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

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznieZad1

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

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

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

Przecinka na kropkę kod działa. :

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

*try*:  
 x = int(input("Enter a number:"))  
 result = 10 /x  
*except* ValueError:*#obsługa błędu konwersji na liczbe całkowita* print("Something went wrong")  
*except* ZeroDivisionError:*#obłsuga 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.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

*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**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie**

|  |  |  |
| --- | --- | --- |
| 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.**

**Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

*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)

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie



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

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

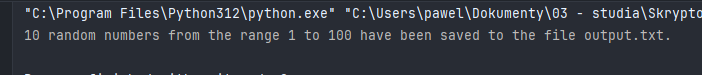
Jeśli katalog docelowy nie istnieje, shutil.copy zgłasza błąd FileNotFoundError.

*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: liczność n, dolną granicę przedziału a oraz górną granicę przedziału b. Wylosować n liczb z zakresu . Zapisać te liczby w kolejnych liniach pliku tekstowego**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie**

****

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie**

*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.**

**Obraz zawierający tekst, zrzut ekranu, menu

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

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.

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie**

*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 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  
 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("Alice", 20, ["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*

**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.**

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający oprogramowanie, Oprogramowanie multimedialne, Oprogramowanie graficzne, Edycja

Opis wygenerowany automatycznie

*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-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("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.