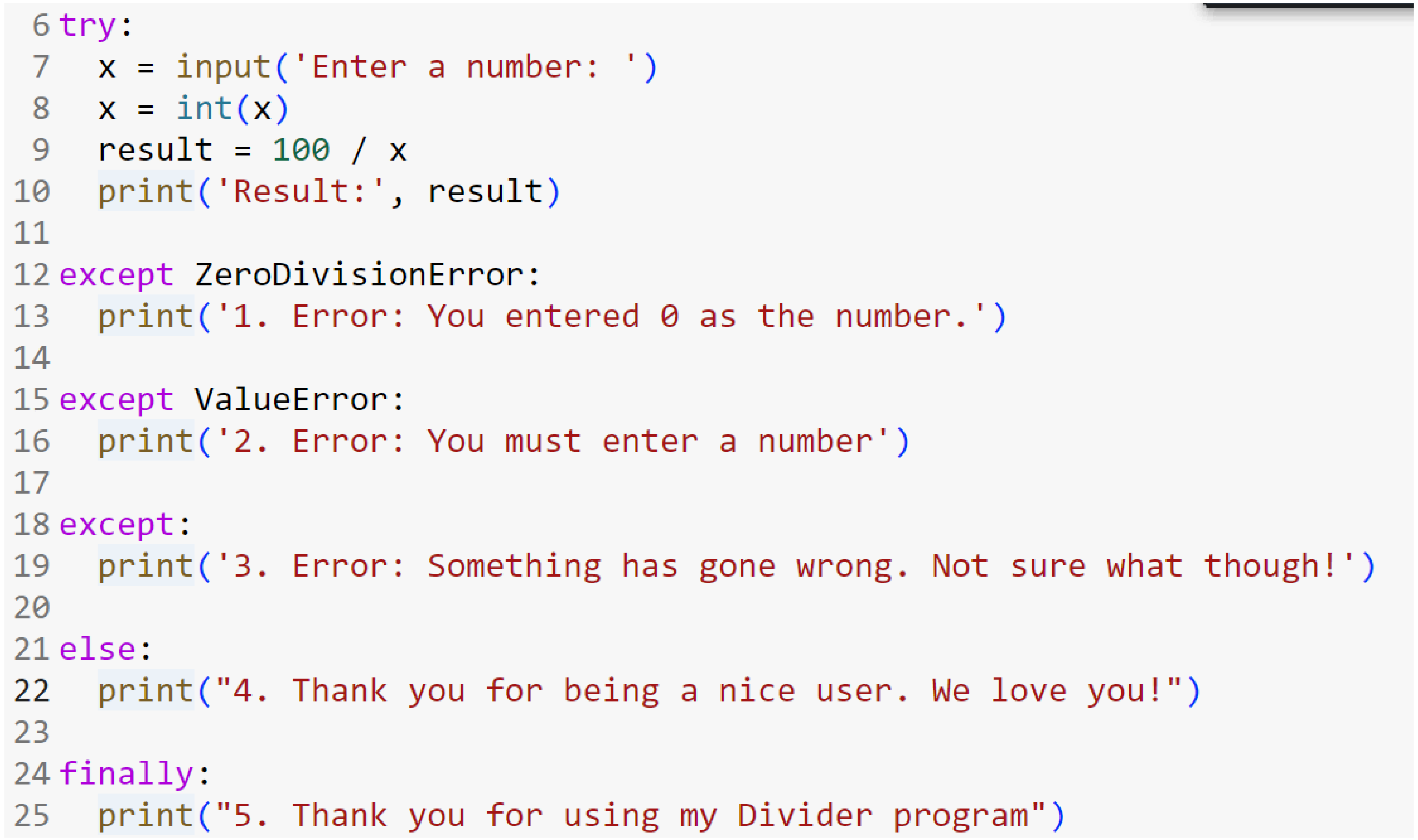
```
try:
      statement 1
      statement 2

except Exception as err:
      do something to catch the errors
```

```
try:
   statement
   statement
   etc.
except  ExceptionName:
   statement
   statement
   etc.
```

Handling Exceptions: Code Flow

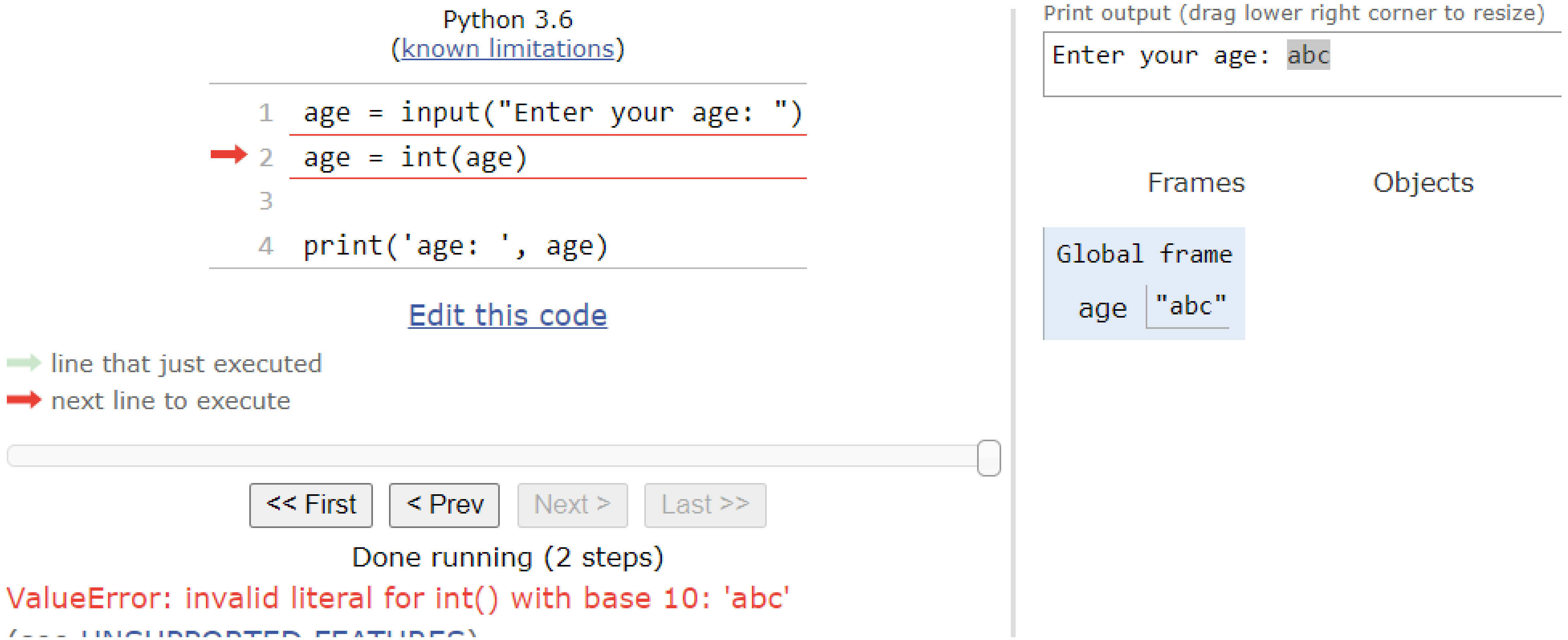

try:
Run this code
except:
Execute this code when there is an exception
else:
No exceptions? Run this code.
finally:
Always run this code.

Mind map: Example code



```python
try:
  x = input('Enter a number: ')
  x = int(x)
  result = 100 / x
  print('Result:', result)

except ZeroDivisionError:
  print('1. Error: You entered 0 as the number.')

except ValueError:
  print('2. Error: You must enter a number')

except:
  print('3. Error: Something has gone wrong. Not sure what though!')

else:
  print("4. Thank you for being a nice user. We love you!")

finally:
  print("5. Thank you for using my Divider program")
```

How do we handle those Exceptions?



Python 3.6
(known limitations)

```
age = input("Enter your age: ")
age = int(age)

print('age: ', age)
```

Edit this code

line that just executed
next line to execute

Done running (2 steps)

ValueError: invalid literal for int() with base 10: 'abc'

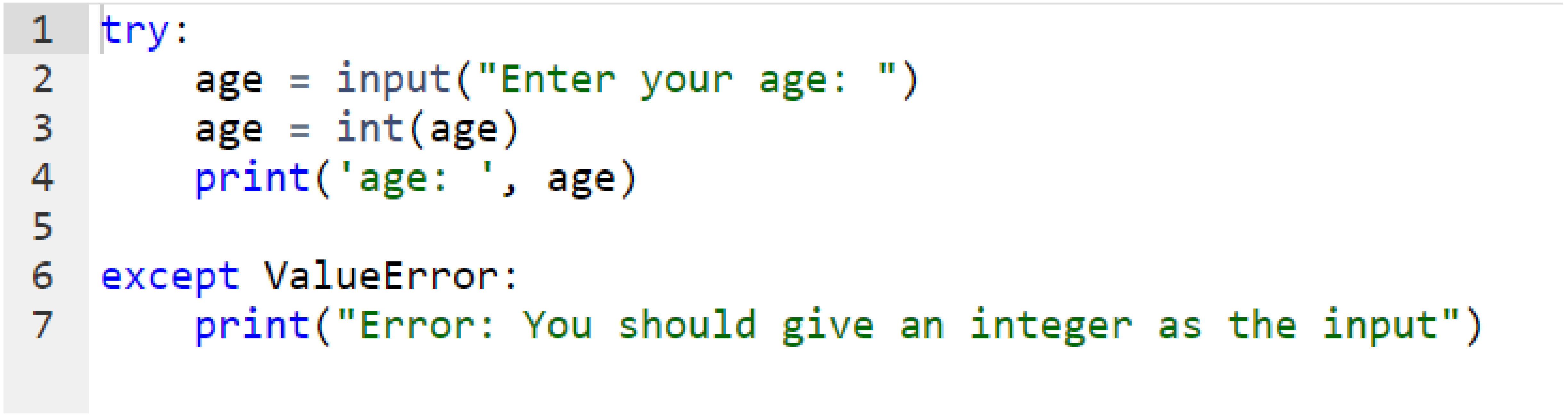
Print output (drag lower right corner to resize)

```
Enter your age: abc
```

Frames	Objects
Global frame	
age "abc"	

How do we handle those Exceptions?

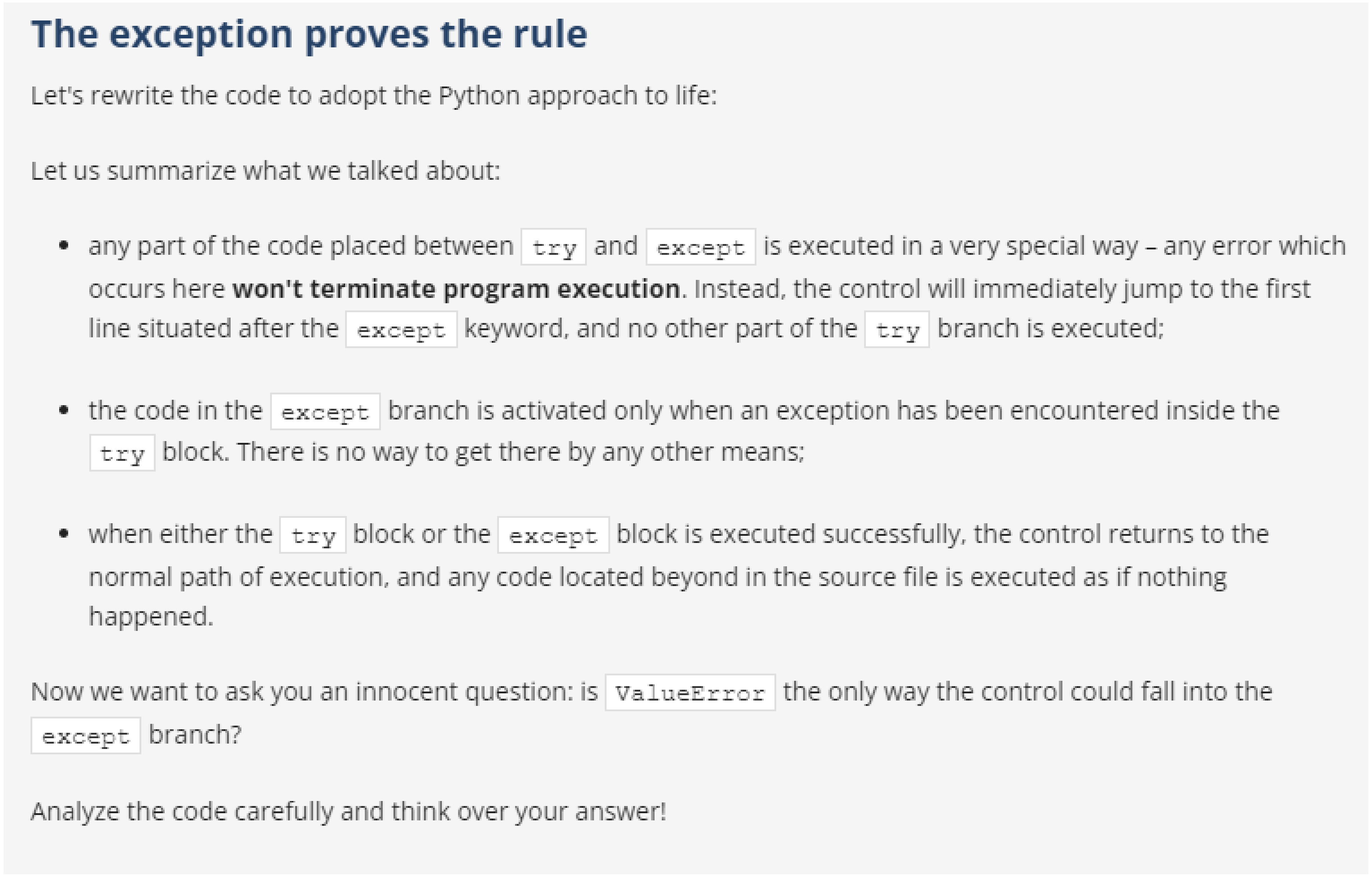
PYTHON PROGRAMMING by SIVA JASTHI



```python
try:
    age = input("Enter your age: ")
    age = int(age)
    print('age: ', age)

except ValueError:
    print("Error: You should give an integer as the input")
```

Catch the exceptions

PYTHON PROGRAMMING by SIVA JASTHI



The exception proves the rule

Let's rewrite the code to adopt the Python approach to life:

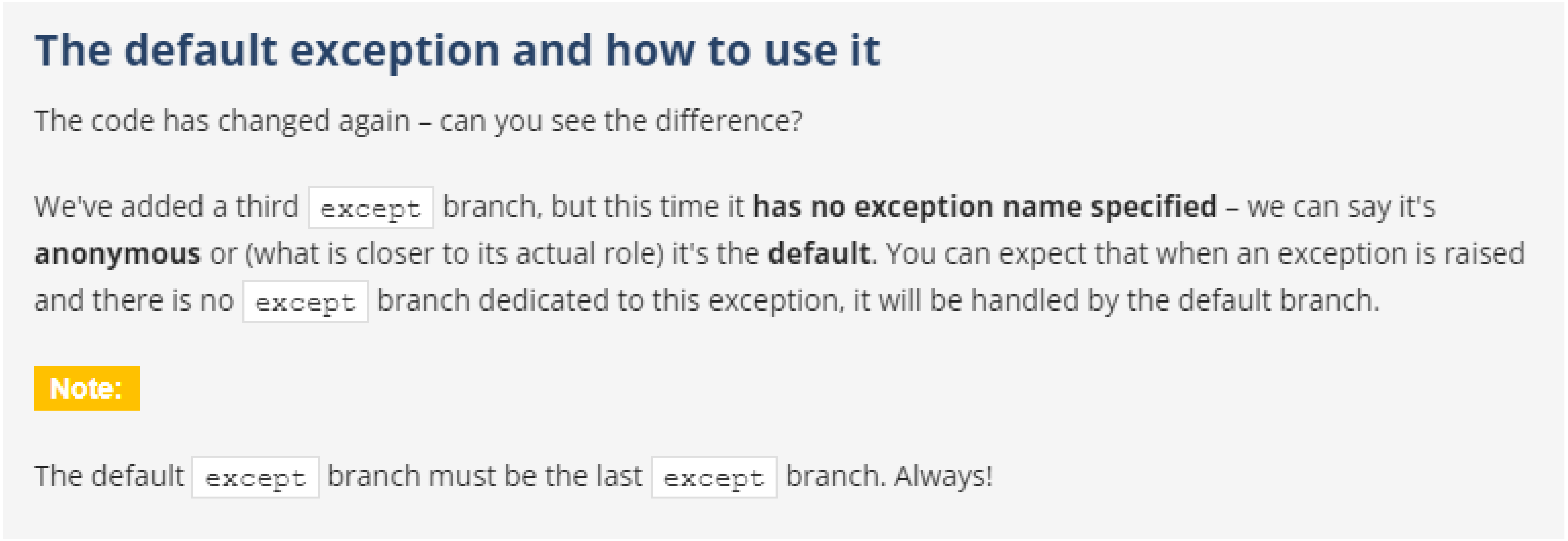
Let us summarize what we talked about:

- any part of the code placed between `try` and `except` is executed in a very special way – any error which occurs here **won't terminate program execution**. Instead, the control will immediately jump to the first line situated after the `except` keyword, and no other part of the `try` branch is executed;
- the code in the `except` branch is activated only when an exception has been encountered inside the `try` block. There is no way to get there by any other means;
- when either the `try` block or the `except` block is executed successfully, the control returns to the normal path of execution, and any code located beyond in the source file is executed as if nothing happened.

Now we want to ask you an innocent question: is `ValueError` the only way the control could fall into the `except` branch?

Analyze the code carefully and think over your answer!

“except” default exception



The default exception and how to use it

The code has changed again – can you see the difference?

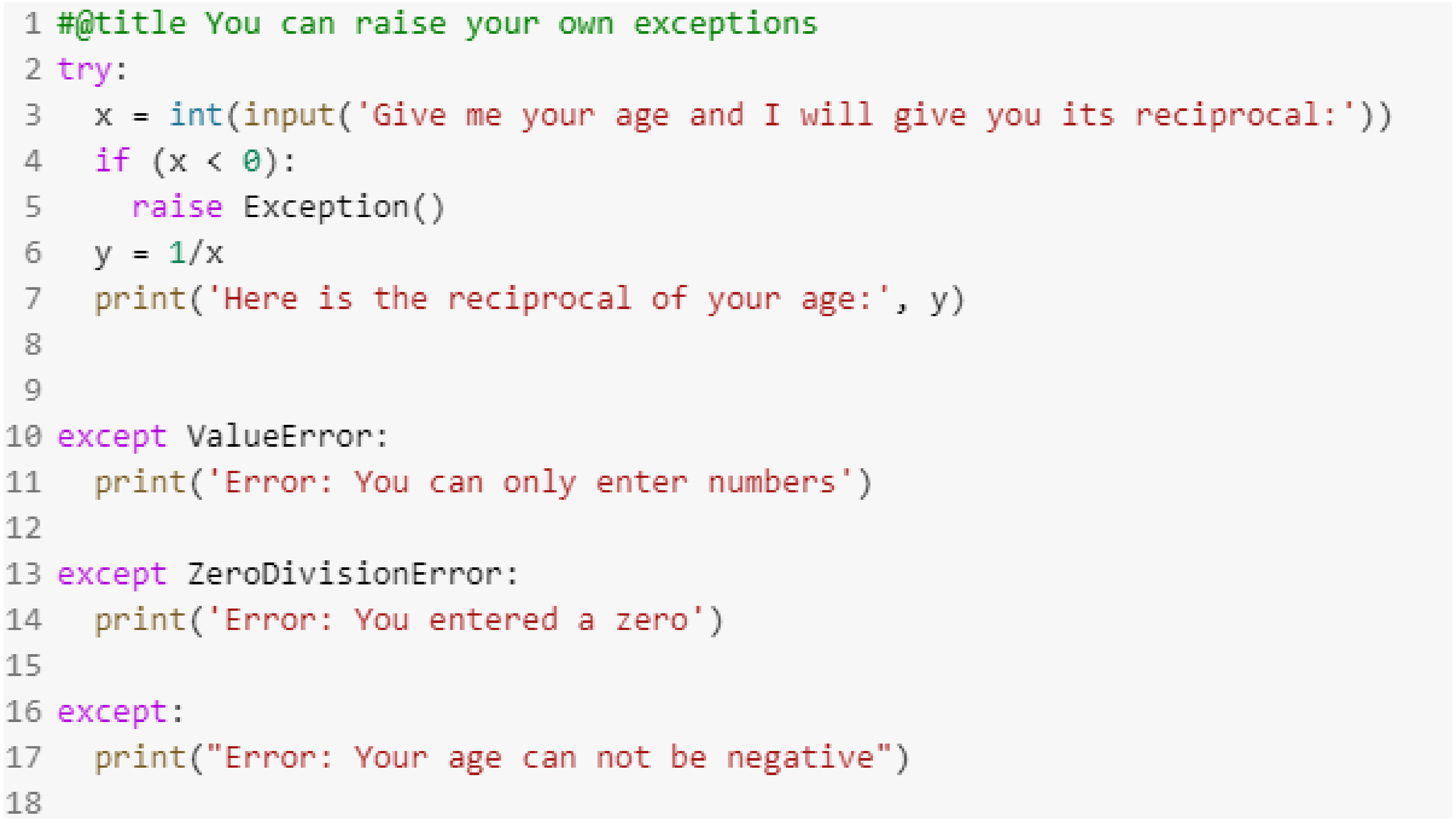
We've added a third `except` branch, but this time it **has no exception name specified** – we can say it's **anonymous** or (what is closer to its actual role) it's the **default**. You can expect that when an exception is raised and there is no `except` branch dedicated to this exception, it will be handled by the default branch.

Note:

The default `except` branch must be the last `except` branch. Always!

PYTHON PROGRAMMING by SIVA JASTHI

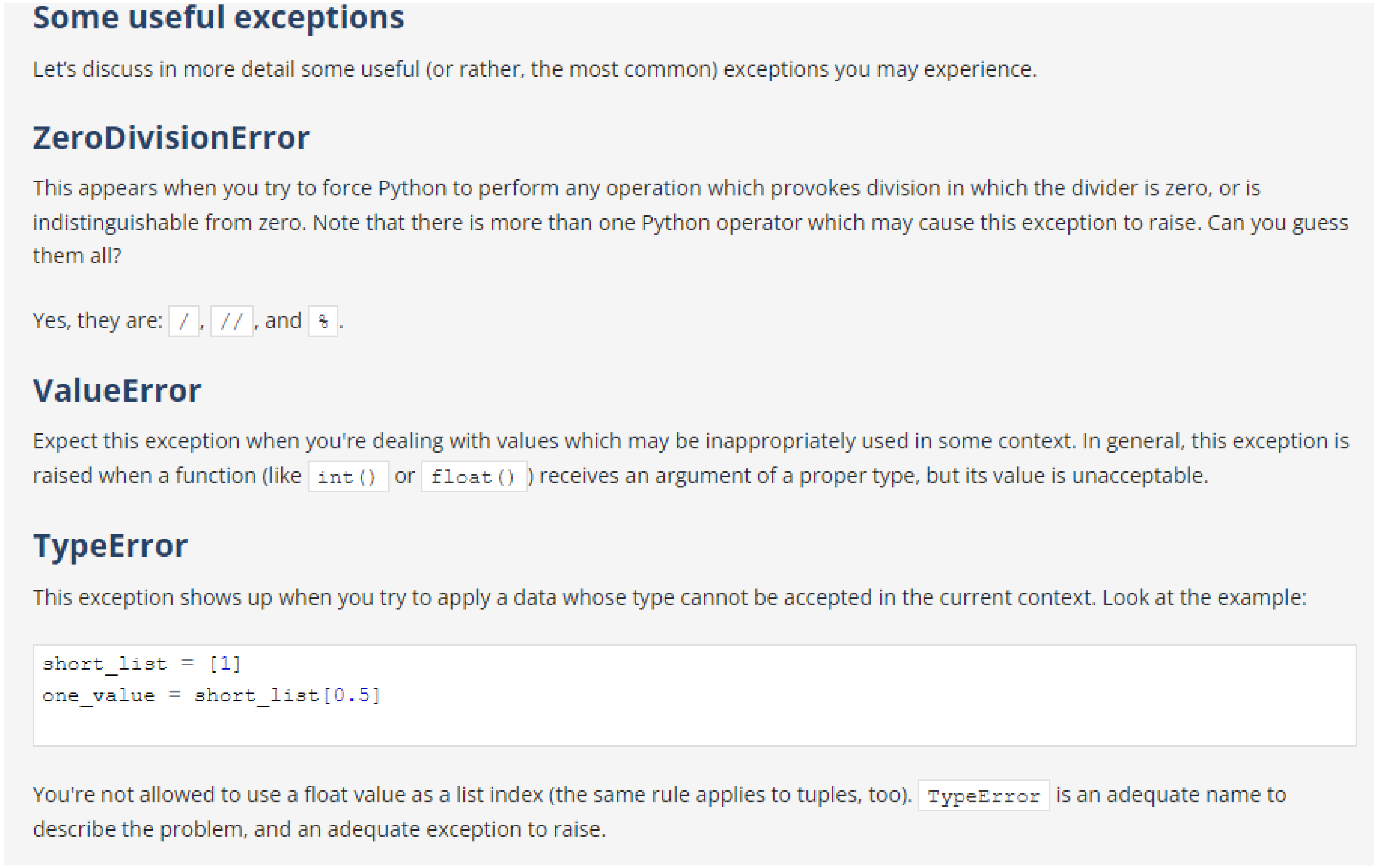
You can catch multiple exceptions



```python
#@title You can raise your own exceptions
try:
  x = int(input('Give me your age and I will give you its reciprocal:'))
  if (x < 0):
    raise Exception()
  y = 1/x
  print('Here is the reciprocal of your age:', y)


except ValueError:
  print('Error: You can only enter numbers')

except ZeroDivisionError:
  print('Error: You entered a zero')

except:
  print("Error: Your age can not be negative")

```

Some useful exceptions

PYTHON PROGRAMMING by SIVA JASTHI



Some useful exceptions

Let's discuss in more detail some useful (or rather, the most common) exceptions you may experience.

ZeroDivisionError

This appears when you try to force Python to perform any operation which provokes division in which the divider is zero, or is indistinguishable from zero. Note that there is more than one Python operator which may cause this exception to raise. Can you guess them all?

Yes, they are: `/`, `//`, and `%`.

ValueError

Expect this exception when you're dealing with values which may be inappropriately used in some context. In general, this exception is raised when a function (like `int()` or `float()`) receives an argument of a proper type, but its value is unacceptable.

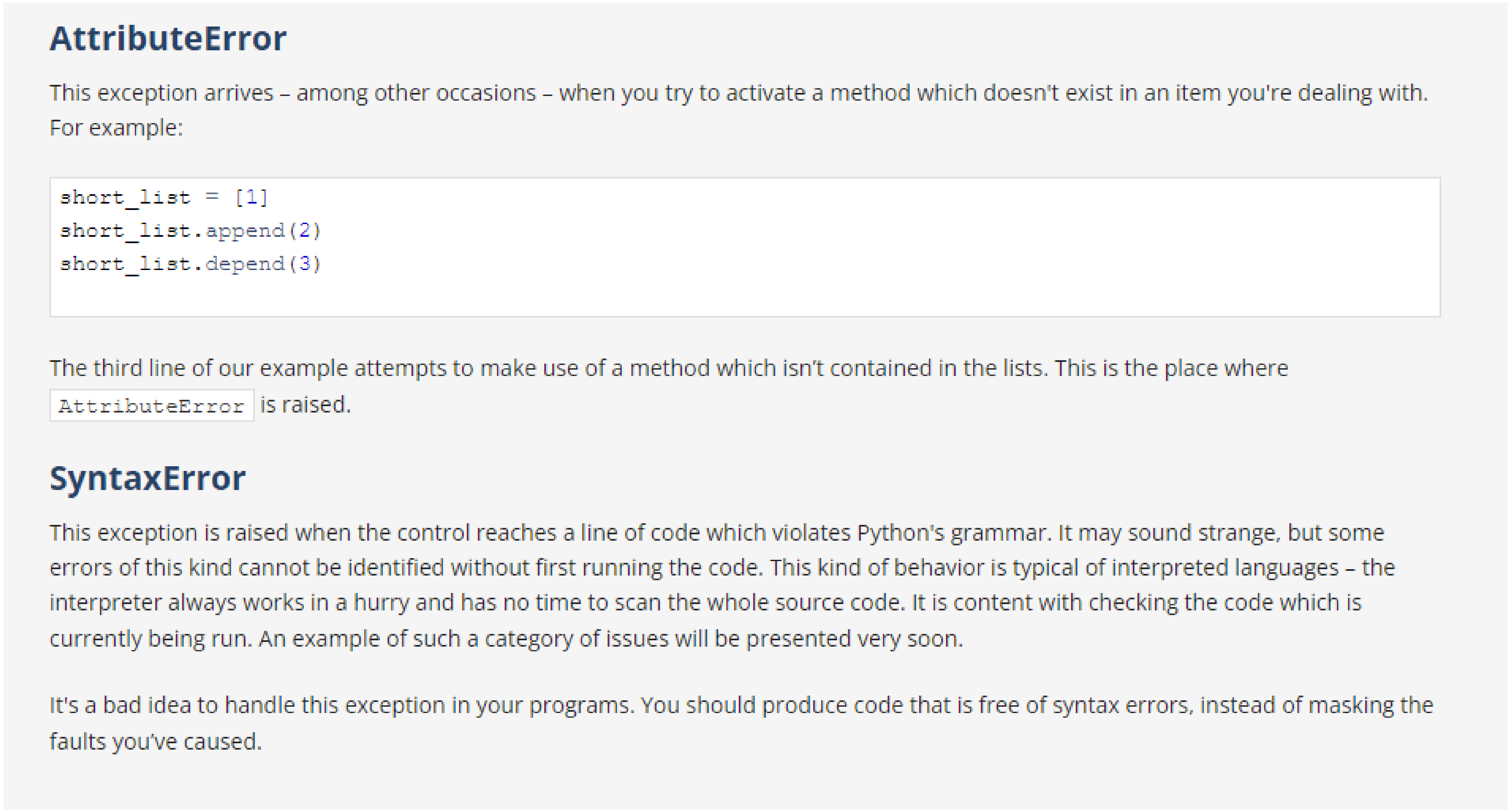
TypeError

This exception shows up when you try to apply a data whose type cannot be accepted in the current context. Look at the example:

```
short_list = [1]
one_value = short_list[0.5]
```

You're not allowed to use a float value as a list index (the same rule applies to tuples, too). `TypeError` is an adequate name to describe the problem, and an adequate exception to raise.

Some useful exceptions (Contd.)



AttributeError

This exception arrives – among other occasions – when you try to activate a method which doesn't exist in an item you're dealing with. For example:

```
short_list = [1]
short_list.append(2)
short_list.depend(3)
```

The third line of our example attempts to make use of a method which isn't contained in the lists. This is the place where `AttributeError` is raised.

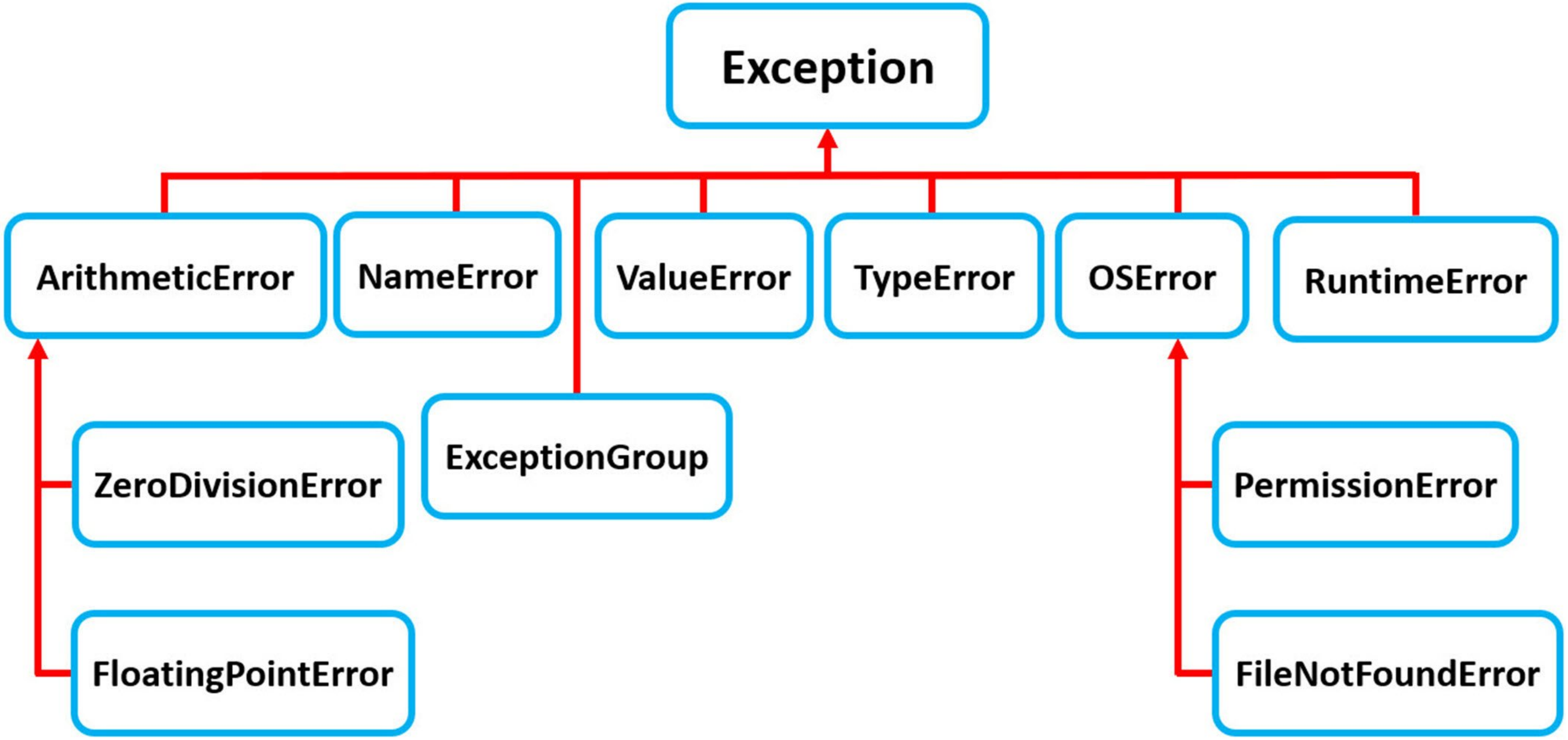
SyntaxError

This exception is raised when the control reaches a line of code which violates Python's grammar. It may sound strange, but some errors of this kind cannot be identified without first running the code. This kind of behavior is typical of interpreted languages – the interpreter always works in a hurry and has no time to scan the whole source code. It is content with checking the code which is currently being run. An example of such a category of issues will be presented very soon.

It's a bad idea to handle this exception in your programs. You should produce code that is free of syntax errors, instead of masking the faults you've caused.

Exception Hierarchy in python

PYTHON PROGRAMMING by SIVA JASTHI



https://realpython.com/python-catch-multiple-exceptions/

Exception Hierarchy in python

PYTHON PROGRAMMING by SIVA JASTHI

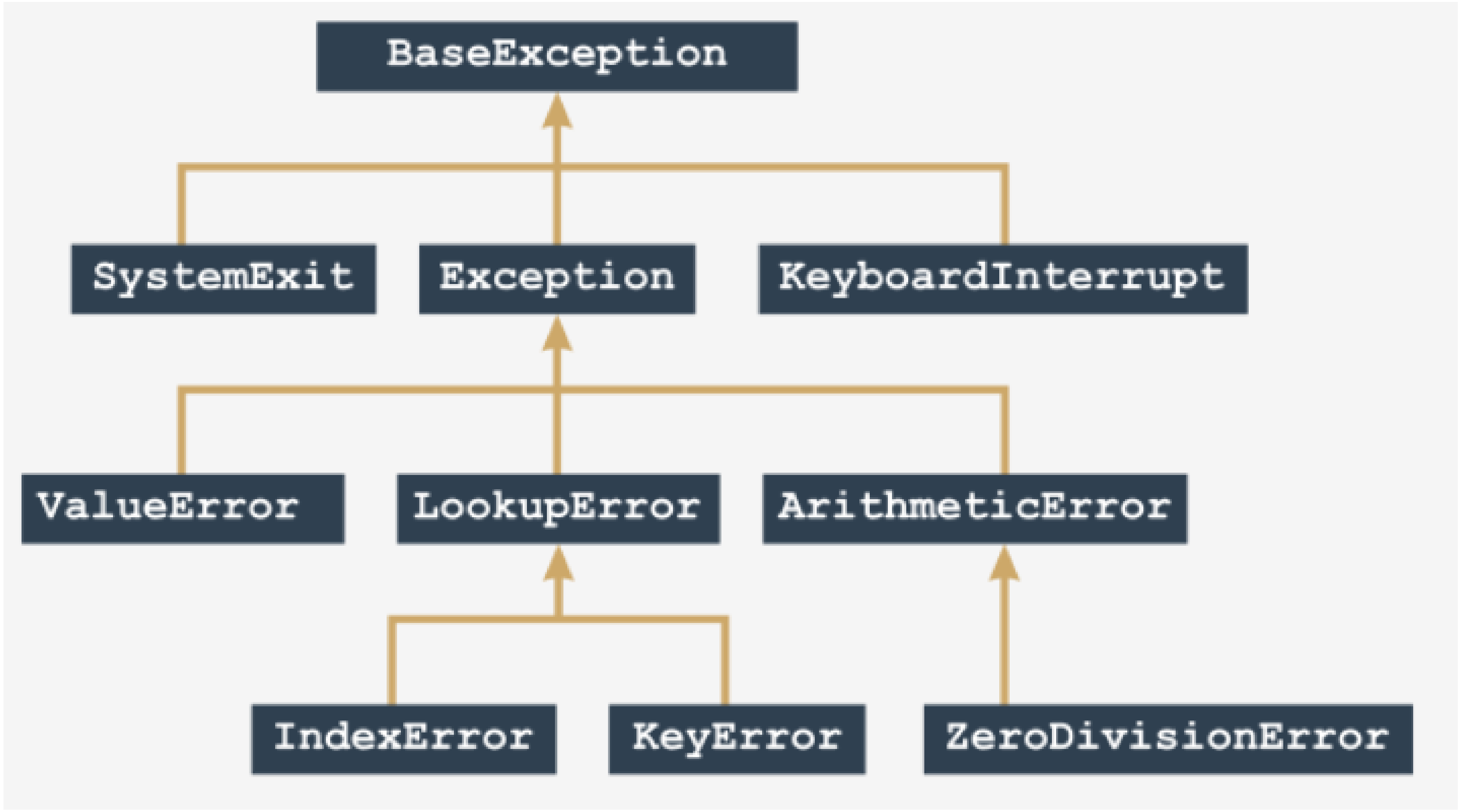
```
BaseException
 ├── BaseExceptionGroup
 ├── GeneratorExit
 ├── KeyboardInterrupt
 ├── SystemExit
 └── Exception
      ├── ArithmeticError
      │    ├── FloatingPointError
      │    ├── OverflowError
      │    └── ZeroDivisionError
      ├── AssertionError
      ├── AttributeError
      ├── BufferError
      ├── EOFError
      ├── ExceptionGroup [BaseExceptionGroup]
      ├── ImportError
      │    └── ModuleNotFoundError
      ├── LookupError
      │    ├── IndexError
      │    └── KeyError
      ├── MemoryError
      ├── NameError
      │    └── UnboundLocalError
      │
```

```
├── OSError
│    ├── BlockingIOError
│    ├── ChildProcessError
│    ├── ConnectionError
│    │    ├── BrokenPipeError
│    │    ├── ConnectionAbortedError
│    │    ├── ConnectionRefusedError
│    │    └── ConnectionResetError
│    ├── FileExistsError
│    ├── FileNotFoundError
│    ├── InterruptedError
│    ├── IsADirectoryError
│    ├── NotADirectoryError
│    ├── PermissionError
│    ├── ProcessLookupError
│    └── TimeoutError
├── ReferenceError
├── RuntimeError
│    ├── NotImplementedError
│    └── RecursionError
├── StopAsyncIteration
├── StopIteration
```

```
├── SyntaxError
│    └── IndentationError
│         └── TabError
├── SystemError
├── TypeError
├── ValueError
│    └── UnicodeError
│         ├── UnicodeDecodeError
│         ├── UnicodeEncodeError
│         └── UnicodeTranslateError
└── Warning
     ├── BytesWarning
     ├── DeprecationWarning
     ├── EncodingWarning
     ├── FutureWarning
     ├── ImportWarning
     ├── PendingDeprecationWarning
     ├── ResourceWarning
     ├── RuntimeWarning
     ├── SyntaxWarning
     ├── UnicodeWarning
     └── UserWarning
```

https://docs.python.org/3/library/exceptions.html

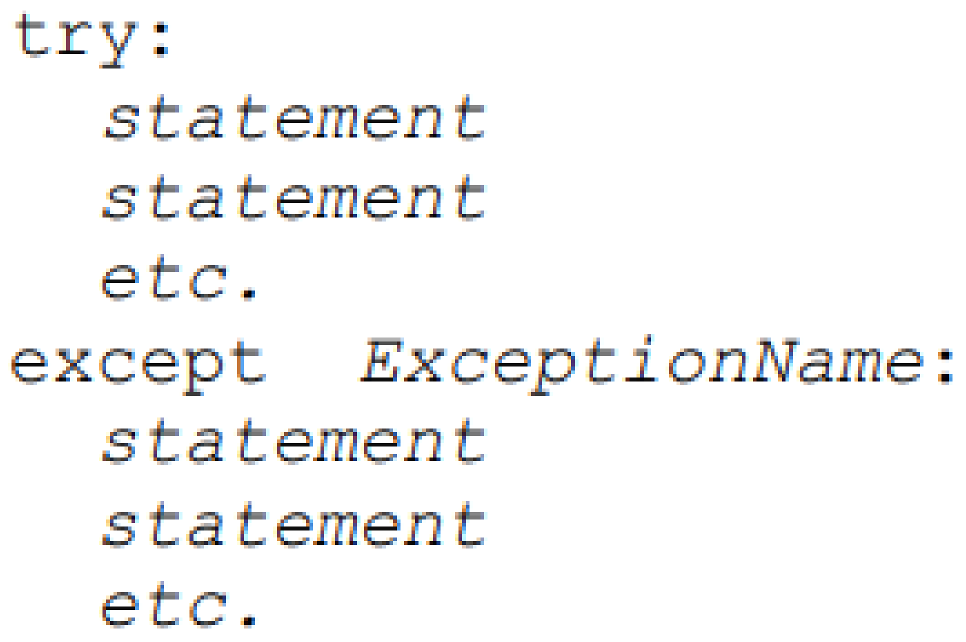
Exception Hierarchy in python



https://docs.python.org/3/library/exceptions.html

Catching multiple exceptions in one go

```
try:
      statement 1
      statement 2

except Exception as err:
      do something to catch the errors
```



```
try:
   statement
   statement
   etc.
except  ExceptionName:
   statement
   statement
   etc.
```

PYTHON PROGRAMMING by SIVA JASTHI

Thank You.

PYTHON PROGRAMMING

by SIVA JASTHI