
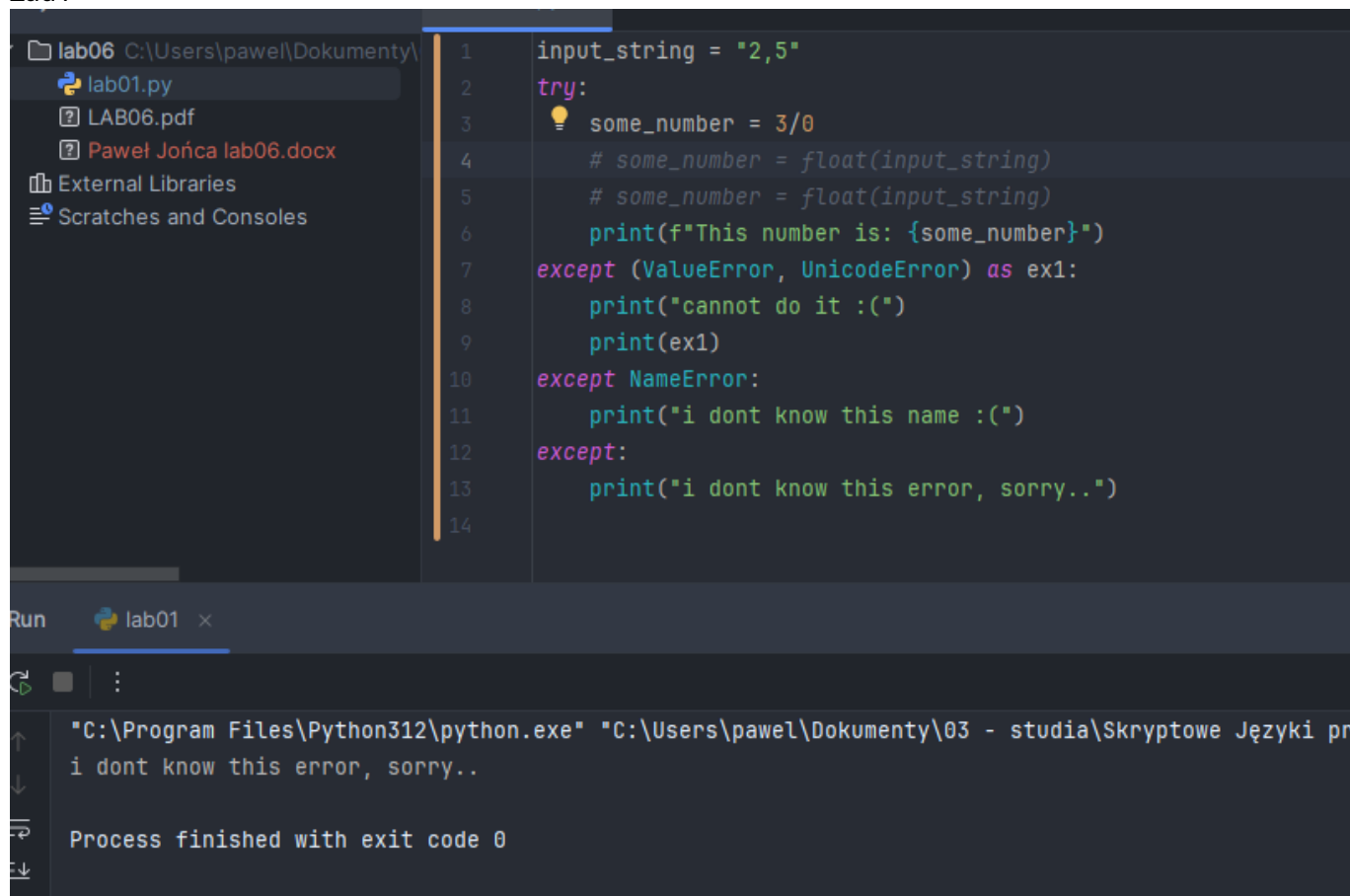
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki <b>Zakład Systemów Teleinformatycznych</b>		
<b>Przedmiot</b>	Skryptowe języki programowania		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	Wyjątki i pliki		
<b>Student</b>	Paweł Jońca		
<b>Nr lab.</b>	6	<b>Data wykonania</b>	25.11.2024r
<b>Ocena</b>		<b>Data oddania spr.</b>	25.11.2024r

Poprawka z kodem w formie tekstu

Tekst można kopiować mimo że wygląda jakby to były ss, każdy kod jest na końcu danego zadania

Zad1



```

1  input_string = "2,5"
2  try:
3      some_number = 3/0
4      # some_number = float(input_string)
5      # some_number = float(input_string)
6      print(f"This number is: {some_number}")
7  except (ValueError, UnicodeError) as ex1:
8      print("cannot do it :(")
9      print(ex1)
10 except NameError:
11     print("i dont know this name :(")
12 except:
13     print("i dont know this error, sorry..")
14
Run lab01 x
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty\03 - studia\Skryptowe Języki pr
i dont know this error, sorry..
Process finished with exit code 0

```

Tutaj mamy błąd dzielenia przez 0 i wypływa to ostatni blok except

```
Project ▾ lab01.py ×
lab06 C:\Users\pawel\Dokumenty\
  lab01.py
  LAB06.pdf
  Paweł Jońca lab06.docx
  External Libraries
  Scratches and Consoles

1 input_string = "2,5"
2 try:
3     # some_number = 3/0
4     some_number = float(input_string)
5     # some_number = float(input_string)
6     print(f"This number is: {some_number}")
7 except (ValueError, UnicodeError) as ex1:
8     print("cannot do it :(")
9     print(ex1)
10 except NameError:
11     print("i dont know this name :(")
12 except:
13     print("i dont know this error, sorry..")
14

Run lab01 ×

"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty\03 - studia\Skryptowe Języki programowania\lab01.py"
cannot do it :(
could not convert string to float: '2,5'

Process finished with exit code 0
```

Linia 4 i 5 są takie same więc mamy ten sam efekt. Tam można było się odwołać do niezdefiniowanej zmiennej, żeby działał blok except NameError

```
Project ▾ lab01.py ×
lab06 C:\Users\pawel\Dokumenty\
  lab01.py
  LAB06.pdf
  Paweł Jońca lab06.docx
  External Libraries
  Scratches and Consoles

1 input_string = "2,5"
2 try:
3     # some_number = 3/0
4     # some_number = float(input_string)
5     some_number = float(input_string)
6     print(f"This number is: {some_number}")
7 except (ValueError, UnicodeError) as ex1:
8     print("cannot do it :(")
9     print(ex1)
10 except NameError:
11     print("i dont know this name :(")
12 except:
13     print("i dont know this error, sorry..")
14

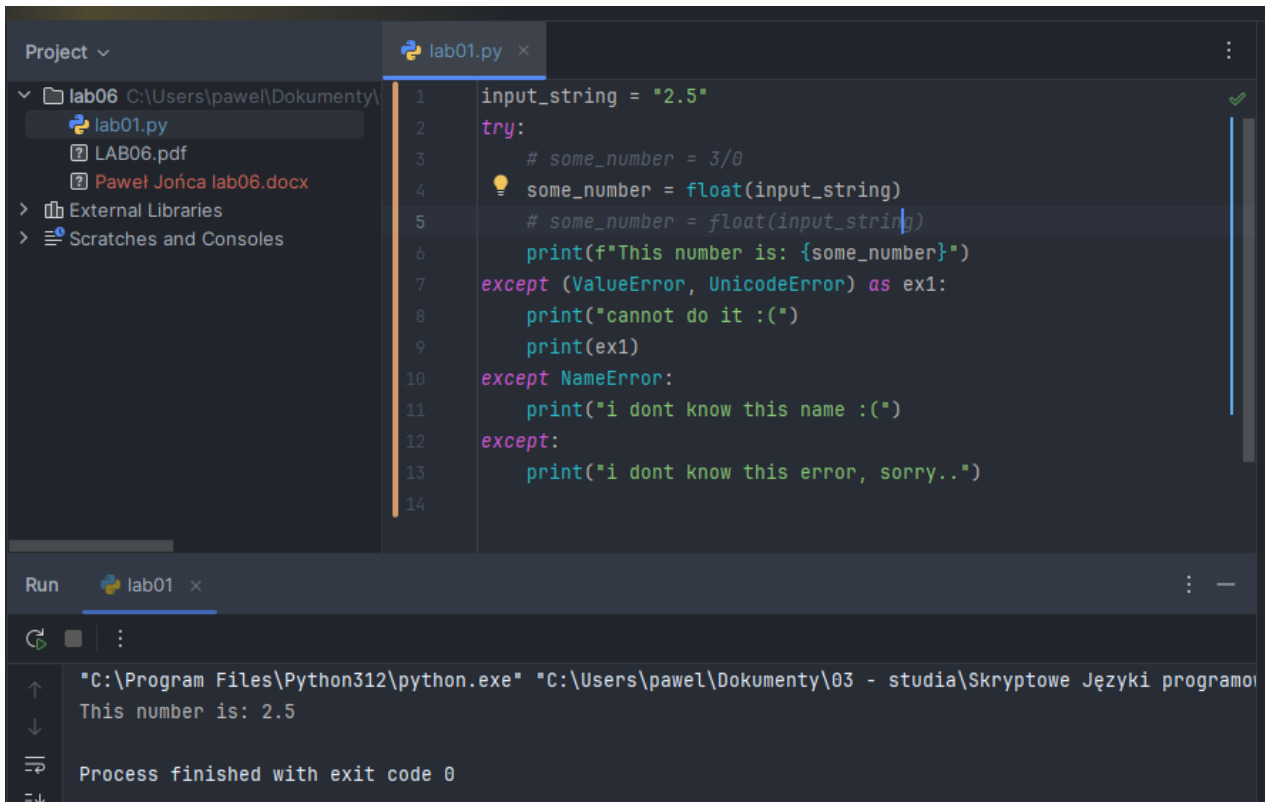
Run lab01 ×

"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty\03 - studia\Skryptowe Języki programowania\lab01.py"
cannot do it :(
could not convert string to float: '2,5'

Process finished with exit code 0
```

Input\_string = "2,5" spowoduje ValueError, ponieważ przecinek jest nieprawidłowym separatorem dziesiętnym. Zostanie obsłużony przez pierwszy except ValueError. Po zmianie

Przecinka na kropkę kod działa. :



The screenshot shows an IDE with a project named 'lab06' and a file 'lab01.py'. The code in 'lab01.py' is as follows:

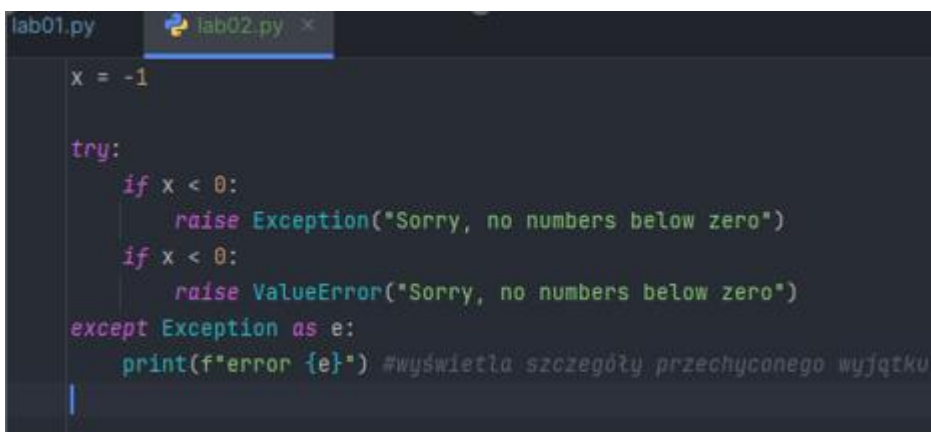
```
1 input_string = "2.5"
2 try:
3     # some_number = 3/0
4     some_number = float(input_string)
5     # some_number = float(input_string)
6     print(f"This number is: {some_number}")
7 except (ValueError, UnicodeError) as ex1:
8     print("cannot do it :(")
9     print(ex1)
10 except NameError:
11     print("i dont know this name :(")
12 except:
13     print("i dont know this error, sorry..")
14
```

The 'Run' tab shows the execution output:

```
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty\03 - studia\Skryptowe Języki programowania\lab01.py"
This number is: 2.5
Process finished with exit code 0
```

```
input_string = "2,5"
try:
    # some_number = 3/0
    # some_number = float(input_string)
    some_number = float(input_string)
    print(f"This number is: {some_number}")
except (ValueError, UnicodeError) as ex1:
    print("cannot do it :(")
    print(ex1)
except NameError:
    print("i dont know this name :(")
except:
    print("i dont know this error, sorry..")
```

Zad 2



The screenshot shows an IDE with a file 'lab02.py'. The code in 'lab02.py' is as follows:

```
x = -1

try:
    if x < 0:
        raise Exception("Sorry, no numbers below zero")
    if x < 0:
        raise ValueError("Sorry, no numbers below zero")
except Exception as e:
    print(f"error {e}") #wyświetla szczegóły przechyczonego wyjątku
```

```
lab02 x
"C:\Program Files\Python312\python.exe" "C:\
error Sorry, no numbers below zero
```

```
x = -1

try:
    if x < 0:
        raise Exception("Sorry, no numbers below zero")
    if x < 0:
        raise ValueError("Sorry, no numbers below zero")
except Exception as e:
    print(f"error {e}") #wyświetla szczegóły przechyconego wyjątku
```

### Zad 3

```
lab01.py lab02.py lab03.py x
1 try:
2     x = int(input("Enter a number:"))
3     result = 10 /x
4 except ValueError:#obsługa błędu konwersji na liczbę ca
5     print("Something went wrong")
6 except ZeroDivisionError:#obsługa dzielenia przez zero
7     print("You can't divide by zero")
8 else: #ten blok wykona się jeśli nie wystąpi błąd
9     print(f"Everything went fine {result}")
10 finally:#wykona się zawsze
11     print("I dont't care whether something went wrong")
12

un lab03 x
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\D
Enter a number:5
Everything went fine 2.0
I dont't care whether something went wrong
Process finished with exit code 0

"C:\Program Files\Python312\python.exe" "C:\Users\p
Enter a number:s
Something went wrong
I dont't care whether something went wrong
```

```
"C:\Program Files\Python312\python.exe" "C:\Use
Enter a number:0
You can't divide by zero
I dont't care whether something went wrong

Process finished with exit code 0
```

```
try:
    x = int(input("Enter a number:"))
    result = 10 /x
except ValueError:#obsługa błędu konwersji na liczbę całkowita
    print("Something went wrong")
except ZeroDivisionError:#obsługa dzielenia przez zero
    print("You can't divide by zero")
else: #ten blok wykona się jeśli nie wystąpi błąd
    print(f"Everything went fine {result}")
finally:#wykona się zawsze
    print("I dont't care whether something went wrong")
```

#### Zad 4

Jaki wyjątek zostanie rzucony w przypadku, gdy plik nie istnieje? – Jeżeli chcemy otworzyć plik w trybie „r” odczytu, a plik nie istnieje, zostanie rzucony komunikat. FileNotFoundError

Jaki wyjątek zostanie rzucony, jeśli będziemy chcieli zapisać do pliku? - Jeśli otworzysz plik w trybie "r" (odczytu) i spróbujesz do niego zapisać, zostanie rzucony wyjątek io.UnsupportedOperation

Jest to wyjątek wskazujący, że operacja zapisu nie jest wspierana dla pliku otwartego w trybie tylko do odczytu.

```
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\
The file does not exist! Please check the file path.
some text

some text

The file does not exist! Please check the file path.
['some text\n', 'hello world']
Cannot write to a file opened in read mode!

Process finished with exit code 0
```

```

1  from io import UnsupportedOperation
2
3  try:
4      # Próba odczytu całego pliku
5      with open(r'C:\pawel\Dokumenty\03 - studia\Skryptowe Języki programowania\lab06\plik.txt', "r") as f:
6          print(f.read())
7  except FileNotFoundError:
8      print("The file does not exist! Please check the file path.")
9  except Exception as e:
10     print(f"An unexpected error occurred: {e}")
11
12
13  try:
14      # Próba odczytu jednej linii
15      with open("plik.txt", "r") as f:
16          print(f.readline())
17  except FileNotFoundError:
18      print("The file does not exist! Please check the file path.")
19
20
21  try:
22      # Próba odczytu pierwszych 10 znaków
23      with open("plik.txt", "r") as f:
24          print(f.read(10))
25  except FileNotFoundError:
26      print("The file does not exist! Please check the file path.")
27
28  try:
29      # Odczyt w pętli
30      with open("thefile.txt", "r") as f:
31          for x in f:
32              print(x)
33  except FileNotFoundError:
34      print("The file does not exist! Please check the file path.")
35
36  try:
37      # Odczyt wszystkich linii do listy
38      with open("plik.txt", "r") as f:
39          lines = f.readlines() # Wszystkie linie do listy
40          print(lines)
41  except FileNotFoundError:

```

```

except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

try:
    # Odczyt wszystkich linii do listy
    with open("plik.txt", "r") as f:
        lines = f.readlines() # Wszystkie linie do listy
        print(lines)
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

try:
    # Próba zapisu do pliku otwartego w trybie odczytu
    with open("plik.txt", "r") as f:
        f.write("Attempt to write") # Nie można zapisać do pliku otwartego w trybie odczytu
except UnsupportedOperation:
    print("Cannot write to a file opened in read mode!")
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

```

```

from io import UnsupportedOperation

try:
    # Próba odczytu całego pliku
    with open(r'C:\pawel\Dokumenty\03 - studia\Skryptowe Języki programowania\lab06\plik.txt', "r") as f:
        print(f.read())
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")

try:
    # Próba odczytu jednej linii
    with open("plik.txt", "r") as f:
        print(f.readline())
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

try:
    # Próba odczytu pierwszych 10 znaków
    with open("plik.txt", "r") as f:
        print(f.read(10))
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

try:
    # Odczyt w pętli
    with open("thefile.txt", "r") as f:
        for x in f:
            print(x)
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

try:
    # Odczyt wszystkich linii do listy
    with open("plik.txt", "r") as f:
        lines = f.readlines() # Wszystkie linie do listy
        print(lines)

```

```
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")

try:
    # Próba zapisu do pliku otwartego w trybie odczytu
    with open("plik.txt", "r") as f:
        f.write("Attempt to write") # Nie można zapisać do pliku otwartego w trybie
    odczytu
except UnsupportedOperation:
    print("Cannot write to a file opened in read mode!")
except FileNotFoundError:
    print("The file does not exist! Please check the file path.")
```

## Zad 5

**Sprawdzić empirycznie działanie parametrów: a, w oraz x dla funkcji open. Spróbować wpisać zawartość (funkcja write) do utworzonego pliku oraz do nieistniejącego dla tych trzech parametrów. W sprawozdaniu dodać wypełnioną tabelkę**

```
Testing mode 'a': append
Mode 'a': Successfully appended to 'test_append.txt'

Testing mode 'w': write
Mode 'w': Successfully wrote to 'test_write.txt'

Testing mode 'x': exclusive creation
Mode 'x': File 'test_exclusive.txt' already exists, cannot overwrite.

Testing writing to files with each mode:
Testing write in 'r' mode:
Write operation failed: Cannot write to a file opened in 'r' mode.
```



```
lab01.py lab02.py lab03.py lab04.py lab05.py x
1 from io import UnsupportedOperation
2 # Test trybu 'a' - append (dopisywanie)
3 print("Testing mode 'a': append")
4 try:
5     with open("test_append.txt", "a") as f:
6         f.write("Appending content.\n") # Dodaje zawartość do pliku (tworzy, jeśli plik nie istnieje)
7         print("Mode 'a': Successfully appended to 'test_append.txt'")
8 except Exception as e:
9     print(f"Mode 'a': Failed to append to 'test_append.txt' - {e}")
10 # Test trybu 'w' - write (nadpisywanie)
11 print("\nTesting mode 'w': write")
12 try:
13     with open("test_write.txt", "w") as f:
14         f.write("Writing new content.\n") # Nadpisuje plik lub tworzy nowy
15         print("Mode 'w': Successfully wrote to 'test_write.txt'")
16 except Exception as e:
17     print(f"Mode 'w': Failed to write to 'test_write.txt' - {e}")
18
19 # Test trybu 'x' - exclusive creation (tworzenie nowego pliku)
20 print("\nTesting mode 'x': exclusive creation")
21 try:
22     with open("test_exclusive.txt", "x") as f:
23         f.write("Creating new file with content.\n") # Tworzy plik, jeśli nie istnieje
24         print("Mode 'x': Successfully created and wrote to 'test_exclusive.txt'")
25 except FileExistsError:
26     print("Mode 'x': File 'test_exclusive.txt' already exists, cannot overwrite.")
27 except Exception as e:
28     print(f"Mode 'x': Failed to create 'test_exclusive.txt' - {e}")
29
30 # Test próby zapisu do plików
31 print("\nTesting writing to files with each mode:")
32 try:
33     # Próba zapisu do pliku otwartego w trybie 'r'
34     print("Testing write in 'r' mode:")
35     with open("test_append.txt", "r") as f:
36         f.write("This will not work.") # To spowoduje błąd
37 except UnsupportedOperation:
38     print("Write operation failed: Cannot write to a file opened in 'r' mode.")
39 except Exception as e:
40     print(f"Unexpected error: {e}")
41
```

Parametr funkcji open	Plik istnieje	Plik nie istnieje
r	Otwiera plik w trybie do odczytu	Błąd
w	Otwiera plik w trybie do zapisu, tworząc go jeśli nie istnieje	Tworzy nowy plik
a	Otwiera plik w trybie do dopisywania, tworząc go jeśli nie istnieje	Tworzy nowy plik
x	Otwiera plik w trybie do zapisu, ale tylko jeśli plik nie istnieje	Błąd

```
from io import UnsupportedOperation
# Test trybu 'a' - append (dopisywanie)
print("Testing mode 'a': append")
try:
    with open("test_append.txt", "a") as f:
        f.write("Appending content.\n") # Dodaje zawartość do pliku (tworzy, jeśli
plik nie istnieje)
    print("Mode 'a': Successfully appended to 'test_append.txt'")
except Exception as e:
    print(f"Mode 'a': Failed to append to 'test_append.txt' - {e}")
# Test trybu 'w' - write (nadpisywanie)
print("\nTesting mode 'w': write")
try:
    with open("test_write.txt", "w") as f:
        f.write("Writing new content.\n") # Nadpisuje plik lub tworzy nowy
    print("Mode 'w': Successfully wrote to 'test_write.txt'")
except Exception as e:
    print(f"Mode 'w': Failed to write to 'test_write.txt' - {e}")

# Test trybu 'x' - exclusive creation (tworzenie nowego pliku)
print("\nTesting mode 'x': exclusive creation")
try:
    with open("test_exclusive.txt", "x") as f:
        f.write("Creating new file with content.\n") # Tworzy plik, jeśli nie istnieje
    print("Mode 'x': Successfully created and wrote to 'test_exclusive.txt'")
except FileExistsError:
    print("Mode 'x': File 'test_exclusive.txt' already exists, cannot overwrite.")
except Exception as e:
    print(f"Mode 'x': Failed to create 'test_exclusive.txt' - {e}")

# Test próby zapisu do plików
print("\nTesting writing to files with each mode:")
try:
    # Próba zapisu do pliku otwartego w trybie 'r'
    print("Testing write in 'r' mode:")
    with open("test_append.txt", "r") as f:
        f.write("This will not work.") # To spowoduje błąd
except UnsupportedOperation:
    print("Write operation failed: Cannot write to a file opened in 'r' mode.")
except Exception as e:
    print(f"Unexpected error: {e}")
```

## Zad 6 Napisać funkcję, która wyświetli drzewo katalogów dla wybranego katalogu.

```
import os
def display_directory_tree(start_path, indent_level=0): 2 usages new *
    """
    Wyświetla drzewo katalogów dla wybranego katalogu.
    :param start_path: Ścieżka do katalogu początkowego
    :param indent_level: Poziom wcięcia (używany dla rekurencji)
    """

    if not os.path.exists(start_path):
        print("The specified path does not exist.")
        return
    if not os.path.isdir(start_path):
        print(f"The specified path '{start_path}' is not a directory.")
        return
    if indent_level == 0:    # Wyświetlenie katalogu głównego
        print(f"Directory tree for: {start_path}")
    # Iteracja przez zawartość katalogu
    for item in os.listdir(start_path):
        item_path = os.path.join(start_path, item)
        print("    " * indent_level + f"-- {item}")
        # Jeśli element jest katalogiem, rekurencyjnie wyświetl jego zawartość
        if os.path.isdir(item_path):
            display_directory_tree(item_path, indent_level + 1)
# Przykładowe użycie
directory_to_explore = input("Enter the directory path to display its tree: ").strip()
display_directory_tree(directory_to_explore)
```

```
Enter the directory path to display its tree: C:\opencv
Directory tree for: C:\opencv
|-- build
|   |-- bin
|       |-- opencv_videoio_ffmpeg490_64.dll
|   |-- etc
|       |-- haarcascades
|           |-- haarcascade_eye.xml
|           |-- haarcascade_eye_tree_eyeglasses.xml
|           |-- haarcascade_frontalcatface.xml
|           |-- haarcascade_frontalcatface_extended.xml
|           |-- haarcascade_frontalface_alt.xml
|           |-- haarcascade_frontalface_alt2.xml
|           |-- haarcascade_frontalface_alt_tree.xml
|           |-- haarcascade_frontalface_default.xml
|           |-- haarcascade_fullbody.xml
|           |-- haarcascade_lefteye_2splits.xml
|           |-- haarcascade_license_plate_rus_16stages.xml
|           |-- haarcascade_lowerbody.xml
|           |-- haarcascade_profileface.xml
|           |-- haarcascade_righteye_2splits.xml
|           |-- haarcascade_russian_plate_number.xml
|           |-- haarcascade_smile.xml
|           |-- haarcascade_upperbody.xml
|-- lbpcascades
|   |-- lbpcascade_frontalcatface.xml
|   |-- lbpcascade_frontalface.xml
```

```

import os
def display_directory_tree(start_path, indent_level=0):
    """
    Wyświetla drzewo katalogów dla wybranego katalogu.
    :param start_path: Ścieżka do katalogu początkowego
    :param indent_level: Poziom wcięcia (używany dla rekurencji)
    """
    if not os.path.exists(start_path):
        print("The specified path does not exist.")
        return
    if not os.path.isdir(start_path):
        print(f"The specified path '{start_path}' is not a directory.")
        return
    if indent_level == 0:    # Wyświetlenie katalogu głównego
        print(f"Directory tree for: {start_path}")
    # Iteracja przez zawartość katalogu
    for item in os.listdir(start_path):
        item_path = os.path.join(start_path, item)
        print("    " * indent_level + f"|-- {item}")
        # Jeśli element jest katalogiem, rekurencyjnie wyświetl jego zawartość
        if os.path.isdir(item_path):
            display_directory_tree(item_path, indent_level + 1)
# Przykładowe użycie
directory_to_explore = input("Enter the directory path to display its tree: ").strip()
display_directory_tree(directory_to_explore)

```

### 3.7. Zadanie 7.

Sprawdzić działanie funkcji copy z modułu shutil. Wykonać kopiowanie na trzy sposoby:

- plik o danej nazwie nie istnieje w docelowym katalogu,
- plik o podanej nazwie istnieje w docelowym katalogu,
- katalog nie istnieje.

```

import shutil
import os

# Helper function to prepare environment
def setup_environment(): 1usage new*
    # Tworzenie folderów i plików testowych
    if not os.path.exists('source'):
        os.makedirs('source')
    with open('source/test_file.txt', 'w') as f:
        f.write('This is a test file.')

    if not os.path.exists('destination'):
        os.makedirs('destination')
    if os.path.exists('destination/test_file.txt'):
        os.remove('destination/test_file.txt')

# 1. Plik o danej nazwie nie istnieje w katalogu docelowym
def copy_file_no_conflict(): 1usage new*
    print("1. Copying when the target file does not exist.")
    shutil.copy(src='source/test_file.txt', dst='destination/test_file.txt')
    print("Copied successfully.\n")

# 2. Plik o podanej nazwie istnieje w katalogu docelowym
def copy_file_with_conflict(): 1usage new*
    print("2. Copying when the target file exists.")
    with open('destination/test_file.txt', 'w') as f:
        f.write('Existing file content.')
    shutil.copy(src='source/test_file.txt', dst='destination/test_file.txt')
    print("Copied successfully, overwriting the existing file.\n")

```

```

# 3. Katalog docelowy nie istnieje
def copy_file_nonexistent_directory(): 1usage new*
    print("3. Copying when the destination directory does not exist.")
    non_existent_dir = 'non_existent_directory'
    if os.path.exists(non_existent_dir):
        shutil.rmtree(non_existent_dir)
    try:
        shutil.copy(src='source/test_file.txt', os.path.join(non_existent_dir, 'test_file.txt'))
    except FileNotFoundError as e:
        print(f"Error: {e}")
    print("Handled non-existent directory.\n")

# Main script
> if __name__ == "__main__":
    setup_environment()

    # Wykonanie testów
    copy_file_no_conflict()
    copy_file_with_conflict()
    copy_file_nonexistent_directory()

    # Sprzątanie po testach
    shutil.rmtree('source')
    shutil.rmtree('destination')
    if os.path.exists('non_existent_directory'):
        shutil.rmtree('non_existent_directory')

    print("Tests completed.")

```

```

1. Copying when the target file does not exist.
Copied successfully.

```

Jeśli plik docelowy nie istnieje w katalogu docelowym, `shutil.copy` po prostu kopiuje plik z katalogu źródłowego do docelowego.

```

2. Copying when the target file exists.
Copied successfully, overwriting the existing file.

```

W przypadku, gdy plik o tej samej nazwie istnieje już w katalogu docelowym, `shutil.copy` nadpisuje istniejący plik bez ostrzeżenia.

```

1. Copying when the target file does not exist.
Copied successfully.

2. Copying when the target file exists.
Copied successfully, overwriting the existing file.

3. Copying when the destination directory does not exist.
Error: [Errno 2] No such file or directory: 'non_existent_directory\\test_file.txt'
Handled non-existent directory.

```

Jeśli katalog docelowy nie istnieje, `shutil.copy` zgłasza błąd `FileNotFoundError`.

```

import shutil
import os

# Helper function to prepare environment
def setup_environment():
    # Tworzenie folderów i plików testowych
    if not os.path.exists('source'):
        os.makedirs('source')
    with open('source/test_file.txt', 'w') as f:
        f.write('This is a test file.')

    if not os.path.exists('destination'):
        os.makedirs('destination')
    if os.path.exists('destination/test_file.txt'):
        os.remove('destination/test_file.txt')

# 1. Plik o danej nazwie nie istnieje w katalogu docelowym
def copy_file_no_conflict():
    print("1. Copying when the target file does not exist.")
    shutil.copy('source/test_file.txt', 'destination/test_file.txt')
    print("Copied successfully.\n")

# 2. Plik o podanej nazwie istnieje w katalogu docelowym
def copy_file_with_conflict():
    print("2. Copying when the target file exists.")
    with open('destination/test_file.txt', 'w') as f:
        f.write('Existing file content.')
    shutil.copy('source/test_file.txt', 'destination/test_file.txt')
    print("Copied successfully, overwriting the existing file.\n")

# 3. Katalog docelowy nie istnieje
def copy_file_nonexistent_directory():
    print("3. Copying when the destination directory does not exist.")
    non_existent_dir = 'non_existent_directory'
    if os.path.exists(non_existent_dir):
        shutil.rmtree(non_existent_dir)
    try:
        shutil.copy('source/test_file.txt', os.path.join(non_existent_dir,
'test_file.txt'))
    except FileNotFoundError as e:
        print(f"Error: {e}")
    print("Handled non-existent directory.\n")

# Main script
if __name__ == "__main__":
    setup_environment()

    # Wykonanie testów
    copy_file_no_conflict()
    copy_file_with_conflict()
    copy_file_nonexistent_directory()

    # Sprzątanie po testach
    shutil.rmtree('source')
    shutil.rmtree('destination')
    if os.path.exists('non_existent_directory'):
        shutil.rmtree('non_existent_directory')

    print("Tests completed.")

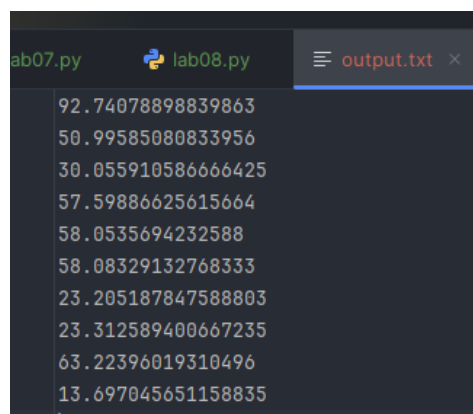
```

Zad 8 Napisać funkcję przyjmującą trzy parametry: licznosc n, dolną granicę przedziału a oraz górną granicę przedziału b. Wylosować n liczb z zakresu . Zapisać te liczby w kolejnych liniach pliku tekstowego

```
import random
def random_num(n, a, b, filename='random_numbers.txt'): 1 usage new *
    #Function to generate n random numbers in the range <a, b> and save them to a file.
    if n <= 0:
        print("The number of random numbers (n) must be positive.")
        return
    if a >= b:
        print("Invalid range: a must be less than b.")
        return
    # Generowanie n losowych liczb
    random_numbers = [random.uniform(a, b) for _ in range(n)]
    # Zapisywanie liczb do pliku tekstowego
    with open(filename, 'w') as file:
        for number in random_numbers:
            file.write(f"{number}\n")
    print(f"{n} random numbers from the range {a} to {b} have been saved to the file {filename}.")
```

```
random_num(n: 10, a: 1, b: 100, filename: 'output.txt')
```

```
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty\03 - studia\Skrypty
10 random numbers from the range 1 to 100 have been saved to the file output.txt.
```

A screenshot of a code editor window with three tabs: 'lab07.py', 'lab08.py', and 'output.txt'. The 'output.txt' tab is active and displays ten lines of floating-point numbers, each on a new line, representing random numbers generated by the program.

```
lab07.py lab08.py output.txt x
92.74078898839863
50.99585080833956
30.055910586666425
57.59886625615664
58.0535694232588
58.08329132768333
23.205187847588803
23.312589400667235
63.22396019310496
13.697045651158835
```

```
import random
def random_num(n, a, b, filename='random_numbers.txt'):
    #Function to generate n random numbers in the range <a, b> and save them to a file.
    if n <= 0:
        print("The number of random numbers (n) must be positive.")
        return
    if a >= b:
        print("Invalid range: a must be less than b.")
        return
    # Generowanie n losowych liczb
    random_numbers = [random.uniform(a, b) for _ in range(n)]
```



```
# Zapisywanie liczb do pliku tekstowego
with open(filename, 'w') as file:
    for number in random_numbers:
        file.write(f"{number}\n")
    print(f"{n} random numbers from the range {a} to {b} have been saved to the file {filename}.")

random_num(10, 1, 100, 'output.txt')
```

**Zad 9** Stworzyć dowolną klasę, której instancje posiadają własną listę, zmienną liczbową i napisową (może to być pracownik, samochód, student, gura geometryczna lub cokolwiek innego, można skorzystać z kodów napisanych na poprzednich zajęciach). Dla instancji stworzonej klasy umożliwić eksport aktualnego stanu instancji do pliku i jego przywrócenie z pliku. Zabezpieczyć program obsługą potencjalnych wyjątków.

```
import json
class Student: 1 usage 1 Pawelgabrieljonca
    def __init__(self, name, age, subjects): # Inicjalizacja instancji studenta new*
        self.name = name # Imię studenta
        self.age = age # Wiek studenta
        self.subjects = subjects # Lista przedmiotów
    def export_to_file(self, filename): # Eksportuje dane instancji do pliku JSON 1 usage 1 Pawelgabrieljonca
        try:
            data = {'name': self.name, 'age': self.age, 'subjects': self.subjects} # Dane instancji
            with open(filename, 'w') as file:
                json.dump(data, file, indent=4) # Zapis danych w formacie JSON
            print(f"Data successfully exported to {filename}.") # Informacja o sukcesie
        except Exception as e:
            print(f"An error occurred during export: {e}") # Obsługa błędów eksportu
    def import_from_file(self, filename): # Importuje dane instancji z pliku JSON 1 usage 1 Pawelgabrieljonca
        try:
            with open(filename, 'r') as file:
                data = json.load(file) # Wczytanie danych z pliku
            self.name = data['name'] # Przywrócenie imienia
            self.age = data['age'] # Przywrócenie wieku
            self.subjects = data['subjects'] # Przywrócenie przedmiotów
            print(f"Data successfully imported from {filename}.") # Informacja o sukcesie
        except FileNotFoundError:
            print(f"The file {filename} does not exist.") # Obsługa braku pliku
        except json.JSONDecodeError:
            print(f"Error decoding JSON from the file {filename}.") # Obsługa błędów formatu JSON
        except KeyError as e:
            print(f"Missing key in data: {e}") # Obsługa brakujących kluczy
        except Exception as e:
            print(f"An unexpected error occurred: {e}") # Obsługa innych wyjątków

if __name__ == "__main__":
    student = Student(name="Alice", age=20, subjects=["Mathematics", "Physics", "Literature"]) # Tworzenie obiektu studenta
    print("Original student data:", student.__dict__) # Wyświetlenie danych przed eksportem
    student.export_to_file("student_data.json") # Eksport danych do pliku
    student.name = "Bob" # Modyfikacja imienia
    student.age = 22 # Modyfikacja wieku
    student.subjects = ["History"] # Modyfikacja przedmiotów
    print("Modified student data:", student.__dict__) # Wyświetlenie zmodyfikowanych danych
    student.import_from_file("student_data.json") # Import danych z pliku
    print("Restored student data:", student.__dict__) # Wyświetlenie przywróconych danych
```

```
Original student data: {'name': 'Alice', 'age': 20, 'subjects': ['Mathematics', 'Physics', 'Literature']}
Data successfully exported to student_data.json.
Modified student data: {'name': 'Bob', 'age': 22, 'subjects': ['History']}
Data successfully imported from student_data.json.
Restored student data: {'name': 'Alice', 'age': 20, 'subjects': ['Mathematics', 'Physics', 'Literature']}
```

Eksport danych (export\_to\_file):



Dane instancji są przekształcane do słownika za pomocą Pythonowego typu dict.

Słownik jest zapisywany do pliku JSON za pomocą json.dump.

Import danych (import\_from\_file):

Dane są wczytywane z pliku JSON za pomocą json.load, co automatycznie przekształca je z formatu tekstowego na Pythonowy słownik.

Atrybuty obiektu są aktualizowane wartościami z wczytanego słownika.



```
import json
class Student:
    def __init__(self, name, age, subjects): # Inicjalizacja instancji studenta
        self.name = name # Imię studenta
        self.age = age # Wiek studenta
        self.subjects = subjects # Lista przedmiotów
    def export_to_file(self, filename): # Eksportuje dane instancji do pliku JSON
        try:
            data = {'name': self.name, 'age': self.age, 'subjects': self.subjects} #
Dane instancji
            with open(filename, 'w') as file:
                json.dump(data, file, indent=4) # Zapis danych w formacie JSON
            print(f"Data successfully exported to {filename}.") # Informacja o sukce-
sie
        except Exception as e:
            print(f"An error occurred during export: {e}") # Obsługa błędów eksportu
    def import_from_file(self, filename): # Importuje dane instancji z pliku JSON
        try:
            with open(filename, 'r') as file:
                data = json.load(file) # Wczytanie danych z pliku
                self.name = data['name'] # Przywrócenie imienia
                self.age = data['age'] # Przywrócenie wieku
                self.subjects = data['subjects'] # Przywrócenie przedmiotów
            print(f"Data successfully imported from {filename}.") # Informacja o suk-
cesie
        except FileNotFoundError:
            print(f"The file {filename} does not exist.") # Obsługa braku pliku
        except json.JSONDecodeError:
            print(f"Error decoding JSON from the file {filename}.") # Obsługa błędów
formatu JSON
        except KeyError as e:
            print(f"Missing key in data: {e}") # Obsługa brakujących kluczy
        except Exception as e:
            print(f"An unexpected error occurred: {e}") # Obsługa innych wyjątków
if __name__ == "__main__":
    student = Student("Alice", 20, ["Mathematics", "Physics", "Literature"]) #
Tworzenie obiektu studenta
    print("Original student data:", student.__dict__) # Wyświetlenie danych przed ek-
sportem
    student.export_to_file("student_data.json") # Eksport danych do pliku
    student.name = "Bob" # Modyfikacja imienia
    student.age = 22 # Modyfikacja wieku
    student.subjects = ["History"] # Modyfikacja przedmiotów
    print("Modified student data:", student.__dict__) # Wyświetlenie zmodyfikowanych
danych
    student.import_from_file("student_data.json") # Import danych z pliku
```

```
print("Restored student data:", student.__dict__) # Wyświetlenie przywróconych danych
```

**Zad 10** Utworzyć nową funkcję, która będzie działać na dwóch plikach jednocześnie. W pierwszym pliku zapisać kilka liczb (można wykorzystać zadanie 8). Do drugiego pliku przepisać te same liczby, jednak zapisać je w postaci binarnej.

```
import struct

def save_num(text_file, binary_file, numbers): 1usage new *
    try:
        # Zapis liczb w pliku tekstowym
        with open(text_file, 'w') as txt_file:
            for number in numbers:
                txt_file.write(f"{number}\n") # Każda liczba w nowej linii
        print(f"Numbers successfully written to {text_file}.") # Informacja o sukcesie

        # Zapis liczb w pliku binarnym
        with open(binary_file, 'wb') as bin_file:
            for number in numbers:
                bin_file.write(struct.pack( fmt: 'i', *v: number)) # Zapis liczby jako 4-bajtowego int
            print(f"Numbers successfully written in binary format to {binary_file}.") # Informacja o sukcesie

    except Exception as e:
        print(f"An error occurred: {e}") # Obsługa błędów

# Przykład użycia funkcji
if __name__ == "__main__":
    numbers = [12, 34, 56, 78, 90] # Lista liczb
    save_num( text_file: "numbers.txt", binary_file: "numbers.bin", numbers) # Wywołanie funkcji
```

	lab10.py	numbers.txt	numbers.bin
S	1	12	
	2	34	
	3	56	
	4	78	
	5	90	
	6		

lab10.py numbers.txt numbers.bin

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

0C 00 00 00 22 00 00 00 38 00 00 00 4E 00 00 00

5A 00 00 00

Binary

☐ ☐ ☐ ☐ ☒ ☐ ☐

Byte

12

Word

3072

Integer

201326592

Long

864691129025560576

Float

9.8607613E-32

Double

6.983508373829698E-251

Character

String

```

import struct
def save_num(text_file, binary_file, numbers):
    try:
        # Zapis liczb w pliku tekstowym
        with open(text_file, 'w') as txt_file:
            for number in numbers:
                txt_file.write(f"{number}\n") # Każda liczba w nowej linii
            print(f"Numbers successfully written to {text_file}.") # Informacja o sukcesie

        # Zapis liczb w pliku binarnym
        with open(binary_file, 'wb') as bin_file:
            for number in numbers:
                bin_file.write(struct.pack('i', number)) # Zapis liczby jako 4-ba-
jtowego int
            print(f"Numbers successfully written in binary format to {binary_file}.") #
Informacja o sukcesie

    except Exception as e:
        print(f"An error occurred: {e}") # Obsługa błędów

# Przykład użycia funkcji
if __name__ == "__main__":
    numbers = [12, 34, 56, 78, 90] # Lista liczb
    save_num("numbers.txt", "numbers.bin", numbers) # Wywołanie funkcji

```

## Wnioski:

Ćwiczenia dotyczące obsługi wyjątków i pracy z plikami w Pythonie pozwoliły mi lepiej zrozumieć, jak zarządzać błędami i zapewnić stabilność aplikacji. Dzięki blokom try, except, else i finally nauczyłem się skutecznie obsługiwać błędy, co jest istotne, zwłaszcza przy interakcji z użytkownikami i zewnętrznymi źródłami danych. Praktyczne zadania związane z otwieraniem, odczytem i zapisem plików pokazały mi, jak ważne jest odpowiednie zarządzanie zasobami oraz unikanie wycieków pamięci przez poprawne zamykanie plików.