Python. Laboratorium 10. Operacje na napisach.

Napisy znane jako typ danych string, są rozumiane jako łańcuch znaków - liter, cyfr i symboli. Stringi definiuje się w podwójnych lub pojedyńczych cudzysłowach.
kawa = "latte"
herbata = 'zielona '
cena = "13.99"
Python dysponuje wbudowanymi możliwościami funkcjonalnymi służącymi do wykonywania operacji na łańcuchach znaków, takich jak dzielenie łańcuchów na części w miejscu wystąpienia określonego znaku lub zmiana wielkości liter w łańcuchu. Jeśli na przykład dysponujemy łańcuchem znaków zapisanym WIELKIMI LITERAMI, a chcemy, by wyświetlić go małymi literami, możemy to zrobić przy użyciu funkcji dostarczanych przez Python.

Są pewne znaki, które użyte w stringach sprawiają kłopoty.
sentence = 'There's a snake in my boot!'
W powyższym wypadku Python uważa, że po 'There' string doszedł do końca. Ten problem można rozwiązać na dwa sposoby. Albo użyć backslash przed pojedyńczym cudzysłowem wewnątrz stringa np.
sentence = 'There\'s a snake in my boot!'
albo użyć podwójnych cudzysłowów.
sentence = "There's a snake in my boot!"

Łańcuchy znaków, podobnie jak listy i krotki, są iterowalne. Oznacza to, że każdy ze znaków łańcucha można odczytać, używając jego indeksu. Podobnie jak inne dane iterowalne, pierwszy znak łańcucha zawsze ma indeks 0, a indeksy kolejnych znaków są większe o 1
Np.
autor = "Kafka"
autor[0]
autor[1]
autor[2]
autor[3]
autor[4]

Metody zwracające nowy napis (string):

1. lower() Zmienia wszystkie duże litery na małe w stringu
2. upper() Zmienia wszystkie małe litery na duże w stringu
3. swapcase() Odwraca rodzaj każdej litery – małe na duże, duże na małe
4. capitalize() Zmienia pierwszą literę w ciągu na dużą
5. title()
   Zwraca string – tytuł, w którym wszystkie wyrazy zaczynają się dużą literą, a reszta jest małymi lub są to znaki nieliterowe
6. join(seq)

Łączenie (konkatenacja) wyrazów w napisie seq w jeden napis, według separatora/stringu na jakim wywołujemy metodę

7. lstrip()
   Usuwa białe znaki z początku napisu – zwraca kopię pozbawioną białych znaków od lewej strony
8. rstrip()
   Usuwa białe znaki z końca napisu – zwraca kopię pozbawioną białych znaków od prawej strony
9. strip([chars]) Usuwa białe znaki lub znak podanny jako char z początku i końca napisu – wykonuje lstrip() i rstrip() na napisie.
10. max(string) Zwraca literę znajdującą się najdalej w alfabecie od A
11. min(string) Zwraca literę znajdującą się najbliżej w alfabecie od A
12. split(str="", num=string.count(str))
    Dzieli łańcuch według separatora str (spacja jeśli nie podano) i zwraca podciągi jako listę lub podzieli na co najwyżej liczbę podciągów, jeśli podano num
13. splitlines(num=string.count('n'))
    Dzieli cały łańcuch (lub wg zadanej liczby num) na osobne linie wg znaku nowej linii'\n' i zwraca je jako tablicę
14. replace(old, new [, max])
    Zamienia wszystkie wystąpienia ciagu old na ciąg new lub jeśli jest podane max – podmiana zostanie wykonana o wskazaną liczbę wystąpień

Przykłady:

```
string="koty to najukochansze zwierzęta na swiecie"
print(string.upper())
print(string.capitalize())
print(string.lower())


#Zmiana jakiegos napisu
string = 'I kocham koty'
newstring = string.replace('kocham', 'jem')
print(newstring)


#Czytanie od końca
string="Ala ma kota"
print(''.join(reversed(string)))

#Dzielenie stringów :
string="Ala ma kota"
print(string.split(' '))
```

Metody zwracające wartość liczbową:

Metody zwracające wartość liczbową:

1. len(string)
   Zwraca długość ciągu znaków
2. count(str, beg=0,end=len(string))
   Zlicza ile razy zadany ciąg znaków(str) wystąpił w ciągu znaków lub wewnątrz podciągu, który zaczyna się od indeksu beg i kończy indeksem end
3. find(str, beg=0 end=len(string))
   Sprawdza gdzie ciąg str występuje w napisie lub podciągu tego napisu jeśli podamy indexbeg i indeks końcowy end. Zwraca indeks początkowy lub -1 jeśli ciąg str nie znajduje się w napisie
4. rfind(str, beg=0,end=len(string))
   Działa jak find(), ale wyszukiwanie od końca ciągu znaków
5. index(str, beg=0, end=len(string))
   Działa jak find(), ale zwraca wyjątek jeśli ciąg str nie zostanie znaleziony
6. rindex(str, beg=0, end=len(string))
7. Działa jak index(), ale wyszukiwanie od końca ciągu znaków

Metody zwracające true/false:

1. isalnum()
   Zwraca true jeśli wszystkie znaki w ciągu są alfanumeryczne (litery lub cyfry)
2. isalpha()
   Zwraca true jeśli wszystkie znaki w ciągu są literami
3. isdigit()
   Zwraca true jeśli wszystkie znaki w ciągu są cyframi
4. islower()
   Zwraca true jeśli wszystkie znaki w ciągu są małymi literami.
5. isspace()
   Zwraca true jeśli wszystkie znaki w ciągu są białymi znakami (spacja, tabulacja, przejście do nowej linii itp)
6. istitle()
   Zwraca true jeśli ciąg spełnia warunek tytułu (każdy wyraz napisu musi zaczynać się dużą literą i składać wyłącznie z małych liter lub znaków nieliterowych)
7. isupper()
   Zwraca true jeśli wszystkie znaki w ciągu są dużymi literami.
8. startswith(str, beg=0,end=len(string))

Zwraca wynik sprawdzenia, czy napis jest rozpoczęty ciągiem str. Przy podaniu indeksu beg, sprawdzenie rozpoczyna się od tego znaku. Przy wystąpieniu argumentu end sprawdzenie zakończy się na tym znaku

9. endswith(str, beg=0, end=len(string))
   Zwraca wynik sprawdzenia, czy napis jest zakończony ciągiem str. Przy podaniu indeksu beg, sprawdzenie rozpoczyna się od tego znaku. Przy wystąpieniu argumentu end sprawdzenie zakończy się na tym znaku

Metoda in pozwala sprawdzić, czy jeden łańcuch znaków występuje w innym; metoda ta zwraca wartości logiczne True lub False:
"Kot" in "Kot w butach." zwróci True.

W Pythonie jest na to kilka sposób, jednak w najnowszych wersjach preferowane są tzw. f-stringi. Rozpoznasz je po literze f znajdującej się przed cudzysłowiem. Zmienne w f-stringach umieszczamy w nawiasch klamrowych tak jak widać to na powyższym przykładzie.
Opisane operacje możliwe są jedynie od wersji 3.6 Pythona. We wcześniejszych wersjach należy skorzystać z metody format.

```
 x = 15
text = f"wynik = {x}"
print(text)
wynik = 15
```

W nawiasach klamrowych umieszczać możemy również operacje:

```
f"wynik = {34 * 15}"
'wynik = 510'
```

Korzystając z f-stringów możemy również narzucać odpowiedni format liczb:

```
pi = 3.1415
f"{pi:.3f}"
'3.142'
```

Liczba zostanie odpowiednio zaokrąglona zgodnie z zasadami matematyki.

Lista zadań:

1. Na podstawie poniższego cytatu z „Wiedźmina”
   „Magia jest w opinii niektórych ucieleśnieniem Chaosu. Jest kluczem zdolnym otworzyć zakazane drzwi. Drzwi, za którymi czai się koszmar, zgroza i niewyobrażalna okropność, za którymi czyhają wrogie, destrukcyjne siły, moce czystego zła, mogące unicestwić nie tylko tego, kto drzwi te uchyli, ale i cały świat. A ponieważ nie brakuje takich, którzy przy owych drzwiach manipulują, kiedyś ktoś popełni błąd, a wówczas zagłada świata będzie przesądzona i nieuchronna. Magia jest zatem zemstą i orężem Chaosu. To, że po Koniunkcji Sfer ludzie nauczyli posługiwać się magią, jest przekleństwem i zgubą świata. Zgubą ludzkości. I tak jest. Ci, którzy uważają magię za Chaos, nie mylą się.”

Przetestuj metody : lower() , swapcase(), capitalize() , replace(), lstrip(), rstrip(), reversed(),count(),find(), isalnum(), startswith(),endswith(). Zastosuj odpowiednie formatowanie przy użyciu f-stringów.

2. Napisz program który zapyta użytkownika o imię, nazwisko, numer telefonu oraz buta, a następnie: sprawdź czy imię i nazwisko składają się tylko z liter, a nr telefonu i numer buta składa się wyłącznie z cyfr (wyświetl tę informację jako true/false).Sprawdź czy imię i nazwisko rozpoczyna się z dużej litery. Jeśli nie popraw ich.Sprawdź czy numer telefonu oraz buta zawiera dziwne znaki np. myślniki. Usuń ich. Zakładając, że twoi użytkownicy noszą polskie imiona, sprawdź czy użytkownik jest kobietą. Uważaj na przypadki np.Nel, Barnaba. Zastosuj odpowiednie formatowanie przy użyciu f-stringów.

3. Dysponując łańcuchem znaków: "Długo na szturm i szaniec poglądał w milczeniu. Na koniec rzekł: 'Stracona'.", zwróć wycinek obejmujący znaki od początku do pierwszej kropki. Następnie dysponując listą ["Zwinny", "lis", "przeskoczył", "nad", "leniwym", "psem", "."], przekształć ją w poprawne gramatycznie zdanie. Pomiędzy poszczególnymi słowami zdania mają być umieszczone odstępy, lecz nie ma być odstępu pomiędzy ostatnim słowem i kropką.
4. Podaj przykłady w pracy programisty, kiedy znajomość powyższych metod może się przydać. Czy widzisz sens ich istnienia?