|  |  |  |  |
| --- | --- | --- | --- |
| Obraz zawierający symbol, design  Opis wygenerowany automatycznie | Politechnika Bydgoska im. J. J. Śniadeckich  **Wydział Telekomunikacji,**  **Informatyki i Elektrotechniki** | | Obraz zawierający symbol, logo, Grafika, Czcionka  Opis wygenerowany automatycznie |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | *Napisy* | | |
| **Student** | Marcin Ogórkiewicz | | |
| **Nr ćw.** | 7 | **Data wykonania** | 14.11.2023 |
| **Ocena** |  | **Data oddania spr.** | 14.11.2023 |

**Zadanie 1**

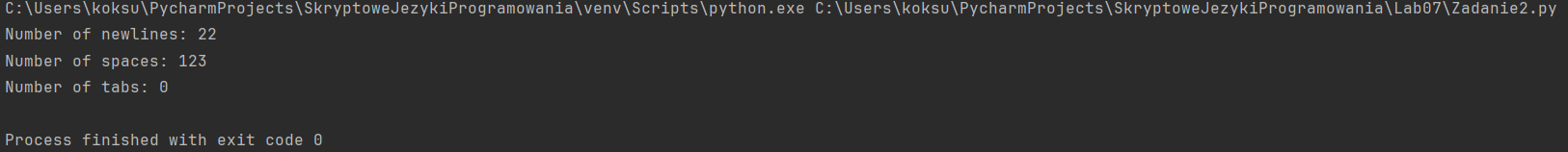
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
# stałe i zmienne globalne  
  
# funkcje  
  
  
  
def process\_text(file\_path: str) -> None:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 lines = file.readlines()  
 total\_lines = len(lines)  
 total\_words = 0  
 total\_characters = 0  
 for i, line in enumerate(lines, start=1):  
 line = line.strip()  
 num\_characters = len(line)  
 num\_words = len(line.split())  
 print(f"Wiersz {i}: Ilość znaków: {num\_characters}, Ilość słów: {num\_words}")  
 total\_words += num\_words  
 total\_characters += num\_characters  
 print("\nIlość wierszy:", total\_lines)  
 print("Ilość wszystkich słów:", total\_words)  
 print("Ilość wszystkich znaków:", total\_characters)  
  
  
def main() -> None:  
 process\_text("inwokacja.txt")  
  
  
main()

**Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie**

**Zadanie 2**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def count\_whitespaces(file\_path: str) -> None:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 file\_text = file.read()  
 num\_newlines = file\_text.count('\n')  
 num\_spaces = file\_text.count(' ')  
 num\_tabs = file\_text.count('\t')  
 print("Number of newlines:", num\_newlines)  
 print("Number of spaces:", num\_spaces)  
 print("Number of tabs:", num\_tabs)  
  
  
def main() -> None:  
 count\_whitespaces("inwokacja.txt")  
  
  
main()



**Zadanie 3**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def replace\_dead\_stops(file\_path: str) -> None:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 original\_text = file.read()  
 lines = original\_text.split('\n')  
 modified\_lines = []  
 for line in lines:  
 words = line.split()  
 modified\_words = []  
 for word in words:  
 if '.' in word and '...' not in word:  
 word = word.replace('.', '...')  
 modified\_words.append(word)  
 modified\_lines.append(' '.join(modified\_words))  
 modified\_text = '\n'.join(modified\_lines)  
 print(modified\_text)  
  
  
def main() -> None:  
 replace\_dead\_stops("inwokacja.txt")  
  
  
main()

**Obraz zawierający tekst, zrzut ekranu, design

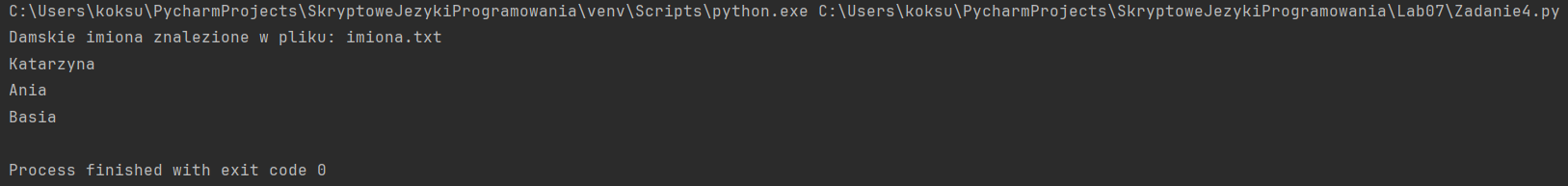
Opis wygenerowany automatycznie**

**Zadanie 4**

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

Opis wygenerowany automatycznie**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import re  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def find\_female\_names(file\_path: str) -> None:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 text = file.read()  
 pattern = r'\b[A-Z][a-z]\*a\b'  
 female\_names = re.findall(pattern, text)  
 print(f"Damskie imiona znalezione w pliku: {file\_path}")  
 for name in female\_names:  
 print(name)  
  
  
def main() -> None:  
 find\_female\_names("imiona.txt")  
  
  
main()

****

**Zadanie 5**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import re  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def polish\_phone\_numbers(file\_path):  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 text = file.readlines()  
 pattern = r'(\+48|0048)'  
 polish\_numbers\_list = []  
 for number in text:  
 if re.match(pattern, number):  
 polish\_numbers\_list.append(number)  
 print("Polskie numery telefoniczne znalezione w pliku:")  
 for number in polish\_numbers\_list:  
 print(number)  
  
  
def main() ->None:  
 polish\_phone\_numbers("numery.txt")  
  
  
main()

**Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie**

**Zadanie 6**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import re  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def read\_phone\_numbers(file\_path: str) -> list:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 numbers = [line.strip() for line in file.readlines()]  
  
 return numbers  
  
  
def validate\_phone\_numbers(numbers: list) -> dict:  
 validation\_patterns = {  
 "1. xxx-xxx-xxx": r'^\d{3}-\d{3}-\d{3}$',  
 "2. xxxxxxxxx": r'^\d{9}$',  
 "3. +48xxxxxxxxx": r'^\+48\d{9}$',  
 "4. 0048xxxxxxxxx": r'^0048\d{9}$',  
 "5. +48 xxx xxx xxx": r'^\+48\s\d{3}\s\d{3}\s\d{3}$',  
 }  
 counts = {key: 0 for key in validation\_patterns}  
 for number in numbers:  
 for pattern\_description, pattern in validation\_patterns.items():  
 if re.match(pattern, number):  
 counts[pattern\_description] += 1  
 break  
 return counts  
  
  
def main() -> None:  
 file\_path = 'numery.txt'  
 phone\_numbers = read\_phone\_numbers(file\_path)  
 counts = validate\_phone\_numbers(phone\_numbers)  
 print("Ilość numerów telefonu o danym formacie:")  
 for pattern\_description, count in counts.items():  
 print(f"{pattern\_description}: {count} numery")  
  
  
main()

**Obraz zawierający tekst, zrzut ekranu, Czcionka

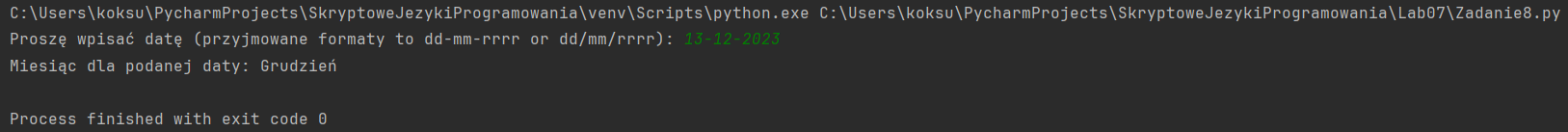
Opis wygenerowany automatycznie**

**Zadanie 7**

*r'^[a-z][a-z0-9\_.-]\*@gmail\.com$'*

**Zadanie 8**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import re  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def validate\_date(input\_date: str) -> None:  
 date\_pattern = re.compile(r'^(0[1-9]|[12][0-9]|3[01])[-/](0[1-9]|1[0-2])[-/](\d{4})$')  
 match = re.match(date\_pattern, input\_date)  
  
 if match:  
 day, month, year = map(int, match.groups())  
 if 1 <= day <= 31 and 1 <= month <= 12:  
 months = [  
 "Styczeń", "Luty", "Marzec", "Kwiecień",  
 "Maj", "Czerwiec", "Lipiec", "Sierpień",  
 "Wrzesień", "Październik", "Listopad", "Grudzień"  
 ]  
 print(f"Miesiąc dla podanej daty: {months[month - 1]}")  
 else:  
 print("Niepoprawna ilość dni lub miesięcy")  
 else:  
 print("Niepoprawny format daty")  
  
  
def main() -> None:  
 user\_input = input("Proszę wpisać datę (przyjmowane formaty to dd-mm-rrrr or dd/mm/rrrr): ")  
 validate\_date(user\_input)  
  
  
main()

****

**Zadanie 9**

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

Opis wygenerowany automatycznie**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import re  
import os  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def list\_txt\_files(directory\_path: str) -> None:  
 txt\_files\_pattern = re.compile(r'.\*\.txt$', re.IGNORECASE)  
 txt\_files = [file for file in os.listdir(directory\_path) if re.match(txt\_files\_pattern, file)]  
 if txt\_files:  
 print(f"Pliki tekstowe w katalogu {directory\_path}:")  
 for txt\_file in txt\_files:  
 print(txt\_file)  
 else:  
 print(f"W katalogu {directory\_path} nie znaleziono żadnych plików tekstowych.")  
  
  
def main() -> None:  
 directory\_path = input("Wprowadź ścieżkę dostępu: ")  
 if os.path.isdir(directory\_path):  
 list\_txt\_files(directory\_path)  
 else:  
 print(f"{directory\_path} nie jest poprawną ścieżką dostępu.")  
  
  
main()

Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

**Zadanie 10**

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

Opis wygenerowany automatycznie**

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import re  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def process\_strings(file\_path):  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 lines = file.readlines()  
 end\_with\_x\_or\_y = []  
 three\_chars\_start\_with\_a = []  
 starts\_with\_vowel = []  
 for line in lines:  
 line = line.strip()  
 if re.search(r'[xy]$', line):  
 end\_with\_x\_or\_y.append(line)  
 if re.match(r'^a\w{2}$', line):  
 three\_chars\_start\_with\_a.append(line)  
 if re.match(r'^[aeiouyAEIOUY]', line):  
 starts\_with\_vowel.append(line)  
 print("Elements ending with x or y:")  
 print(end\_with\_x\_or\_y)  
 print("\nThree-character elements starting with a:")  
 print(three\_chars\_start\_with\_a)  
 print("\nElements starting with a vowel:")  
 print(starts\_with\_vowel)  
  
  
def main() -> None:  
 file\_path = 'napisy.txt'  
 process\_strings(file\_path)  
  
  
main()

**Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie**

**Wnioski**

Ćwiczenie nauczyło mnie pracy na ciągach znaków przy pomocy funkcji wbudowanych oraz bilbioteki re.