Object Oriented Programming using Python (I)

Lecture(9)

Error handling

Prepared by:Ahmed Eskander Mezher

University of Information Technology and Communications

College of Business Informatics

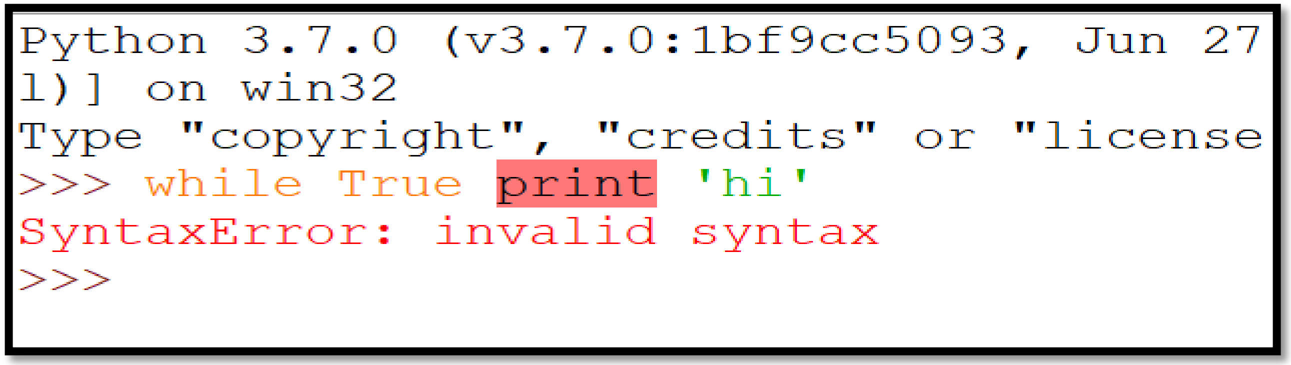
Handling Exceptions

What is an exception?

- Exceptions are run-time errors
- Whenever the interpreter has a problem it notifies the user/programmer by raising an *exception*
- The parser repeats the line and displays a little ‘arrow’ pointing at the earliest point in the line where the error was detected.

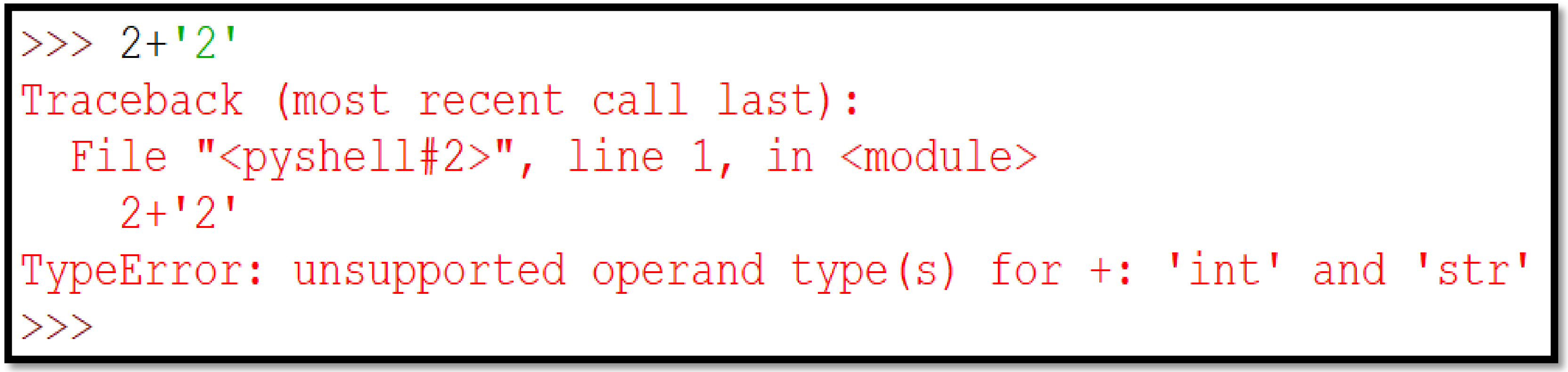
Some Error types

➢ <u>SyntaxError:</u> Raised when there is an error in Python syntax.



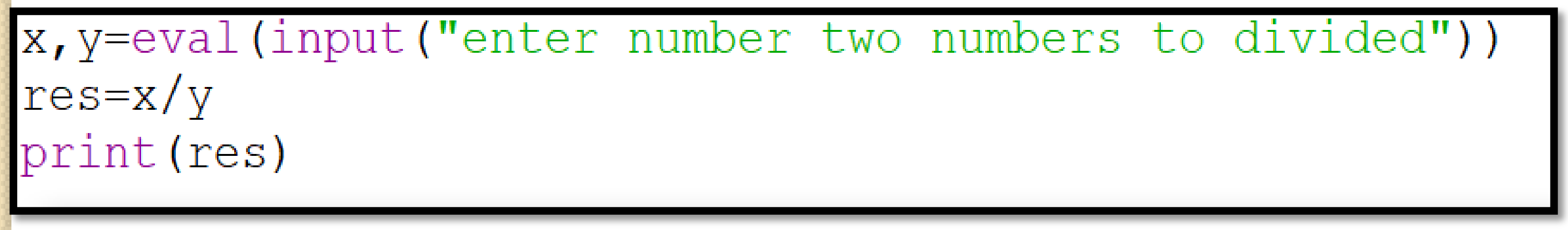
```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27
1)] on win32
Type "copyright", "credits" or "license
>>> while True print 'hi'
SyntaxError: invalid syntax
>>>
```

➢ <u>TypeError</u> Raised when an operation or function is attempted that is invalid for the specified data type

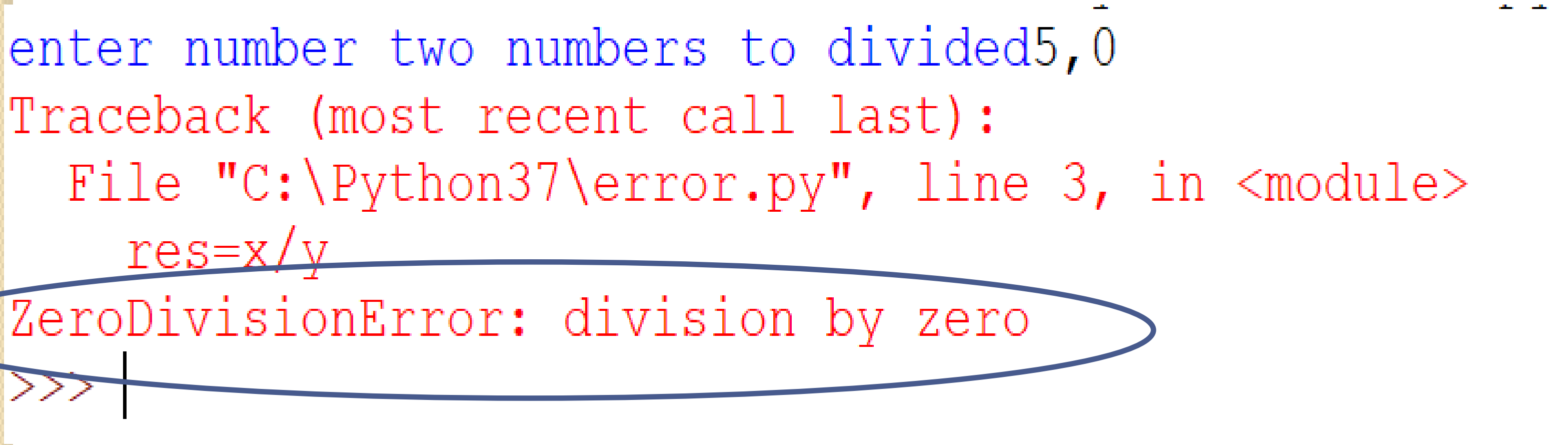


```
>>> 2+'2'
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    2+'2'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

➢**ZeroDivisonError**: Raised when division or modulo by zero takes place for all numeric types.

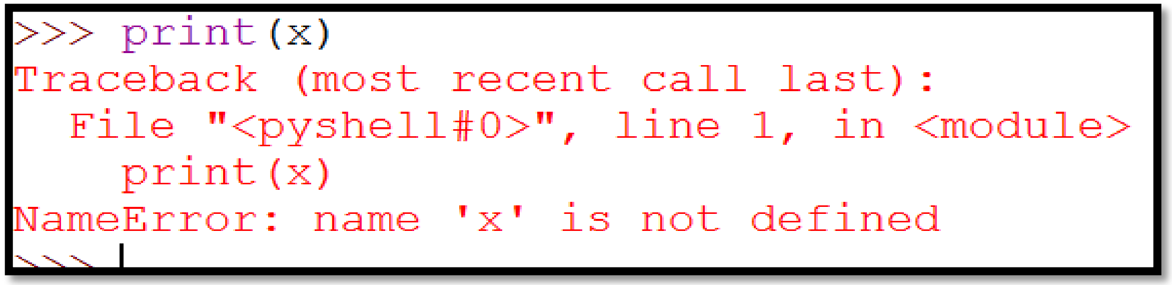


```
x,y=eval(input("enter number two numbers to divided"))
res=x/y
print(res)
```



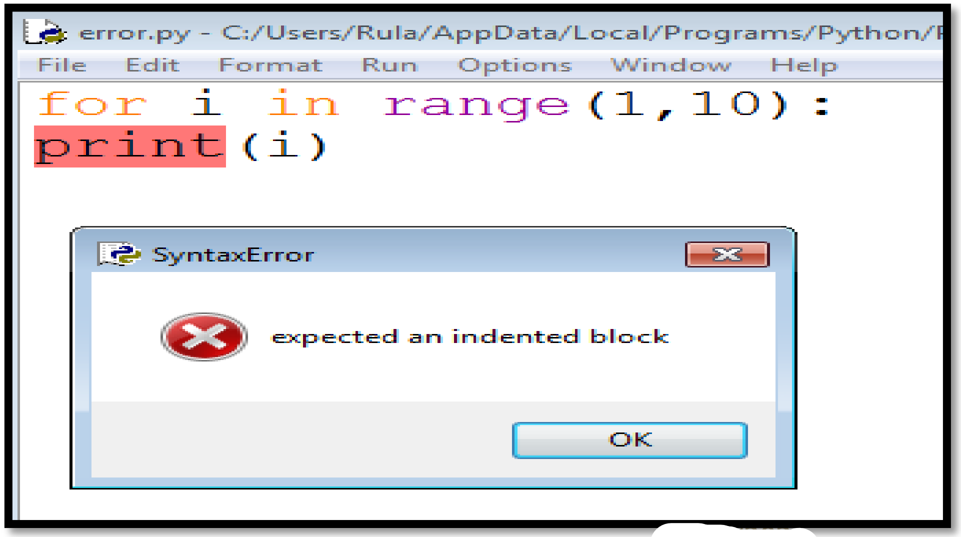
```
enter number two numbers to divided5,0
Traceback (most recent call last):
  File "C:\Python37\error.py", line 3, in <module>
    res=x/y
ZeroDivisionError: division by zero
>>>
```

➢**NameError**:Raised when an identifier is not found in the local or global namespace.



```
>>> print(x)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
>>>
```

➢**IndentationError** Raised when indentation is not specified properly.



error.py - C:/Users/Rula/AppData/Local/Programs/Python/
File Edit Format Run Options Window Help

```
for i in range(1,10):
print(i)
```

SyntaxError

expected an indented block

Error handling

- Once you catch the error, you need to handle it
  1. Perform an action to correct the program
  2. Generate a customized error message and gracefully end the program

Handling exceptions

- By default, the interpreter handles exceptions by stopping the program and printing an error message
- we can override this behavior by *catching* the exception

Handling exceptions

- The try block lets you test a block of code for errors.
- The except block lets you handle the error.
- The finally block lets you execute code, regardless of the result of the try- and except blocks.

```python
#The try block will generate an error, because x is not defined:

try:
  print(x)
except:
  print("An exception occurred")
```

```
An exception occurred
```

```python
#The finally block gets executed no matter if the try block raises any errors
or not:

try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

```
Something went wrong
The 'try except' is finished
```

```python
while True:
    try:
        n=int(input('enter integer value'))
        break
    except:
        print('invalid the number must be integer')
print('done')
```

```
enter integer value5.3
invalid the number must be integer
enter integer value3.4
invalid the number must be integer
enter integer value2
done
>>> 
```