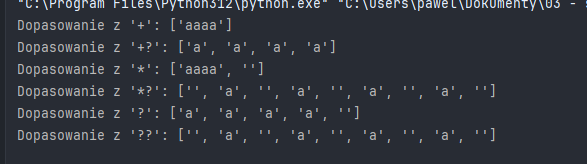
|  |  |  |  |
| --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji,  Informatyki i Elektrotechniki  **Zakład Systemów Teleinformatycznych** | |  |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | Regex – drugie starcie | | |
| **Student** | Paweł Jońca | | |
| **Nr lab.** | 8 | **Data wykonania** | 06.12 |
| **Ocena** |  | **Data oddania spr.** | 08.12 |

Zad 1

1. Kwantykator +(zachłanny) Dopasowuje jeden lub więcej wystąpień poprzedniego symbolu, maksymalizując dopasowanie
2. +?(leniwy) Dopasowuje jedne lub więcej wystąpień poprzedniego symbolu ale minimalizuje dopasowanie
3. \*(zachłanny) Dopasowuje jeden lub więcej wystąpień poprzedniego symbolu maksymalizując dopasowanie
4. \*?(leniwy) Dopasowuje zero lub więcej wystąpień poprzedniego symbolu, ale minimalizuje dopasowanie.
5. ?(zachłanny) Dopasowuje zero lub jedno wystąpienie poprzedniego symbolu.
6. ??(leniwy) Dopasowuje zero lub jedno wystąpienie poprzedniego symbolu, minimalizując dopasowanie.



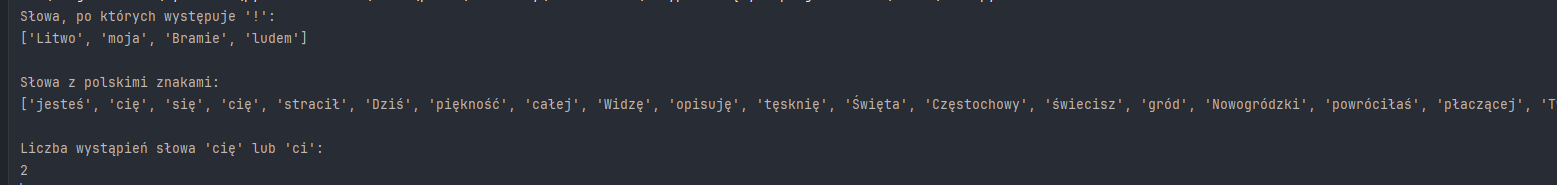


*import* re  
  
*# Test z kwantyfikatorem "+"*print("Dopasowanie z '+':", re.findall('a+', 'aaaa'))  
  
*# Test z kwantyfikatorem "+?" (leniwy)*print("Dopasowanie z '+?':", re.findall('a+?', 'aaaa'))  
  
*# Test z kwantyfikatorem "\*"*print("Dopasowanie z '\*':", re.findall('a\*', 'aaaa'))  
  
*# Test z kwantyfikatorem "\*?" (leniwy)*print("Dopasowanie z '\*?':", re.findall('a\*?', 'aaaa'))  
  
*# Test z kwantyfikatorem "?"*print("Dopasowanie z '?':", re.findall('a?', 'aaaa'))  
  
*# Test z kwantyfikatorem "??" (leniwy)*print("Dopasowanie z '??':", re.findall('a??', 'aaaa'))

Zad 2

Obraz zawierający zrzut ekranu, tekst, oprogramowanie

Opis wygenerowany automatycznie



*import* re  
*# Funkcja do odczytania pliku i przetworzenia go  
def* process\_inwokacja\_file():  
 *with* open('inwokacja.txt', 'r', encoding='utf-8') *as* file:  
 text = file.read()  
 *# 1. Wypisać słowa, po których występuje „!”* words\_after = re.findall(r'\b\w+(?=\!)', text)  
 print("Słowa, po których występuje '!':")  
 print(words\_after)  
  
 *# 2. Wypisać słowa z polskimi znakami* polish\_words = re.findall(r'\b\w\*[\u0104\u0106\u0118\u0119\u0141\u0142\u0143\u015A\u015B\u0179\u017A\u017C\u017C\u00F3\u00F1\u00F3]\w\*\b', text)  
 print("\nSłowa z polskimi znakami:")  
 print(polish\_words)  
  
 *# 3. Zliczyć wystąpienia słowa "cię" lub "ci"* ci\_cie = len(re.findall(r'\bci[ęe]\b', text))  
 print("\nLiczba wystąpień słowa 'cię' lub 'ci':")  
 print(ci\_cie)  
*# Uruchomienie funkcji*process\_inwokacja\_file()

Zad 3

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

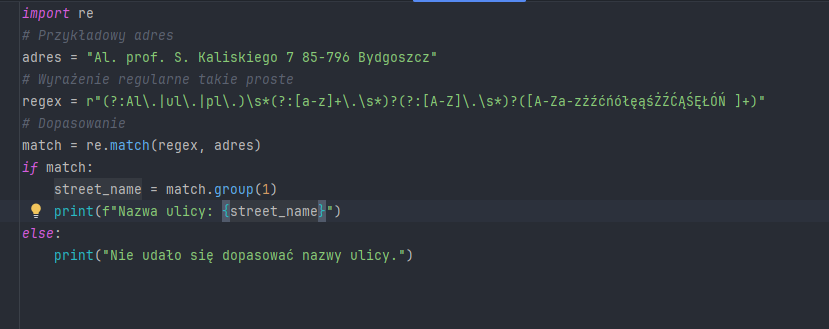
Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*import* re  
*# Wczytuje plik*file\_path = "adresy.txt"  
*with* open(file\_path, "r", encoding="utf-8") *as* file:  
 data = file.readlines()  
*# Definicja wyrażenia regularnego o taka nie długa*regex = r"ul\. ([A-Za-zżźćńółęąśŻŹĆĄŚĘŁÓŃ ]+) (\d+)(?:/(\d+))? (\d{2}-\d{3}) ([A-Za-zżźćńółęąśŻŹĆĄŚĘŁÓŃ ]+)"  
*# Przetwarzanie adresow z uzyciem peli  
for* line *in* data:  
 match = re.match(regex, line.strip())  
 *if* match:  
 ulica = match.group(1)  
 numer\_domu = match.group(2)  
 numer\_mieszkania = match.group(3) *if* match.group(3) *else* "brak"  
 kod\_pocztowy = match.group(4)  
 miasto = match.group(5)  
 print(f"Ulica: {ulica}")  
 print(f"Numer domu: {numer\_domu}")  
 print(f"Numer mieszkania: {numer\_mieszkania}")  
 print(f"Kod pocztowy: {kod\_pocztowy}")  
 print(f"Miasto: {miasto}")  
 print()  
 *else*:  
 print(f"Nie udało się dopasować adresu w linii: {line.strip()}")

Zad 4

 Obraz zawierający tekst, Czcionka, zrzut ekranu, typografia

Opis wygenerowany automatycznie

*import* re  
*# Przykładowy adres*adres = "Al. prof. S. Kaliskiego 7 85-796 Bydgoszcz"  
*# Wyrażenie regularne takie proste*regex = r"(?:Al\.|ul\.|pl\.)\s\*(?:[a-z]+\.\s\*)?(?:[A-Z]\.\s\*)?([A-Za-zżźćńółęąśŻŹĆĄŚĘŁÓŃ ]+)"  
*# Dopasowanie*match = re.match(regex, adres)  
*if* match:  
 nazwa\_ulicy = match.group(1)  
 print(f"Nazwa ulicy: {nazwa\_ulicy}")  
*else*:  
 print("Nie udało się dopasować nazwy ulicy.")

Zad 5

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

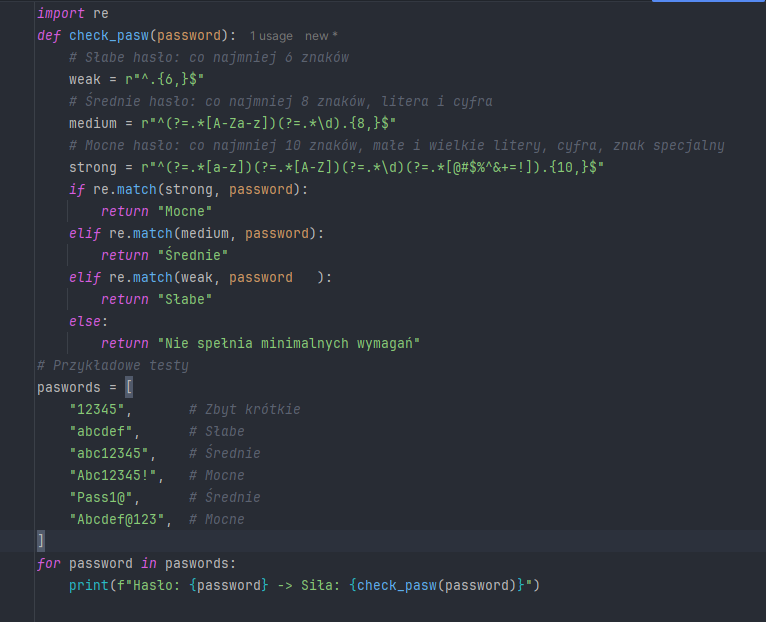
Opis wygenerowany automatycznie

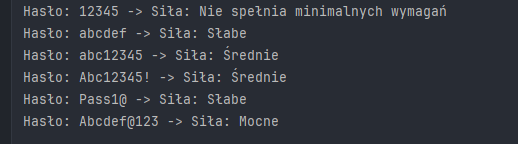
Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

*import* re  
*# Tekst i wyrażenie regularne*text = "Ala ma kota a kot ma Ale"  
pattern = "[a-z]{3}" *# Trzy małe litery z alfabetu  
# Dopasowanie bez flagi*no\_flag = re.match(pattern, text)  
*if* no\_flag:  
 print(f"Dopasowanie bez flagi: {no\_flag.group()}")  
*else*:  
 print("Brak dopasowania bez flagi.")  
*# Dopasowanie z flagą re.I (ignorowanie wielkości liter)*with\_flag = re.match(pattern, text, flags=re.I)  
*if* with\_flag:  
 print(f"Dopasowanie z flagą re.I: {with\_flag.group()}")  
*else*:  
 print("Brak dopasowania z flagą.")

Zad 6

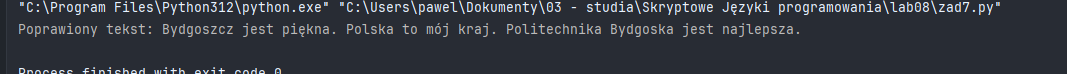




*import* re  
*def* check\_pasw(password):  
 *# Słabe hasło: co najmniej 6 znaków* weak = r"^.{6,}$"  
 *# Średnie hasło: co najmniej 8 znaków, litera i cyfra* medium = r"^(?=.\*[A-Za-z])(?=.\*\d).{8,}$"  
 *# Mocne hasło: co najmniej 10 znaków, małe i wielkie litery, cyfra, znak specjalny* strong = r"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@#$%^&+=!]).{10,}$"  
 *if* re.match(strong, password):  
 *return* "Mocne"  
 *elif* re.match(medium, password):  
 *return* "Średnie"  
 *elif* re.match(weak, password ):  
 *return* "Słabe"  
 *else*:  
 *return* "Nie spełnia minimalnych wymagań"  
*# Przykładowe testy*paswords = [  
 "12345", *# Zbyt krótkie* "abcdef", *# Słabe* "abc12345", *# Średnie* "Abc12345!", *# Mocne* "Pass1@", *# Średnie* "Abcdef@123", *# Mocne*]  
*for* password *in* paswords:  
 print(f"Hasło: {password} -> Siła: {check\_pasw(password)}")

Zad 7





*import* re  
*def* popraw\_bledy\_w\_tekście(tekst):  
 *# Wzorzec do znalezienia błędnych słów* pattern = r'\b([A-Z][A-Z][a-zA-Z]\*)\b'  
 *def* popraw\_wyraz(match):  
 word = match.group(1)  
 *# Zmieniam pierwszą literę na wielką, resztę na małe  
 return* word[0] + word[1:].lower()  
  
 *# Zamiana błędnych słów na poprawione* up\_text = re.sub(pattern, popraw\_wyraz, tekst)  
 *return* up\_text  
*# Przykładowy tekst*text = "BYdgoszcz jest piękna. POlska to mój kraj. POlitechnika BYdgoska jest najlepsza."  
*# Popraw błędy w tekście*up\_text = popraw\_bledy\_w\_tekście(text)  
print("Poprawiony tekst:", up\_text)

Wnioski:

Wyrażenia regularne są bardzo przydatnym narzędziem do automatyzacji pracy z tekstem, szczególnie w zadaniach związanych z wyszukiwaniem i poprawianiem błędów. Dzięki odpowiednio zdefiniowanym wzorcom można łatwo identyfikować niepoprawne fragmenty tekstu i wprowadzać poprawki w sposób szybki i efektywny. Zadania te pokazują, że warto zrozumieć podstawy regexów, bo znajdują one zastosowanie w wielu praktycznych problemach.