Operacje na łańcuchach znaków

Wyszukiwanie / Sprawdzanie

Metoda **index**() przeszukuje sekwencję od początku, aby znaleźć pierwszy element o wartości określonej w argumencie.

```
# Przedstawienie metody index()
print("aAbByYzZaA".index("b"))   #2
print("aAbByYzZaA".index("Z"))   #7
```

Metoda **find**() jest podobna do metody **index**(), którą już znasz - szuka ona podłańcucha znaków i zwraca indeks pierwszego jego wystąpienia, ale:

- Jest bezpieczniejsza - nie generuje błędu dla argumentu zawierającego nieistniejący podłańcuch znaków (zwraca w takim przypadku -1)
- działa wyłącznie z łańcuchami znaków - nie próbuj stosować jej do żadnej innej sekwencji.

```
# Przedstawienie metody find()
print("Fajna tawerna".find("ta"))

# Przedstawienie metody rfind()
print("tau tau tau".rfind("ta"))
print("tau tau tau".rfind("ta", 9))
print("tau tau tau".rfind("ta", 3, 9))


# Przedstawienie metody endswith()
if "epsilon".endswith("on"):
    print("tak")
else:
    print("nie")


# Przedstawienie metody startswith()
print("omega".startswith("meg"))
print("omega".startswith("om"))
```

Bezparametrowa metoda o nazwie **isalnum**() sprawdza, czy ciąg zawiera tylko cyfry lub znaki alfabetyczne (litery), i zwraca True lub False zgodnie z wynikiem

```
# Przedstawienie metody isalnum()
print('lambda30'.isalnum())
print('lambda_30'.isalnum())
```

Metoda **isalpha**() jest bardziej wyspecjalizowana - interesuje się wyłącznie literami.

Z kolei metoda **isdigit**() analizuje tylko cyfry

```
# Przykład 1: Przedstawienie metody isalpha()
print("Moooo".isalpha())
print('Mu40'.isalpha())


# Przykład 2: Przedstawienie metody isdigit()
print('2018'.isdigit()) # True
print("Year2019".isdigit())#
```

Inne

```
# Przykład 1: Przedstawienie metody islower()
print("Moooo".islower())
print('moooo'.islower())

# Przykład 2: Przedstawienie metody isspace()
print(' \n '.isspace())
print(" ".isspace())

# Przykład 3: Przedstawienie metody isupper()
print("Moooo".isupper())
print('MOOOO'.isupper())
```

Rozkładanie / Składanie

Funkcja **list**() bierze swój argument (łańcuch) i tworzy nową listę zawierającą wszystkie znaki łańcucha, po jednym na element listy.

Uwaga: nie jest to dokładnie funkcja łańcucha znaków - list() jest w stanie stworzyć nową listę z wielu innych elementów (np. z krotek i słowników).

```
print(list("abcabc")) # wynik ['a', 'b', 'c', 'a', 'b', 'c']
```

join / split

Metoda **join**() jak sugeruje nazwa (join - ang. łączyć), metoda ta wykonuje połączenie - oczekuje jednego argumentu jako listy; należy upewnić się, że wszystkie elementy listy są łańcuchami - w innym przypadku metoda ta zgłosi wyjątek TypeError;

wszystkie elementy listy zostaną połączone w jeden łańcuch, ale łańcuch znaków, z którego wywołano metodę, zostanie użyty jako separator, umieszczony w łańcuchach; nowo utworzony łańcuch jest zwracany jako wynik.

```
# Przedstawienie metody join()
print(",".join(["omicron", "pi", "rho"]))
```

Metoda **split**() dzieli łańcuch i tworzy listę wszystkich wykrytych podłańcuchów. Metoda ta zakłada, że podłańcuchy znaków są rozdzielane spacjami - spacje nie biorą udziału w operacji i nie są kopiowane do listy wynikowej.Jeśli łańcuch znaków jest pusty, lista wynikowa również będzie pusta.

```
# Przedstawienie metody split()
print("phi       chi\npsi".split())
```

Metoda **count**() zlicza wszystkie wystąpienia elementu wewnątrz sekwencji. Brak takich elementów nie powoduje żadnych problemów.

```
print("abcabc".count("b"))  #2
print('abcabc'.count("d"))  #0
```

MODYFIKACJE

Metoda **capitalize**() wykonuje dokładnie to, na co wskazuje jej nazwa - tworzy nowy łańcuch wypełniony znakami zaczerpniętymi z łańcucha źródłowego, ale próbuje je zmodyfikować w następujący sposób:

jeśli pierwszy znak w łańcuchu jest literą (uwaga: pierwszy znak jest elementem o indeksie równym 0, nie tylko pierwszym widocznym znakiem), zostanie on przekonwertowany na wielkie litery,

wszystkie pozostałe litery z łańcucha zostaną przekonwertowane na małe litery.

Przedstawienie metody **capitalize**() i **title**()

```
print("wiem, ze nic nie wiem. Czesc 1.".title())
print("wiem, ze nic nie wiem. Czesc 1.".capitalize())
```

strip() usuwa spacje lub to co jest w parametrze

Przedstawienie metody lstrip()

```
print("[" + " tau ".lstrip() + "]")
print("cisco.com".lstrip("cisco"))  # .com
```

Przedstawienie metody rstrip()

```
print("[" + " upsilon ".rstrip() + "]")
print("cisco.com".rstrip(".com"))
```

Przedstawienie metody strip() # usuwa widzące i kończące spacje

```
print("[" + "   aleph   ".strip() + "]")
```

Przedstawienie metody lower()

```
print("SiGmA=60".lower())


# Przedstawienie metody upper()
print("Wiem, ze nic nie wiem. Czesc 2.".upper())

# Przedstawienie metody swapcase()
print("Wiem, ze nic nie wiem.".swapcase())
```

Metoda dwuparametrowa **replace**() zwraca kopię oryginalnego łańcucha znaków, w której wszystkie wystąpienia pierwszego argumentu zostały zastąpione przez drugi argument.

```
# Przedstawienie metody replace()
print("To jest to!".replace("jest", "sa"))
Można dodać parametr  replace("jest", "sa", 1))
```