Programowanie Python 1

(CP1S02005)

Politechnika Białostocka - Wydział Elektryczny

Cyfryzacja przemysłu, sem. II, studia stacjonarne I stopnia

Rok akademicki 2023/2024

Wykład nr 4 (27.03.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 4

- Ciągi tekstowe
  - implementacja, sposób zapisu
  - odwołania do elementów
  - wybrane metody
  - powiązanie z listami
  - porównywanie ciągów tekstowych
  - zastosowanie operatorów + i *

Python - ciągi tekstowe

- Ciąg tekstowy to seria znaków, które służą do przechowywania informacji (danych) tekstowych
- Ciąg tekstowy może być ujęty w cudzysłów lub apostrofy

```
napis1 = "Witaj świecie!"
napis2 = 'Witaj swiecie!'

napis3 = """Witaj świecie!"""
napis4 = '''Witaj świecie!'''
```

  - w przypadku ograniczenia trzema znakami, ciąg tekstowy może znajdować się w więcej niż jednym wierszu kodu programu

- Ciągi tekstowe są obiektami klasy str
  - https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Python - ciągi tekstowe

- nie można jednocześnie stosować cudzysłowu i apostrofu do ograniczenia tekstu

```
napis1 = "Witaj świecie!'
```

```
File "d:\MyApp.py", line 1
    napis1 = "Witaj świecie!'
             ^
SyntaxError: unterminated string literal (detected at line 1)
```

```
napis1 = 'Witaj świecie!"
```

```
File "d:\MyApp.py", line 1
    napis1 = 'Witaj świecie!"
             ^
SyntaxError: unterminated string literal (detected at line 1)
```

Python - ciągi tekstowe

- można stosować inne znaki (cudzysłów / apostrof) jako cytaty

```
kod1 = "Przedmiot 'Programowanie C' (CP1S01005)"
kod2 = 'Przedmiot "Programowanie Python 1" (CP1S02005)'
```

- umieszczenie znaku cudzysłowu (") w tekście ograniczonym tymi samymi znakami wymaga dodania znaku \ (ang. backslash)

```
kod1 = "Przedmiot \"Programowanie C\" (CP1S01005)"
```

- umieszczenie znaku apostrofu (') w tekście ograniczonym tymi samymi znakami wymaga dodania znaku \ (ang. backslash)

```
kod2 = 'Przedmiot \'Programowanie Python 1\' (CP1S02005)'
```

Python - ciągi tekstowe

- przykład występowania znaku apostrofu w tekście

```
tekst1 = "Prawo Ampere'a"
tekst2 = "Kupiłem nowego iPhone'a"

print(tekst1)
print(tekst2)
```

```
Prawo Ampere'a
Kupiłem nowego iPhone'a
```

- uwaga: apostrof stawiamy wtedy, gdy ostatnia litera wyrazu obcego nie jest wymawiana w języku polskim

Python - ciągi tekstowe (długość ciągu)

- funkcja len() - zwraca długość ciągu tekstowego (liczbę znaków)

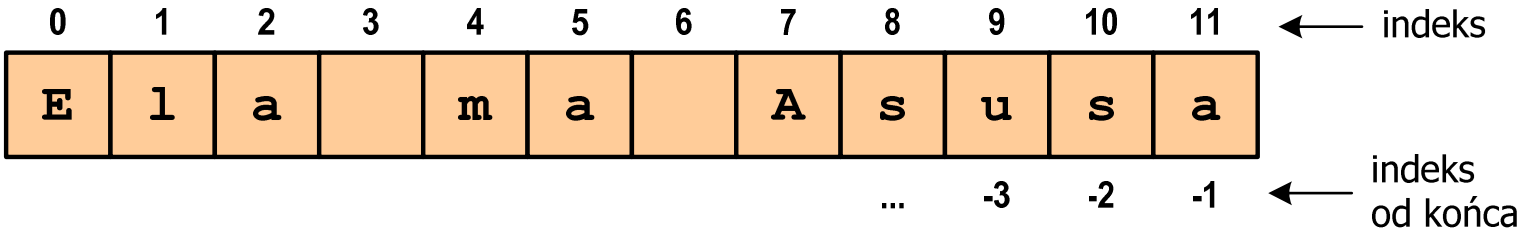
```
tekst = input("Podaj tekst: ")
dlugosc = len(tekst)
print(f"Liczba wprowadzonych znaków: {dlugosc}")
```

```
Podaj tekst: Programowanie Python 1
Liczba wprowadzonych znaków: 22
```

Python - ciągi tekstowe (odwołania do elementów)

- ciąg tekstowy jest typem sekwencyjnym - możliwy jest zatem dostęp do dowolnego elementu poprzez podanie jego indeksu

```
tekst = "Ela ma Asusa"
```



- odwołanie ma postać: nazwa_ciągu[indeks]

```
tekst = "Ela ma Asusa"

print(f"Pierwszy znak od początku: {tekst[0]}")
print(f"Drugi znak od początku:    {tekst[1]}")
print(f"Drugi znak od końca:       {tekst[-2]}")
```

```
E
l
s
```

Python - ciągi tekstowe (odwołania do elementów)

- zastosowanie dwukropka (:) do tworzenia indeksów elementów

```
lista[indeks_początkowy : indeks_końcowy : krok]
```

- każdy z indeksów może być pominięty, wtedy przyjmowane są wartości domyślne:
  - indeks_początkowy: 0 (indeks pierwszego elementu)
  - indeks_końcowy: len(lista) (indeks ostatniego elementu)
  - krok: 1

```
tekst = "Ela ma Asusa"
print(f"Trzy pierwsze znaki: {tekst[0:3]}")    # 0,1,2
```

```
Trzy pierwsze znaki: Ela
```

Python - ciągi tekstowe (odwołania do elementów)

```
tekst = "Ela ma Asusa"
print(f"Trzy pierwsze znaki: {tekst[:3]}")
print(f"Tekst bez trzech pierwszych znaków: {tekst[3:]}")
```

```
Trzy pierwsze znaki: Ela
Tekst bez trzech pierwszych znaków:  ma Asusa
```

```
tekst = "Ela ma Asusa"
print(f"Ostatni znak: {tekst[-1]}")
print(f"Dwa ostatnie znaki: {tekst[-2:]}")
print(f"Bez dwóch ostatnich znaków: {tekst[0:-2]}")
```

```
Ostatni znak: a
Dwa ostatnie znaki: sa
Bez dwóch ostatnich znaków: Ela ma Asu
```

Python - ciągi tekstowe (odwołania do elementów)

```
tekst = "Ela ma Asusa"
print(f"Co drugi element od pierwszego: {tekst[::2]}")
print(f"Co drugi element od drugiego: {tekst[1::2]}")
```

```
Co drugi element od pierwszego: Eam ss
Co drugi element od drugiego: l aAua
```

```
tekst = "Ela ma Asusa"
print(f"Od końca: {tekst[::-1]}")
```

```
Od końca: asusA am alE
```

Python - ciągi tekstowe (wybrane metody)

- metoda title() - zamienia pierwsze litery każdego wyrazu na wielkie (nie modyfikuje oryginalnego tekstu)

```
napis = "ela ma asusa"
print(napis.title())
print(napis)
```

```
Ela Ma Asusa
ela ma asusa
```

- trwała zmiana wielkości liter w tekście

```
napis = "ela ma asusa"
napis = napis.title()
print(napis)
```

```
Ela Ma Asusa
```

Python - ciągi tekstowe (wybrane metody)

- metoda upper() - zamienia małe litery na wielkie

```
napis = "Ela ma Asusa"
print(napis.upper())
```

```
ELA MA ASUSA
```

- metoda lower() - zamienia wielkie litery na małe

```
napis = "Ela ma Asusa"
print(napis.lower())
```

```
ela ma asusa
```

Python - ciągi tekstowe (wybrane metody)

- metoda removeprefix() - usuwa prefiks z łańcucha znaków (jeśli istnieje)

```
url = "https://we.pb.edu.pl"
url = url.removeprefix("https://")
print(url)
```

```
we.pb.edu.pl
```

- metoda removesuffix() - usuwa sufiks z łańcucha znaków (jeśli istnieje)

```
fname = "ocena_python.txt"
print(f"Nazwa pliku: {fname.removesuffix(".txt")}")
```

```
Nazwa pliku: ocena_python
```

Python - ciągi tekstowe (wybrane metody)

- metoda lstrip() - usuwa białe znaki z lewej strony ciągu tekstowego (na początku tekstu)
- metoda rstrip() - usuwa białe znaki z prawej strony ciągu tekstowego (na końcu tekstu)
- metoda strip() - usuwa białe znaki z lewej i z prawej strony ciągu tekstowego (na początku i na końcu tekstu)
- białe znaki: spacje (" "), tabulatory ("\t"), znaki nowej linii ("\n")

```
tekst = "   Jan Kowalski  "

print(f"[{tekst}]")
print(f"[{tekst.lstrip()}]")
print(f"[{tekst.rstrip()}]")
print(f"[{tekst.strip()}]")
```

```
[   Jan Kowalski  ]
[Jan Kowalski  ]
[   Jan Kowalski]
[Jan Kowalski]
```

Python - ciągi tekstowe (wybrane metody)

- metody lstrip(), lstrip() i strip() mogą mieć argument w postaci zestawu innych znaków do usunięcia

```
tekst = "###Jan Kowalski###"

print(f"[{tekst}]")
print(f"[{tekst.lstrip("#")}]")
print(f"[{tekst.rstrip("#")}]")
print(f"[{tekst.strip("#")}]")
```

```
[###Jan Kowalski###]
[Jan Kowalski###]
[###Jan Kowalski]
[Jan Kowalski]
```

Python - ciągi tekstowe (wybrane metody)

- metoda startswith(prefix) - zwraca wartość True jeśli dany ciąg tekstowy zaczyna się od podanego prefiksu

```
tekst = "Witaj świecie"
if tekst.startswith("Witaj"):
    print("Ciąg zaczyna się od: 'Witaj'")
else:
    print("Ciąg nie zaczyna się od: 'Witaj'")
```

- metoda endswith(suffix) - zwraca wartość True jeśli dany ciąg tekstowy kończy się podanym sufiksem

```
tekst = "Witaj świecie"
if tekst.endswith("świecie"):
    print("Ciąg kończy się na: 'świecie'")
else:
    print("Ciąg nie kończy się na: 'świecie'")
```

Python - ciągi tekstowe (wybrane metody)

- metoda count(substring) - zwraca liczbę wystąpień określonego podciągu w danym ciągu

```
tekst = "Ela ma laptopa"
ile_a = tekst.count("a")
print(f"Liczba wystąpień litery 'a': {ile_a}")
```

```
Liczba wystąpień litery 'a': 4
```

Python - ciągi tekstowe (wybrane metody)

- metoda find(substring) - wyszukuje określony podciąg w danym ciągu
- zwraca indeks pierwszego wystąpienia tego podciągu lub -1 jeśli podciąg nie został znaleziony

```
tekst = "Ela ma laptopa"
indeks = tekst.find("laptop")
if indeks == -1:
    print("Brak podciągu")
else:
    print(f"Początek podciągu to elementu nr: {indeks}")
```

```
Początek podciągu to elementu nr: 7
```

- metoda rfind(substring) - zwraca indeks ostatniego wystąpienia podciągu w ciągu lub zwraca -1 jeśli podciąg nie został znaleziony

Python - ciągi tekstowe i listy

- ciągi tekstowe mogą być elementami list

```
bierne = ["rezystor", "cewka", "kondensator"]
laptop = ['IBM', 'Asus', 'Lenovo']
```

- do każdego elementu listy będącego ciągiem tekstowym można stosować opisane wcześniej metody

```
laptop = ['IBM', 'Asus', 'Lenovo']
print(laptop[0].lower())
print(laptop[1].upper())
print(laptop[0].title())
```

```
ibm
ASUS
Ibm
```

Python - ciągi tekstowe i listy

- metoda sort() - sortuje elementy na liście w kolejności alfabetycznej (elementy są sortowane trwale)

```
bierne = ["rezystor", "cewka", "kondensator"]
bierne.sort()
print(bierne)
```

```
['cewka', 'kondensator', 'rezystor']
```

- elementy można posortować w odwrotnej kolejności alfabetycznej

```
bierne = ["rezystor", "cewka", "kondensator"]
bierne.sort(reverse = True)
print(bierne)
```

```
['rezystor', 'kondensator', 'cewka']
```

Python - ciągi tekstowe i listy

- funkcja sorted() - sortuje elementy na liście w kolejności alfabetycznej (bez trwałego zapisywania)

```
bierne = ["rezystor", "cewka", "kondensator"]
print(bierne)
print(sorted(bierne))
print(bierne)
```

```
['rezystor', 'cewka', 'kondensator']
['cewka', 'kondensator', 'rezystor']
['rezystor', 'cewka', 'kondensator']
```

- funkcja sorted() może także mieć argument reverse = True (sortowanie w odwrotnej kolejności alfabetycznej)

Python - ciągi tekstowe i listy

- do wyświetlenia elementów listy można zastosować pętlę for i jedną z przedstawionych wcześniej metod wykonujących operacje na ciągach tekstowych

```
bierne = ["rezystor", "cewka", "kondensator"]

for element in bierne:
    print(f"Element bierny: {element.title()}")
```

```
Element bierny: Rezystor
Element bierny: Cewka
Element bierny: Kondensator
```

Python - ciągi tekstowe i listy

- metoda split(separator, maxsplit) - dzieli ciąg tekstowy na fragmenty, zwane tokenami, na podstawie określonego separatora
- separator (opcjonalny) - znak lub ciąg znaków, który służy jako separator podziału tekstu, domyślnie separator to białe znaki (spacje, tabulatory, znaki nowej linii)
- maxsplit (opcjonalny) - maksymalna liczba podziałów, które zostaną wykonane

```
tekst = input("Podaj tekst: ")
lista = tekst.split()
print(lista)
```

```
Podaj tekst: Ela ma laptopa
['Ela', 'ma', 'laptopa']
```

Python - porównywanie ciągów tekstowych

- do ciągów tekstowych można zastosować operatory porównania

Operator	Znaczenie	Operator	Znaczenie
==	równa się	!=	nie równa się
>	większe niż	>=	mniejsze lub równe
<	mniejsze niż	<=	większe lub równe

- w wyniku porównania otrzymujemy wartość True lub False

```
pass = input("Podaj hasło: ")
if pass == "123456":
    print("To jest najpopularniejsza hasło na świecie")
else:
    print("Hasło jest OK")
```

- dwa ciągi tekstowe są sobie równe, gdy składają się z takich samych znaków umieszczonych na identycznych pozycjach

Python - porównywanie ciągów tekstowych

- przy porównywaniu tekstów należy zwrócić uwagę na wielkość liter

```
imie = input("Jak na imię miał Kopernik? ")
if imie.lower() == "mikołaj":
    print("Poprawna odpowiedź!")
else:
    print("Błędna odpowiedź!")
```

```
Jak na imię miał Kopernik? Mikołaj
Poprawna odpowiedź!
```

```
Jak na imię miał Kopernik? mikołaj
Poprawna odpowiedź!
```

```
Jak na imię miał Kopernik? mikolaj
Błędna odpowiedź!
```

Python - ciągi tekstowe (przykład)

- określenie liczby cyfr w wierszu tekstu

```
tekst = (input("Podaj tekst: "))
ile = 0
for znak in tekst:
    if ord(znak) >= 48 and ord(znak) <= 57:
        ile = ile + 1
print(f"W tekście jest {ile} cyfr")
```

```
Podaj tekst: asd58Dr4Hik2189
W tekście jest 7 cyfr
```

- funkcja ord() - argumentem jest pojedynczy znak (typu str), zwraca odpowiadającą mu wartość liczbową Unicode tego znaku

Python - ciągi tekstowe (przykład)

- określenie liczby cyfr w wierszu tekstu

```
cyfry = "0123456789"
tekst = (input("Podaj tekst: "))
ile = 0
for znak in tekst:
    if znak in cyfry:
        ile = ile + 1
print(f"W tekście jest {ile} cyfr")
```

```
Podaj tekst: asd58Dr4Hik2189
W tekście jest 7 cyfr
```

- operator in w warunku if służy do sprawdzania czy dany element jest zawarty w określonej sekwencji (lista, krotka, ciąg znaków, inny iterowalny obiekt)

Python - ciągi tekstowe (przykład)

- sprawdzenie, czy element znajduje się na liście

```
lista = ["Nowak", "Kowalski", "Wiśniewski", "Kowalczyk"]
nazwisko = input("Podaj nazwisko: ")
if nazwisko in lista:
    print(f"{nazwisko} znajduje się na liście")
else:
    print(f"{nazwisko} nie znajduje się na liście")
```

```
Podaj nazwisko: Kowalski
Kowalski znajduje się na liście
```

```
Podaj nazwisko: Kowal
Kowal nie znajduje się na liście
```

Python - ciągi tekstowe i operatory: +, *

- operator + jest używany do łączenia dwóch ciągów tekstowych w jeden

```
imie = "Jan"
nazwisko = "Kowalski"
osoba = imie + " " + nazwisko
print(osoba)
```

```
Jan Kowalski
```

- operator * jest używany do powielania ciągu określoną liczbę razy

```
tekst = "<->"
nowy = tekst * 10
print(nowy)
```

```
<-><-><-><-><-><-><-><-><-><->
```

Koniec wykładu nr 4

Dziękuję za uwagę!