ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
Faculty of English

Robert Dyzman, M.Sc.Eng.

PYTHON PROGRAMMING
CLASS 07

Run „Teams”

Start your IDE

AGENDA:

- Create a file *„class_pp_07.py”*
- Exceptions
- Handling exceptions
- Raising exceptions
- Pt-02
- Exercises
- Push to GitHub, alternatively Copy/Paste to Teams

TERNARY OPERATOR

- Ternary operators are more commonly known as conditional expressions in Python
- It allows testing a condition in a single line
- Syntax:

var_1 = stat_if_true if condition else statement_if_false

egg = 'boiled_egg' if temperature == 100 and c_time == 5 else 'raw_egg'

```
#statement => value or expression returning a #value
a, b = 11, 15
min = a if a < b else b
print(min)
```

```
a, b = 11, 15
min = a+b if a < b else a-b
print(min)
```

EXCEPTIONS

- An exception is a kind of error which terminates the execution of the program;
- It usually happens because of programmer's mistakes, bad data the program get from the user or resources not being available.
- It is our job as a programmers to handle these exceptions and prevent the application from crashing

Exceptions versus Syntax Errors

- Syntax Error, when interpreter points the error

```
print( 0 / 0 ))
```

- Exception, this type of error occurs whenever syntactically correct Python code results in an error.

```
print( 0 / 0 )
```

TYPES OF EXCEPTIONS (COMMON RUNTIME ERRORS)

- NameError: local or global name not found,
- TypeError: operand doesn’t have correct type,
- ValueError: value is illegal,
- IndexError: if index out of range,
- IOError: IO system error, e.g., file not found
- ZeroDivisionError: if you try to divide a numer by zero
- Others

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

EXAMPLES

```
# 01, IndexError
numbers = [1, 2, 3]
print(numbers[3])
print('Here I am’)
# 02, TypeError
L = [1, 7, 4]
print(int(L))
# 03, TypeError
print('a'/4)
# 04, NameError
print(a)
num = int(input('Enter the integer number: … ,))
```

EXAMPLES

```
# 05, ValueError
num = int(input('Enter the integer numer: …')) # input a
print(f'Our number is {num}’)
# 06, IOError-> FileNotFoundError
file = open('ztest.py')
print('File is opened')
file.close()
# 07, FileNotFoundError
file = open('test.py')
numbers = [1, 2, 3]
print(numbers[3])
print('File is opened')
file.close()
```

HANDLING EXCEPTIONS

```
# 01, handle IndexError
try:
    numbers = [1, 2, 3]
    print(numbers[3])
except IndexError:
    print('Wrong Index!!! ')
print('Here I am')
```

HANDLING EXCEPTIONS

```
# 05 how to handle it ?
try:
    num = int(input('Enter the integer numer: … '))  # input a
    print(f'Our number is: {num}')
except ValueError:
    print('You didn\'t enter the valid number!’)
```

HANDLING EXCEPTIONS

```
# 05 how to handle it ?
try:
    a = int(input('Enter the integer numer \'a\': … '))
    b = int(input('Enter the integer numer \'b\': … '))
    num = a / b
    print(f'The result of division {a} / {b} = {num}')
except ValueError:
    print('You didn\'t enter the valid number!')
```

HANDLING EXCEPTIONS

```
# 05 how to handle it ?
try:
    a = int(input('Enter the integer numer \'a\': … '))
    b = int(input('Enter the integer numer \'b\': … '))
    num = a / b
    print(f'The result of division {a} / {b} = {num}')
except ValueError:
    print('You didn\'t enter the valid number!')
except ZeroDivisionError:
    print('You can\'t  divide by 0!')
```

RAISING EXCEPTIONS

```
def divide(a,b):
    '''
    input: a,b : positive int
    output : float
    '''
    if a<0 or b < 0:
        raise ValueError('You didn\'t enter the valid numbers!')
    return a/b

try:
    a = int(input('Enter the positive integer numer \'a\': … '))
    b = int(input('Enter the positive integer numer \'b\': … '))
    num = divide(a,b)
    print(f'The result of division {a} / {b} = {num}')
except ValueError as error:
    print(error)
except ZeroDivisionError:
    print('You can\'t  divide by 0!')
```

EXERCISE 140

```
'''
Exercise 140, file pp_140.py

Variables are defined below:

sum = 3000
counter = 0

We want to divide sum by counter.

Use the try... except... clause to handle a division by zero exception.
If the division is done correctly, print the result to the console. At the time
of error, let the following text be printed to the console: "You can't  divide
by 0!"
'''
```

EXERCISE 150, based on exercise 100

```
'''
Exercise 150, file pp_150.py (on the basis of exercie 100)
Write a program that computes the value of n factorial - n!
Use iterative implementation
Expected results
0! = 1
1! = 1
2! = 1 * 2 = 2
3! = 1 * 2 * 3 = 6
4! = 1 * 2 * 3 * 4
10! = 3628800
32! = 263130836933693530167218012160000000
n! = 1 * 2 * 3 * .... * n

In order to do it implement the function
def factorial(n: int) -> int:

Improve error handling using exceptions
'''
```