Lista w Pythonie to jedna z podstawowych struktur danych, która jest używana do przechowywania uporządkowanej sekwencji elementów. Listy są bardzo elastyczne i mogą przechowywać elementy różnych typów danych, takich jak liczby, ciągi znaków, a nawet inne listy.

Cechy list:

1. **Zmienność -** listy są modyfikowalne, co oznacza, że można zmieniać ich zawartość po utworzeniu, np. dodawać, usuwać, modyfikować elementy.
2. **Uporządkowanie** - elementy w liście mają ustaloną kolejność i są indeksowane. Indeksy zaczynają się od 0.
3. **Możliwość przechowywania różnych typów danych -** jednej liście można przechowywać elementy różnych typów, np. liczby, ciągi znaków, a nawet inne listy lub krotki.
4. **Zastosowanie nawiasów kwadratowych** - listy są definiowane przy użyciu nawiasów kwadratowych [].

Kiedy używać list:

- Listy są idealne, gdy chcesz pracować z sekwencją elementów, które **mogą zmieniać się w trakcie działania programu.**
- Gdy potrzebujesz operacji na sekwencji: Sortowanie, filtrowanie, dodawanie i usuwanie elementów – listy świetnie się do tego nadają.

Przykłady tworzenia listy:

lista1 = [1, 2, 3]

lista2 = ['jabłko', 'banan', 'gruszka']

Podstawowe operacje na listach:

1. **Dodawanie elementów**:
    - append(element) – dodaje element na końcu listy.
    - insert(index, element) – dodaje element na podanym indeksie.
    - extend(inna_lista) – rozszerza listę o elementy z innej listy.
2. **Usuwanie elementów**:
    - remove(element) – usuwa pierwszy napotkany element o podanej wartości.
    - pop(index) – usuwa i zwraca element z podanego indeksu (domyślnie ostatni).
    - clear() – usuwa wszystkie elementy z listy.
3. **Indeksowanie i wycinanie**:
    - lista[index] – zwraca element o podanym indeksie.
    - lista[start:stop:step] – zwraca podlistę od start do stop co step element.

4. **Sortowanie i odwracanie**:

   - sort() – sortuje listę.
   - reverse() – odwraca kolejność elementów w liście.

Operacje na listach

Lista=[1, 2, 3, 4]	definiowanie 4ro elementowej listy
lista[0] lista[-1]	odwołanie do pierwszego/ostatniego elementu listy
len(lista)	zwraca długość listy (liczbę elementów)
max(lista) min(lista)	zwraca największa/najmniejszą wartość z list
sum(lista)	zwraca sumę elementów listy
sorted(lista)	Sortuje elementy listy (od najmniejszego lub alfabetycznie
lista.append(6)	dodanie kolejnego elementu na końcu listy
lista.insert(i,5)	wstawienie elementu w określonym indeksie i oraz odpowiednie przesunięcie kolejnych elementów
lista.pop()	zwrócenie ostatniego elementu z listy z jednoczesnym usunięciem go z niej
lista.reverse()	odwrócenie kolejności elementów w liście
lista = lista1 + lista2 listaM = lista1*5	łączenie dwóch list w jedną Pięciokrotne powtórzenie wartości listy1 w nowej liście
lista[:n] lista[m:] lista[m:n:k] lista[::-1]	Wycinek listy: - od początku do elementu n - od elementu o indeksie m do końca - dany zakres elementów m-n z krokiem k (jak w range) - odwrócenie listy

Napisy (string), to ciągi znaków zachowujące się bardzo podobnie do tablic, ale nie są modyfikowalne – każda modyfikacja zapisuje tak naprawdę nowy string.

Dodatkowe operacje na string'ach:

tekst.find(„fragment")	zwraca pozycję (indeks pierwszego znaku) odnalezionego w tekście, fragmentu -1 oznacza, że go nie znaleziono	tekst = "Ala ma kota" indeks = tekst.find("ma") print(indeks) # Wypisze 4, ponieważ "ma" zaczyna się na 4. pozycji

tekst.replace	(„frDoZast”,„NowyFr”) zastąpienie pierwszego fragmentu drugim w tekści	tekst = "Ala ma kota, a kot ma Alę."
		nowy_tekst = tekst.replace("Ala", "Ola")
		print(nowy_tekst)
		# Wypisze: "Ola ma kota, a kot ma Olę."
		
		# Przykład z ograniczeniem liczby zamian (max=1)
		nowy_tekst_ograniczony = tekst.replace("ma", "posiada", 1)
		print(nowy_tekst_ograniczony)
		# Wypisze: "Ala posiada kota, a kot ma Alę."

tekst.split(znak)	Podział tekstu względem określonego znaku	tekst = "jabłko,banan,gruszka,śliwka"
		# Dzielimy tekst na części według przecinka
		owoce = tekst.split(",")
		print(owoce)
		# Wypisze: ['jabłko', 'banan', 'gruszka', 'śliwka']

tekst.upper() tekst.lower()	zamiana wszystkich liter na duże lub małe	tekst1 = "qwerty" tekst2 = "QWERTY" tekst1 = tekst1.upper() tekst2 = tekst2.lower() print(tekst1) # Wypisze: QWERTY print(tekst2) # Wypisze: qwerty

Zad. 1

Utwórz listę: Moja_lista=[1, 17, 3, 5, 3, 4, 86, 90, 2, 21], przetestuj działanie wszystkich zaprezentowanych operacji na listach, a ich wynik wyświetl w konsoli.

Zad. 2

Stwórz listę z imionami 4 osób.

a. Posortuj ją alfabetycznie i wyświetl,

b. Dodaj na końcu dwie osoby i pobierz ostatnią z nich poleceniem pop().

c. Na pozycji 3 dodaj jeszcze jedną osobę.

d. Odwróć kolejność na liście i zduplikuj ją.

Zad. 3

Napisz program, który:

a. Wczyta (zmienną) imię oraz wyświetli tekst „Witaj" oraz wczytane imię.

b. Wczyta (zmienną) wiek oraz wyświetli tekst „Twój wiek to:" oraz wczytany wiek.

c. Wczyta (zmienne) imię i nazwisko i wyświetli inicjały.

d. Wczyta łańcuch i wyświetli go pięć razy.

e. Wczyta dwa łańcuchy, a w trzecim łańcuchu zapamięta połączone te dwa łańcuchy.

f. Wczyta dwa łańcuchy, a w trzecim łańcuchu zapamięta pierwszą połowę pierwszego i drugą połowę drugiego.

Zad. 4 dodatkowe

Napisz program, który wczyta od użytkownika zdanie. Wypisz z niego wszystkie litery w kolejności alfabetycznej, a następnie wypisze których litera alfabetu nie zawiera.

Krotka (ang. *tuple*) w Pythonie to struktura danych, która jest uporządkowaną sekwencją elementów. Krotki są podobne do list, ale mają kilka kluczowych różnic:

Cechy krotek:

1. **Niezmienność -** po utworzeniu krotki nie można zmieniać, dodawać ani usuwać jej elementów. Oznacza to, że krotka jest niezmienialna, co odróżnia ją od listy, która jest zmienialna.
2. **Uporządkowanie -** elementy krotki mają swoje miejsce, czyli każdemu elementowi przypisany jest indeks (numer porządkowy), który zaczyna się od 0.
3. **Mogą zawierać różne typy danych** - w jednej krotce można przechowywać elementy różnych typów, np. liczby, ciągi znaków, inne krotki itp.
4. **Zastosowanie nawiasów okrągłych -** krotki są definiowane przy użyciu nawiasów okrągłych (), podczas gdy listy są definiowane przy użyciu nawiasów kwadratowych [].

Składnia: Tuple definiuje się, umieszczając elementy w nawiasach okrągłych ().

Przykład tworzenia krotki:

moja_krotka = (1, 2, 3, "Adam", True)

Praca z krotką:

Tuple są przydatne w wielu sytuacjach, zwłaszcza gdy chcesz przechowywać zestawy danych, które nie powinny być zmieniane, jak na przykład współrzędne geograficzne, dane konfiguracyjne aplikacji lub informacje o punktach w czasie.

Indeksowanie: Elementy tuple są uporządkowane i można odwoływać się do nich za pomocą indeksów. Indeksowanie rozpoczyna się od 0, więc pierwszy element ma indeks 0, drugi element ma indeks 1, itd.

pierwszy_element = moja_krotka [0] # Pobranie pierwszego elementu

Rozpakowywanie tuple: Możesz przypisywać elementy tuple do zmiennych w jednej linii, co nazywa się rozpakowywaniem tuple.

a, b, c, d, e = moja_krotka

Jeśli krotka ma inną liczbę elementów niż liczba zmiennych, możesz użyć symbolu `*`, aby przypisać resztę elementów do listy:

*a, b, *c = moja_krotka*

Przykład:

 # Przykładowa krotka z 5 elementami

moja_krotka = (1, 2, 3, 4, 5)

*# Przypisanie z użyciem **

*a, b, *c = moja_krotka*

print("a:", a) # Wynik: 1

print("b:", b) # Wynik: 2

print("c:", c) # Wynik: [3, 4, 5]

Długość tuple: Możesz sprawdzić, ile elementów zawiera tuple, korzystając z funkcji len().

długość = len(moja_krotka)

Iterowanie po tuple: Możesz użyć pętli for do iterowania po elementach tuple.

for element in my_tuple:

*   print(element)*

Wyszukiwania, zliczanie

auta = ('Audi', 'BMW', 'Opel', 'Renault', 'Maluch')

print(auta.index("Opel")) # Znajduje indeks pierwszego elementu równego "Opel"

print("Opel" in auta) # Sprawdza wystąpienie stringa w krotce

print(auta.count("Opel")) # Zliczenia występowania elementu w krotce, możesz użyć funkcji

Operacje na krotkach

krotka = (1, 2, 3, 4) krotka = 1, 2, 3, 4	definiowanie 4ro elementowej krotki bez nawiasów- niezalecane ale możliwe
krotka = krotka1 + krotka2 krotkaM = krotka1 * 5	łączenie dwóch krotek pięciokrotne powtórzenie krotki

Słowniki (dict)

Zawiera zestaw par klucz – wartość, przy czym klucze zastępujące indeksy muszą być unikalne

Operacje na słownikach

slownik = {k1 : w1, k2:w2}	definiowanie 2 elementowego słownika
slownik[k1]	odwołanie do elementu słownika o kluczu k1
slownik.keys()	zwraca wszystkie klucze
slownik.values()	zwraca wszystkie wartości
slownik.items()	zwraca gotowe pary krotek klucz-wartość
slownik.update({k3:w3}) slownik[k4] = w4	dwa sposoby na dodanie elementu do słownika
del slownik[k4]	Usunięcie elementu słownika o wskazanym kluczu

Zbiory (set)

Zawiera nieuporządkowane elementy bez powtórzeń. Jest lepiej zoptymalizowany niż lista i zapewnia unikalne wartości.

Operacje na zbiorach

zbior = {1, 2, 3, 4}	definiowanie 4ro elementowego zbioru
zbior.add(5)	dodaje element o zbioru
zbior.remove(4)	usuwa dany element ze zbioru
zbior.pop()	zwraca losowy element ze zbioru (brak kolejności)
4 in zbior	Zwraca wartość logiczną prawda/fałsz jeśli dany element znajduje się w zbiorze

Zad 5.

Utwórz krotkę zwierająca dni tygodnia.

Zad 6.

Podana jest poniższa krotka owoce:

owoce = ('jabłko', 'banan', 'gruszka', 'banan', 'banan', 'malina")

Stwórz program, który wykorzystuje odpowiednią metodę do zliczenia wystąpień elementu 'banan' w krotce.

Zad 7.

Posortu krotke moja_krotka= (10, 2, 6, 6, 9, 13, 0,1), operacji tej dokonaj posługując się pomocniczo listą

Zad 8.

Załóżmy, że pracujesz nad systemem wojskowym dotyczącym planowania operacji wojskowych.

Podana jest poniższa krotka:

```
stopnie = (
    "Szeregowy",
    "Kapral",
    "Sierżancie",
    "Porucznik",
    "Kapitan",
    "Major",
    "Pułkownik",
)
```

Wykonaj poniższe kroki:

- wyznacz liczbę wszystkich stopni wojskowych i przypisz do zmiennej ilość_stopnii,

- znajdź indeks krotki dla elementu "Major" i przypisz do zmiennej Major_index,

- sprawdź, czy wartość "Pułkownik" znajduje się w krotce stopnie i przypisz do zmiennej Pułkownik_wstepowanie.

W wydrukuj otrzymane zmienne do konsoli w podanej kolejności.

Zad 9.

Stwórz listę zakupów jako słownik (artykuł : kwota). Wyświetl go i podsumuj wydatki.