
	Politechnika Bydgoska im. J. J. Śniadeckich  <b>Wydział Telekomunikacji, Informatyki i Elektrotechniki</b>		
<b>Przedmiot</b>	Skryptowe języki programowania		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	<i>Regex - drugie starcie</i>		
<b>Student</b>	Marcin Ogórkiewicz		
<b>Nr ćw.</b>	8	<b>Data wykonania</b>	22.11.2023
<b>Ocena</b>		<b>Data oddania spr.</b>	22.11.2023

## Zadanie 1

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310
# importy
import typing
import re
# stałe i zmienne globalne

# funkcje

def main() -> None:
    import re
    text = 'aaaa'
    print("Kwantyfikator '+':", re.findall('a+', text))
    print("Kwantyfikator '+?':", re.findall('a+?', text))
    print("Kwantyfikator '*':", re.findall('a*', text))
    print("Kwantyfikator '*?':", re.findall('a*?', text))
    print("Kwantyfikator '?':", re.findall('a?', text))
    print("Kwantyfikator '??':", re.findall('a??', text))

main()
"""Kwantyfikator '+': ['aaaa'] - Dopasowuje jedno lub więcej wystąpień 'a'.

Kwantyfikator '+?': ['a', 'a', 'a', 'a'] - Leniwy odpowiednik '+',
dopasowuje jak najmniejszą ilość 'a'.

Kwantyfikator '*': ['aaaa', ''] - Dopasowuje zero lub więcej wystąpień 'a'.

Kwantyfikator '*?': ['', 'a', 'a', 'a', 'a', ''] - Leniwy odpowiednik '*',
dopasowuje jak najmniejszą ilość 'a'.
```

Kwantyfikator '?': ['a', 'a', 'a', 'a', ''] - Dopasowuje zero lub jedno wystąpienie 'a'.

Kwantyfikator '??': ['', 'a', '', 'a', '', 'a', '', 'a', ''] - Leniwy odpowiednik '?', dopasowuje jak najmniejszą ilość 'a'."""

```
Kwantyfikator '+': ['aaaa']
Kwantyfikator '+?': ['a', 'a', 'a', 'a']
Kwantyfikator '*': ['aaaa', '']
Kwantyfikator '*?': ['', 'a', '', 'a', '', 'a', '', 'a', '']
Kwantyfikator '?': ['a', 'a', 'a', 'a', '']
Kwantyfikator '??': ['', 'a', '', 'a', '', 'a', '', 'a', '']

Process finished with exit code 0
```

## Zadanie 2

```
#!C:\Users\koku\AppData\Local\Programs\Python\Python310

# importy
import typing
import re
# stałe i zmienne globalne

# funkcje

def main() -> None:
    with open('../Lab07/inwokacja.txt', 'r', encoding='utf-8') as file:
        text = file.read()
    words_before_exclamation = re.findall(r'\b\w+\b(?:\s*!)', text)
    print("Słowa po których występuje '!':", words_before_exclamation)
    words_with_polish_chars = re.findall(r'\b\w*[ąćęłńóśźż]\w*\b', text)
    print("Słowa z polskimi znakami:", words_with_polish_chars)
    count_ci = len(re.findall(r'\bcię\b|\bci\b', text,
flags=re.IGNORECASE))
    print("Liczba wystąpień 'cię' lub 'ci':", count_ci)

main()
```

```
Słowa po których występuje '!': ['Litwo', 'moja', 'Bramie', 'ludem']
Słowa z polskimi znakami: ['jesteś', 'cię', 'cenić', 'się', 'cię', 'stracił', 'Dziś', 'piękność', 'twą', 'całej', 'Widzę', 'opisuję', 'tęsknię', 'Święta', 'Częstochowy', 'świecis']
Liczba wystąpień 'cię' lub 'ci': 2

Process finished with exit code 0
```

## Zadanie 3

```

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing
import re
# stałe i zmienne globalne

# funkcje

def main() -> None:
    with open('adresy.txt', 'r', encoding='utf-8') as file:
        lines = file.readlines()
        address_pattern = re.compile(
            r"ul\. (?P<street_name>[\w\s]+)
            (?:(?P<house_number>\d+) (?:/ (?P<apartment_number>\d+))?)? (\d{2}-\d{3})")
        for line in lines:
            match = address_pattern.search(line)
            if match:
                street_name = match.group('street_name').strip()
                apartment_number = match.group('apartment_number').strip() if
match.group('apartment_number') else "None"
                postal_code = match.group(4).strip()
                print(
                    f"Nazwa ulicy: {street_name}, Numer mieszkania:
{apartment_number}, Kod pocztowy: {postal_code}")

main()

```

```

Nazwa ulicy: Zielona, Numer mieszkania: None, Kod pocztowy: 86-223
Nazwa ulicy: Jasna, Numer mieszkania: 5, Kod pocztowy: 85-444
Nazwa ulicy: Jasna, Numer mieszkania: 44, Kod pocztowy: 85-444
Nazwa ulicy: Biała, Numer mieszkania: None, Kod pocztowy: 86-656

Process finished with exit code 0

```

#### Zadanie 4

```

r"(?P<street_name>\D+)\s"

```

#### Zadanie 5

```

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

```

```

import re
# stałe i zmienne globalne

# funkcje

def main() -> None:
    print(re.match("[a-z]{3}", "Ala ma kota a kot ma Ale"))
    print(re.match("[a-z]{3}", "Ala ma kota a kot ma Ale", flags=re.I))

main()
"""Flaga re.I albo inaczej re.IGNORECASE pozwala na ignorowanie wielkości
liter w dopasowaniach."""

```

```

C:\Users\koku\PycharmProjects\SkrptoweJęzykiProgramowania\venv\
None
<re.Match object; span=(0, 3), match='Ala'>

Process finished with exit code 0

```

## Zadanie 6

```

#!C:\Users\koku\AppData\Local\Programs\Python\Python310

# importy
import typing
import re
# stałe i zmienne globalne

# funkcje

def validate_password(password):
    # Słabe hasło: Minimum 8 znaków
    weak_pattern = re.compile(r'^.{8,}$')
    # Średnie hasło: Minimum 8 znaków, przynajmniej jedna duża litera,
    jedna mała litera i jedna cyfra
    medium_pattern = re.compile(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$')
    # Mocne hasło: Minimum 10 znaków, przynajmniej jedna duża litera, jedna
    mała litera, jedna cyfra i jeden znak specjalny
    strong_pattern = re.compile(r'^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*\W).{10,}$')
    if strong_pattern.match(password):
        return "Password is strong"
    elif medium_pattern.match(password):
        return "Password is medium"
    elif weak_pattern.match(password):
        return "Password is weak"
    else:
        return "Password does not meet the criteria for weak, medium, or

```

```

strong"

def main() -> None:
    password1 = "weakPass"
    password2 = "MediumPass1"
    password3 = "Strong@Pass123"
    print(validate_password(password1))
    print(validate_password(password2))
    print(validate_password(password3))

main()

```

```

C:\Users\koksu\PycharmProjects\Skryp
Password is weak
Password is medium
Password is strong

Process finished with exit code 0

```

## Zadanie 7

```

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing
import re
# stałe i zmienne globalne

# funkcje

def fix_capitalization(match):
    word = match.group(0)
    if len(word) > 2:
        return word.capitalize()
    else:
        user_input = input(f"Czy chcesz naprawić słowo: '{word}'? (t/n): ")
        if user_input.lower().strip() == 't':
            return word.capitalize()
        else:
            return word

def correct_capitalization(text):
    pattern = re.compile(r'\b([A-Z]{2}[a-z]*|[a-z]+[A-Z]+[a-z]*|[A-Z][a-z]*[A-Z]+[a-z]*)\b')
    corrected_text = pattern.sub(fix_capitalization, text)
    return corrected_text

```

```
def main() -> None:
    text = \
        "BYdgoszcz to piękne miasto. POlska ma niezwykłą historię.
PoliTEchnika Bydgoska jest super. IT jest ciekawe."
    corrected_text = correct_capitalization(text)
    print("Pierwotny tekst:", text)
    print("Poprawiony tekst:", corrected_text)

main()
```

```
Czy chcesz naprawić słowo: 'IT'? (t/n): n
Pierwotny tekst: BYdgoszcz to piękne miasto. POlska ma niezwykłą historię. PoliTEchnika Bydgoska jest super. IT jest ciekawe.
Poprawiony tekst: Bydgoszcz to piękne miasto. Polska ma niezwykłą historię. Politechnika Bydgoska jest super. IT jest ciekawe.

Process finished with exit code 0
```

## Wnioski

Ćwiczenie pomogło mi bardziej zaznajomić się z działaniami na ciągach znaków przy pomocy funkcji wbudowanych oraz biblioteki re. Przepraszam za opóźnienie w przesłaniu. Sprawozdanie jest już gotowe od tygodnia, tylko zapomniałem go wysłać.