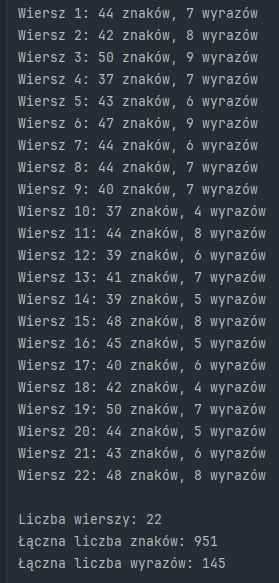
|  |  |  |  |
| --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji,  Informatyki i Elektrotechniki  **Zakład Systemów Teleinformatycznych** | |  |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | Napisy | | |
| **Student** | Paweł Jońca | | |
| **Nr lab.** | 7 | **Data wykonania** | 01.12.2024 |
| **Ocena** |  | **Data oddania spr.** | 01.12.2024 |

Zad 1

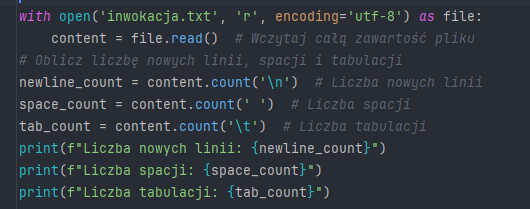
Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

*with* open('inwokacja.txt', 'r', encoding='utf-8') *as* file:  
 lines = file.readlines()  
*# Usuń białe znaki z lewej i prawej strony*lines = [line.strip() *for* line *in* lines]  
  
*# Liczba wierszy*line\_count = len(lines)  
  
*# Inicjalizuj zmienne do sumowania znaków i wyrazów*total\_bag = 0  
total\_words = 0  
  
*for* i, line *in* enumerate(lines, start=1):*# Przetwarzaj każdy wiersz* char\_count = len(line) *# Liczba znaków w wierszu* word\_now = len(line.split()) *# Liczba wyrazów w wierszu* total\_bag += char\_count  
 total\_words += word\_now  
 print(f"Wiersz {i}: {char\_count} znaków, {word\_now} wyrazów")  
  
print(f"\nLiczba wierszy: {line\_count}")  
print(f"Łączna liczba znaków: {total\_bag}")  
print(f"Łączna liczba wyrazów: {total\_words}")



Zad 2



Obraz zawierający tekst, Czcionka, zrzut ekranu, typografia

Opis wygenerowany automatycznie

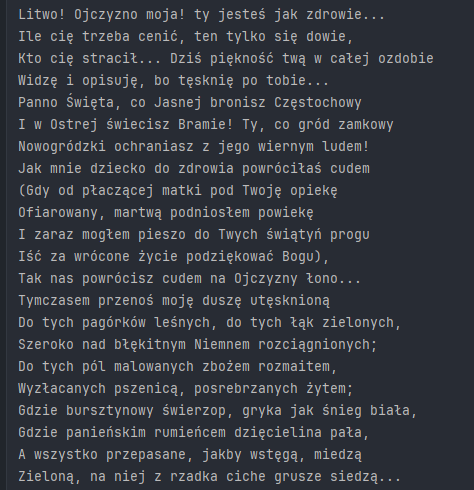
*with* open('inwokacja.txt', 'r', encoding='utf-8') *as* file:  
 content = file.read() *# Wczytaj całą zawartość pliku  
# Oblicz liczbę nowych linii, spacji i tabulacji*newline\_count = content.count('\n') *# Liczba nowych linii*space\_count = content.count(' ') *# Liczba spacji*tab\_count = content.count('\t') *# Liczba tabulacji*print(f"Liczba nowych linii: {newline\_count}")  
print(f"Liczba spacji: {space\_count}")  
print(f"Liczba tabulacji: {tab\_count}")

Zad 3

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, Czcionka

Opis wygenerowany automatycznie

*# Otwórz plik do odczytu  
with* open('inwokacja.txt', 'r', encoding='utf-8') *as* file:  
 content = file.read() *# Wczytaj zawartość pliku  
# Zamień wszystkie kropki na wielokropek, pozostawiając wielokropek bez zmian*modified\_content = content.replace('...', '<<<TEMP>>>') *# Tymczasowa zamiana wielokropków*modified\_content = modified\_content.replace('.', '...') *# Zamień kropki na wielokropki*modified\_content = modified\_content.replace('<<<TEMP>>>', '...') *# Przywróć oryginalne wielokropki*print(modified\_content)



Wczytanie zawartości pliku: Zawartość pliku jest wczytywana do zmiennej Content Tymczasowa zamiana wielokropków: Aby nie zamieniać już istniejących wielokropków na kolejne wielokropki, zastępujemy je tymczasowym znacznikiem ( ). Zamiana kropek na wielokropki: Zwykłe kropki są zamieniane na wielokropki. Przywrócenie oryginalnych wielokropków: Znaczniki tymczasowe są zamieniane z powrotem na wielokropki. Wypisanie nowej zawartości: Na końcu zmodyfikowany tekst jest wypisywany na ekranie.

Zad 4

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Lista names: Przechowuje dowolne imiona.

Wyrażenie regularne r'^[A-Z][a-z]\*$':

* [A-Z]: Imię musi zaczynać się wielką literą.
* [a-z]\*: Dowolna liczba małych liter.

List comprehension: Iteruje po liście names, wybierając imiona pasujące do wzorca.

Wynik: Zmienia female\_names przechowujące listę imion damskich, które spełniają kryteria.

*import* re *# Import modułu do wyrażeń regularnych  
  
# Lista imion*names = ["Paweł", "Adam", "Kacper", "Filip", "Szymon", "Artur", "Marcelina", "Kasia", "Kamila"]  
  
*# Korzystając z wyrażeń regularnych, znajdź imiona damskie*female\_names = [name *for* name *in* names *if* re.match(r'^[A-Z][a-z]\*a$', name)]  
  
*# Wypisz imiona damskie*print("Imiona damskie rozpoczynające się wielką literą i kończące się na 'a':")  
print(female\_names)

Zad 5

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, Czcionka, zrzut ekranu

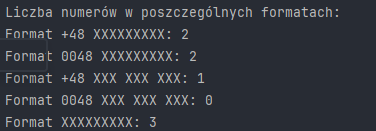
Opis wygenerowany automatycznie

1. Wczytanie zawartości pliku:
   * Każda linia z pliku jest wczytywana jako element listy phone\_numbers.
2. Wyrażenie regularne ^(\+48|8048):
   * ^: Numer musi zaczynać się od początku linii.
   * (\+48|8048): Numer musi zaczynać się od +48 lub 8048.
3. strip(): Usuwa zbędne białe znaki z początku i końca linii.
4. Lista polish\_numbers: Przechowuje tylko numery, które spełniają warunki.
5. Wyświetlenie numerów: Każdy polski numer jest wypisywany w osobnej linii.

*import* re *# Import modułu do wyrażeń regularnych  
# Otwórz plik i wczytaj jego zawartość  
with* open('numery.txt', 'r', encoding='utf-8') *as* file:  
 phone\_numbers = file.readlines() *# Wczytaj wszystkie linie jako listę  
# Usuń białe znaki i znajdź polskie numery telefonów*polish\_numbers = [num.strip() *for* num *in* phone\_numbers *if* re.match(r'^(?:\+48|0048)', num.strip())]  
*# Wypisz polskie numery telefonów*print("Polskie numery telefonów:")  
*for* number *in* polish\_numbers:  
 print(number)

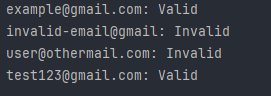
Zad 6





*import* re *# Import modułu do wyrażeń regularnych  
# Wczytaj numery telefonów z pliku  
with* open('numery.txt', 'r', encoding='utf-8') *as* file:  
 phone\_numbers = [line.strip() *for* line *in* file.readlines()] *# Usuń białe znaki  
# Propozycja wyrażeń regularnych dla różnych formatów*patterns = {  
 "Format +48 XXXXXXXXX": r'^\+48\d{9}$', *# +48 bez spacji, 9 cyfr* "Format 0048 XXXXXXXXX": r'^0048\d{9}$', *# 0048 bez spacji, 9 cyfr* "Format +48 XXX XXX XXX": r'^\+48 \d{3} \d{3} \d{3}$', *# +48 z trzema blokami cyfr* "Format 0048 XXX XXX XXX": r'^0048 \d{3} \d{3} \d{3}$', *# 0048 z trzema blokami cyfr* "Format XXXXXXXXX": r'^\d{9}$', *# 9 cyfr, bez prefiksu*}  
*# Zliczanie numerów w każdym formacie*format\_counts = {key: 0 *for* key *in* patterns.keys()}  
*# Sprawdzenie numerów i przypisanie do formatu  
for* number *in* phone\_numbers:  
 *for* format\_name, pattern *in* patterns.items():  
 *if* re.match(pattern, number):  
 format\_counts[format\_name] += 1  
 *break # Gdy znajdziemy dopasowanie, kończymy iterację*print("Liczba numerów w poszczególnych formatach:")  
*for* format\_name, count *in* format\_counts.items():  
 print(f"{format\_name}: {count}")

Zad 7



*import* re  
*# Wyrażenie regularne dla adresów e-mail Gmail*gmail\_regex = r"^[a-zA-Z0-9.\_%+-]+@gmail\.com$"  
*# Funkcja walidacji  
def* validate\_gmail(email):  
 *return* bool(re.match(gmail\_regex, email))  
emails = ["example@gmail.com", "invalid-email@gmail", "user@othermail.com", "test123@gmail.com"]  
*for* email *in* emails:  
 print(f"{email}: {'Valid' *if* validate\_gmail(email) *else* 'Invalid'}")

^: Początek ciągu.

[a-zA-Z0-9.\_%+-]+: Dopuszczalne znaki przed @ (litery, cyfry, ., \_, %, +, -).

@: Znak "małpy".

gmail\.com: Dosłowny ciąg "gmail.com".

$: Koniec ciągu.

Zad 8

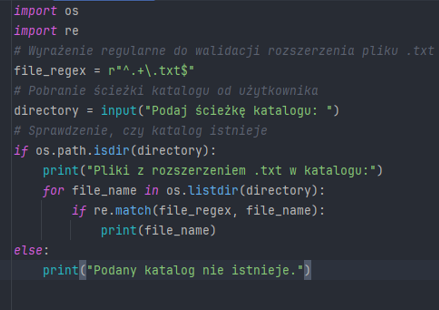


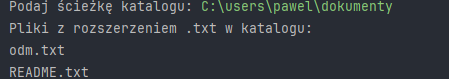
Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

*import* re  
*# Wyrażenie regularne do walidacji daty w formacie DD-MM-YYYY*date\_regex = r"^(0[1-9]|[12][0-9]|3[01])-(0[1-9]|1[0-2])-(\d{4})$"  
months = {  
 "01": "styczeń",  
 "02": "luty",  
 "03": "marzec",  
 "04": "kwiecień",  
 "05": "maj",  
 "06": "czerwiec",  
 "07": "lipiec",  
 "08": "sierpień",  
 "09": "wrzesień",  
 "10": "październik",  
 "11": "listopad",  
 "12": "grudzień"  
}  
*# Pobranie daty od użytkownika*user\_date = input("Podaj datę w formacie DD-MM-YYYY: ")  
*# Walidacja daty*match = re.match(date\_regex, user\_date)  
*if* match:  
 day, month, year = match.groups()  
 print(f"Podana data jest poprawna. Miesiąc to: {months[month]}")  
*else*:  
 print("Podana data jest niepoprawna.")

Zad 9





*import* os  
*import* re  
*# Wyrażenie regularne do walidacji rozszerzenia pliku .txt*file\_regex = r"^.+\.txt$"  
*# Pobranie ścieżki katalogu od użytkownika*directory = input("Podaj ścieżkę katalogu: ")  
*# Sprawdzenie, czy katalog istnieje  
if* os.path.isdir(directory):  
 print("Pliki z rozszerzeniem .txt w katalogu:")  
 *for* file\_name *in* os.listdir(directory):  
 *if* re.match(file\_regex, file\_name):  
 print(file\_name)  
*else*:  
 print("Podany katalog nie istnieje.")

^: Początek nazwy pliku.

.+: Co najmniej jeden dowolny znak (nazwa pliku przed rozszerzeniem).

\.txt: Rozszerzenie .txt (kropka jest ucieczkowana, aby była traktowana dosłownie).

$: Koniec ciągu.

1. **Sprawdzenie katalogu**: Skrypt sprawdza, czy podana ścieżka istnieje i jest katalogiem.
2. **Lista plików**: Iteruje po plikach w katalogu za pomocą os.listdir.
3. **Filtrowanie plików**.txt: Używa wyrażenia regularnego do sprawdzenia, czy nazwa pliku kończy się na .txt.
4. **Wypisanie wyników**: Jeśli pliki .txt są znalezione, ich nazwy zostają wypisane

Zad 10

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*import* re  
*# Lista 10 napisów*words = ["lis", "drzewo", "ananas", "pies", "pan", "orzechy", "lizak", "telefon", "papier", "myszka"]  
*# Elementy kończące się na x lub y*ends\_with\_x\_or\_y = [word *for* word *in* words *if* re.search(r"[xy]$", word)]  
*# Elementy trzyznakowe zaczynające się od a*three\_char\_starting\_a = [word *for* word *in* words *if* re.search(r"^a.{2}$", word)]  
*# Elementy rozpoczynające się samogłoską*starts\_with\_vowel = [word *for* word *in* words *if* re.search(r"^[aeiouAEIOU]", word)]  
*# Wyniki*print("Elementy kończące się na 'x' lub 'y':", ends\_with\_x\_or\_y)  
print("Elementy trzyznakowe zaczynające się od 'a':", three\_char\_starting\_a)  
print("Elementy rozpoczynające się samogłoską:", starts\_with\_vowel)

Wnioski:

Podczas zajęć nauczyłem się efektywnie pracować z napisami w Pythonie, korzystając z wbudowanych funkcji oraz modułu re do zaawansowanych operacji. Wyrażenia regularne okazały się niezwykle przydatne, zwłaszcza przy walidacji danych, takich jak adresy e-mail, daty czy numery telefonów.