Inspiring Excellence

Course Title: Programming Language II

Course Code: CSE 111

Semester: Summer 2020

Topic: Exception

Table of Contents

1. Types of error

- **Syntax Error/Compile Time Error**

  - ✓ Error caused by not following the proper structure (syntax) of the language is called syntax error or parsing error.
  - ✓ Common Python syntax errors include:

    - leaving out a keyword
    - putting a keyword in the wrong place
    - leaving out a symbol, such as a colon, comma or brackets
    - misspelling a keyword
    - incorrect indentation
    - empty block

```
# test.py
a = 5
if a < 8
   print(a)
```

```
# output
File "<interactive input>", line 2
    if a < 8
           ^
SyntaxError: invalid syntax
```

- **Runtime Error**

  - ✓ If a program is syntactically correct – that is, free of syntax errors – it will be run by the Python interpreter. However, the program may exit unexpectedly during execution if it encounters a runtime error – a problem which was not detected when the program was parsed, but is only revealed when a particular line is executed.
  - ✓ When a program comes to a halt because of a runtime error, we say that it has crashed.
  - ✓ Some examples of Python runtime errors:

    - division by zero
    - performing an operation on incompatible types
    - using an identifier which has not been defined
    - accessing a list element, dictionary value or object attribute which doesn't exist
    - trying to access a file which doesn't exist

```
# test.py
1/0
```

```
# output
Traceback (most recent call last):
"<interactive input>", line 1, in
<module>
ZeroDivisionError: division by zero
```

```
# test.py
Open("test.txt")
```

```
# output
Traceback (most recent call last):
"<interactive input>", line 1, in
<module>
FileNotFoundError: [Errno 2] No such
file or directory: 'test.txt'
```

- **Logical Error**

  - ✓ Logical errors are the most difficult to fix. They occur when the program runs without crashing, but produces an incorrect result.
  - ✓ The error is caused by a mistake in the program's logic.
  - ✓ You won't get an error message, because no syntax or runtime error has occurred.

```
# test.py
x = 3
y = 4

z = x + y / 2
print ('The average of the two numbers is: ', z)
```

```
# output
The average of the two numbers is: 5.0
```

The example above should calculate the average of the two numbers the user enters. But, because of the order of operations in arithmetic (the division is evaluated before addition) the program will not give the right answer.

2. Exception

An exception is an error that happens during execution of a program. When that error occurs, Python generates an exception that can be handled, which avoids your program from being crashed.

Why we use Exceptions?

Exceptions are convenient in many ways for handling errors and special conditions in a program. When you think that you have a code which can produce an error then you can use exception handling.

3. Working procedure of exception

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

```
# test.py
try:
  print(x)
except:
  print("An exception occurred")
```

```
# output
An exception occurred
```

Since the try block raises an error, the except block will be executed. Without the try block, the program will crash and raise an error.

Examples:

```
# test1.py
a = 5
b = 0
try:
  x = a / b
  print(x)
except:
  print('can't divide by zero')
```

```
# output
can't divide by zero
```

```
# test2.py
a=6
b=0
try:
    print("start procedure")
    x=a/b
    print("end procedure")
    print(x)
except:
    print("can't divide by zero")
```

```
# output
start procedure
can't divide by zero
```

4. Different types of exception

Exception	Cause of Error
IndexError	Raised when the index of a sequence is out of range.
NameError	Raised when a variable is not found in local or global scope.
IndentationError	Raised when there is incorrect indentation.
ZeroDivisionError	Raised when the second operand of division or modulo operation is zero.
TypeError	Raised when a function or operation is applied to an object of incorrect type.
RuntimeError	Raised when an error does not fall under any other category.

5. Multiple exception

- See the following code

```
# test.py
try:
    a=input()
    a=int(a)
    b=input()
    b=int(b)
    x=a/b
    print(x)
except ZeroDivisionError:
    print("can't divide by zero")
except ValueError:
        print("please input an integer")
finally:
   print("program terminates")
```

```
# Output
# If inputs are 6 and 2
3.0
program terminates
```

```
# Output
# If inputs are 6 and 0
can't divide by zero
program terminates
```

```
# Output
# If inputs are not integers. Say input is 'x'
please input an integer
program terminates
```

Practice

- Take 4 integer in a list from input and find whether the list is palindrome or not. Your program should handle indexError, valueError and typeError.
- Take 4 integer in a list from input and find whether the list is prime or not. Your program should handle indexError, valueError and typeError.