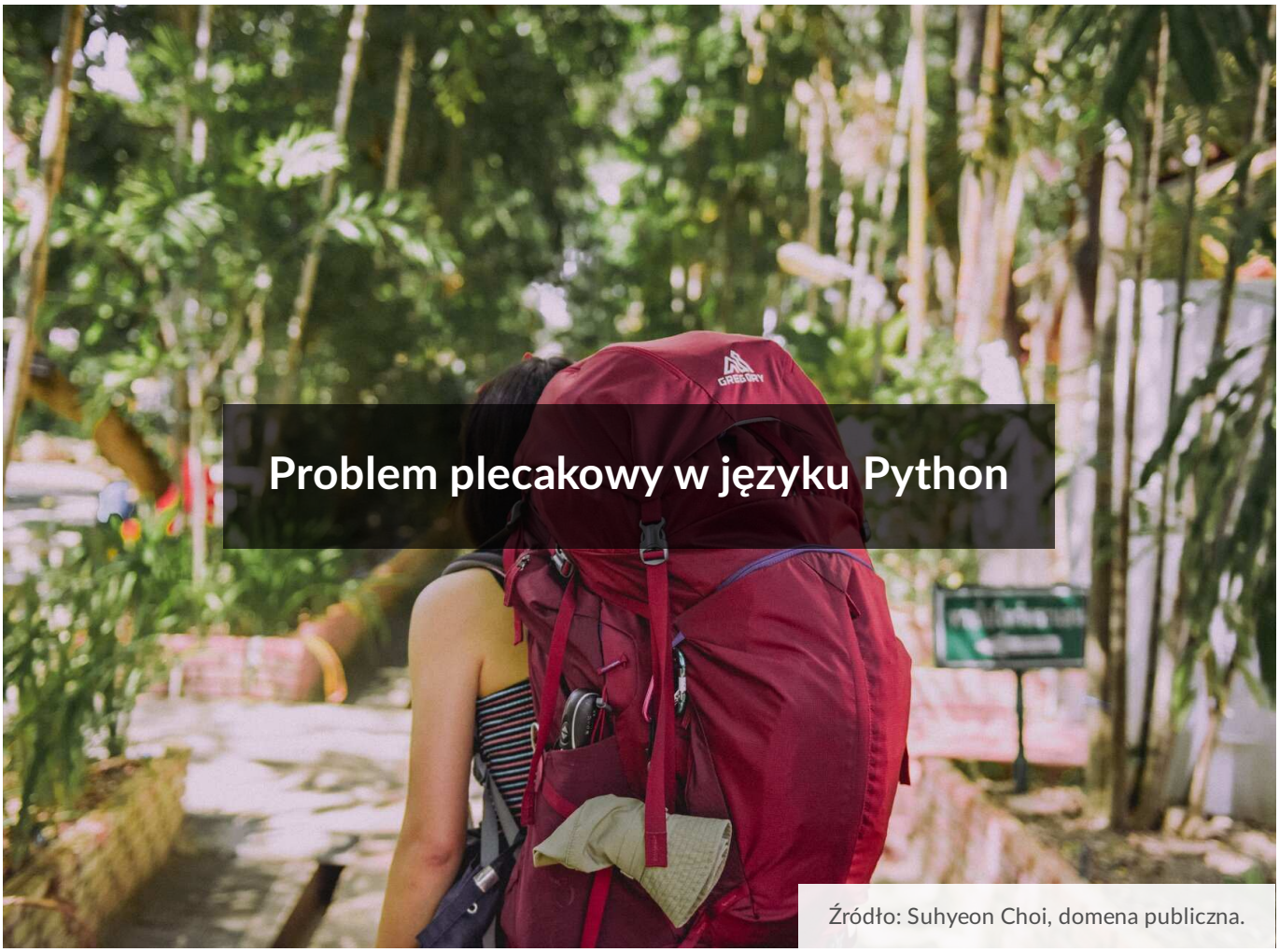
Problem plecakowy w języku Python

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela



Źródło: Suhyeon Choi, domena publiczna.

Wiemy już, że problem plecakowy dotyczy optymalnego spakowania przedmiotów o jak największej wartości do plecaka mogącego pomieścić ładunek o określonej wadze.
W e-materiale Problem plecakowy prześledziliśmy rozwiązanie tego zagadnienia za pomocą algorytmu zachłannego.

W tym e-materiale zaimplementujemy je w języku Python.

Implementacje w pozostałych językach programowania zostały omówione w e-materiałach:

- Problem plecakowy w języku C++,
- Problem plecakowy w języku Java.

Twoje cele

- Przeanalizujesz zapisane w języku Python rozwiązania trzech wariantów problemu plecakowego, które wykorzystują algorytmy zachłanne.
- Przypomnisz, czym charakteryzują się ogólne, decyzyjne oraz ciągłe problemy plecakowe.
- Zaimplementujesz rozwiązanie problemu plecakowego w języku Python.
- Rozwiążesz ćwiczenia wymagające znajomości problemu plecakowego.

Przeczytaj

Implementacja rozwiązań różnych wariantów problemu plecakowego w języku Python

> **Ważne!**
>
> W kolejnych implementacjach wykorzystamy funkcję pomocniczą `sortuj`, której zadaniem jest posortowanie dostępnych przedmiotów nierosnąco ze względu na iloraz wartości oraz wag. Użyjemy zmodyfikowanego algorytmu sortowania bąbelkowego (nie będziemy go tu omawiać).

```
def sortuj(wartoscDoWagi, indeksy, n):
    for i in range(n - 1):
        for j in range(n - i - 1):
            if wartoscDoWagi[j] < wartoscDoWagi[j + 1]:
                wartoscDoWagi[j], wartoscDoWagi[j + 1] = wartos
                indeksy[j], indeksy[j + 1] = indeksy[j + 1], in
```

> **Ważne!**
>
> Podejście zachłanne może nie dać najlepszego możliwego wyniku w wypadku problemu plecakowego.

Ogólny problem plecakowy

Specyfikacja problemu:

Dane:

- `nazwy` – tablica liczb łańcuchów znaków; nazwy poszczególnych przedmiotów
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów
- `n` – liczba naturalna; liczba rodzajów przedmiotów
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka

Wynik:

- liczby sztuk poszczególnych przedmiotów (również równe 0), których całkowita waga nie przekracza wartości przechowywanej w zmiennej `pojemnosc`, a których całkowita wartość jest największa wśród możliwych wypełnień plecaka rzeczami o takiej wadze, która nie przekracza wartości zmiennej `pojemnosc`

- całkowita wartość spakowanego plecaka

Zaprezentowane rozwiązania będziemy testować dla następujących danych:

- maksymalna pojemność plecaka wynosi **41**;
- do dyspozycji mamy **nieograniczoną liczbę sztuk każdego rodzaju przedmiotów**;
- przedmioty mają następujące wartości oraz wagi:

Przedmiot – i	**0**	**1**	**2**	**3**	**4**	**5**	
Nazwa	**anemon**	**bławatek**	**chaber**	**dziurawiec**	**eustoma**	**forsycja**	**gip**
Wartość	**21**	**15**	**4**	**4**	**14**	**1**	
Waga	**11**	**8**	**5**	**4**	**7**	**1**	

Już wiesz

Przypomnijmy, w jaki sposób działa algorytm, analizując zaprezentowane dane.

Zaczniemy od obliczenia ilorazów wartości i wag.

Obliczone ilorazy: 21/11, 15/8, 4/5, 4/4, 14/7, 1/1, 3/7.

Sortujemy je nierosnąco.

Posortowane nierosnąco ilorazy: 14/7, 21/11, 15/8, 4/4, 1/1, 4/5, 3/7.

Daje to następującą kolejność posortowanych nierosnąco przedmiotów: `eustoma`, `anemon`, `bławatek`, `dziurawiec`, `forsycja`, `chaber`, `gipsówka` (indeksy: 4, 0, 1, 3, 5, 2, 6).

Dla każdego posortowanego przedmiotu sprawdzamy, czy można go spakować, a jeśli tak, ile sztuk.

Wybieramy pierwszy przedmiot. To `eustoma`, której waga to 7. Do plecaka wkładamy 5 sztuk tego przedmiotu. Pojemność plecaka po spakowaniu tych przedmiotów wynosi 6.

Waga kolejnego przedmiotu (`anemonu`) wynosi 11, zatem nie można go spakować do plecaka.

Następny przedmiot, `bławatek`, waży 8. On również nie może zostać spakowany do plecaka.

Kolejny przedmiot to `dziurawiec`. Jego waga wynosi 4, zatem do plecaka można spakować jego jedną sztukę. Po spakowaniu kolejnych przedmiotów pojemność plecaka wynosi 2.

> Następnym przedmiotem jest `forsycja`. Jej waga to 1. Do plecaka pakujemy jej dwie sztuki.
>
> W plecaku nie ma więcej miejsca, zatem kolejne przedmioty nie mogą zostać spakowane.
>
> Obliczmy całkowitą wartość spakowanych przedmiotów:
>
> - `eustoma`: 14 × 5 sztuk
> - `dziurawiec`: 1 × 4 sztuki
> - `forsycja`: 2 × 1 sztuka
>
> Całkowita wartość spakowanych przedmiotów wynosi 76. Wykorzystaliśmy całe miejsce.

W programie wykorzystamy dwie funkcje:

- `ogolnyPlecak` – funkcja mająca na celu znalezienie największej łącznej wartości przedmiotów, które można zmieścić w plecaku o określonej pojemności z wykorzystaniem podejścia zachłannego;
- `sortuj` – funkcja pomocnicza.

Jak wspomnieliśmy, nie będziemy omawiać funkcji `sortuj`. Przejdziemy zatem do zapisania funkcji właściwej.

Zapisujemy jej nagłówek. Funkcja przyjmie pięć argumentów:

- `nazwy` – tablica łańcuchów znaków; nazwy poszczególnych przedmiotów,
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów,
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów,
- `n` – liczba naturalna; liczba rodzajów przedmiotów,
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka.

```
1 def ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
```

W pierwszym kroku tworzone są dwie kolejne tablice:

- `wartoscDoWagi` – tablica liczb rzeczywistych,
- `indeksy` – tablica liczb naturalnych.

Obie tablice składają się z `n` elementów. Zapisujemy pętlę `for`, która będzie iterować po kolejnych elementach tablic, by je zapełnić.

Każdy `i`-ty element tablicy `wartosci` dzielimy przez `i`-ty element tablicy `wagi`. Pamiętajmy o wykonaniu rzutowania, by uzyskany wynik był liczbą zmiennoprzecinkową.

Wynik dodajemy do tablicy `wartoscDoWagi`. Następnie do tablicy `indeksy` dodajemy kolejne wartości licznika, czyli zmiennej `i`.

```
def ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):
        wartoscDoWagi.append(float(wartosci[i]) / wagi[i])
        indeksy.append(i)
```

Po wyjściu z pętli wywołujemy funkcję `sortuj`.

Funkcja `sortuj` przyjmuje trzy argumenty:

- `wartoscDoWagi` – tablica liczb rzeczywistych,
- `indeksy` – tablica liczb naturalnych,
- `n` – liczba naturalna.

```
def ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):
        wartoscDoWagi.append(wartosci[i] / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)
```

Po wykonaniu sortowania tworzymy zmienną `wynik` i przypisujemy jej wartość 0. Zmienna ta będzie przechowywała sumę wartości wybranych przedmiotów.

Tworzymy również tablicę `spakowane`. Będzie przechowywała liczniki spakowanych przedmiotów.

W kolejnym kroku zapisujemy pętlę `for`, która będzie iterowała przez posortowane przedmioty. Wykorzystamy ją do wypełnienia tablicy `spakowane` zerami.

Zapisujemy kolejną pętlę `for`. Ponownie iterujemy przez posortowane przedmioty.

W pętli utworzymy zmienną `indeks`, której z każdą iteracją będziemy przypisywać wartość `i`-tego elementu tablicy `indeksy` – pobieramy w ten sposób indeks bieżącego przedmiotu.

Zapisujemy pętlę `while`. Będzie ona wykonywała się tak długo, jak długo waga pakowanego przedmiotu będzie albo mniejsza od pojemności, albo jej równa. Dzięki tej pętli kolejne przedmioty będą mogły być dodawane więcej niż jeden raz.

```
def ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):
        wartoscDoWagi.append(wartosci[i] / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)

    wynik = 0
    spakowane = []
    for i in range(n):
        spakowane.append(0)

    for i in range(n):
        indeks = indeksy[i]
        while wagi[indeks] <= paojemnosc:
```

Jeśli przedmiot o danym indeksie zmieści się w plecaku, waga przedmiotu odejmowana jest od dostępnej pojemności plecaka (element tablicy `wagi` o indeksie `indeks` jest odejmowany od wartości zmiennej `pojemnosc`). Równocześnie dodajemy wartość tego przedmiotu do wyniku (element tablicy `wartosci` o indeksie `indeks` jest dodawany do wartości zmiennej `wynik`). Inkrementujemy wartość elementu tablicy `spakowane` o indeksie `indeks`.

```
def ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):
        wartoscDoWagi.append(wartosci[i] / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)

    wynik = 0
    spakowane = []
```

```
    for i in range(n):
        spakowane.append(0)

    for i in range(n):
        indeks = indeksy[i]
        while wagi[indeks] <= pojemnosc:
            pojemnosc -= wagi[indeks]
            wynik += wartosci[indeks]
            spakowane[indeks]++
```

Po wyjściu z pętli `for` zapisujemy kolejną pętlę `for`. Wykorzystamy ją do wypisywania informacji na temat tego, ile sztuk danego rodzaju przedmiotu zostało spakowanych.

Wypisujemy całkowitą wartość spakowanych przedmiotów.

```
def ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):
        wartoscDoWagi.append(wartosci[i] / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)

    wynik = 0
    spakowane = []
    for i in range(n):
        spakowane.append(0)

    for i in range(n):
        indeks = indeksy[i]
        while wagi[indeks] <= pojemnosc:
            pojemnosc -= wagi[indeks]
            wynik += wartosci[indeks]
            spakowane[indeks]++

    for i in range(n):
        print("Liczba spakowanych sztuk przedmiotu " + nazwy[i] +

    print("\nWartosc spakowanych przedmiotow: ", wynik
```

> **Ważne!**
>
> W **linii 23** wykonujemy rzutowanie typu (zmiana typu całkowitoliczbowego na łańcuch znaków), by dokonać konkatenacji łańcuchów znaków, a w konsekwencji uniknąć znaków spacji umieszczanych między zmiennymi a łańcuchami znaków.

Dodajemy wywołanie funkcji.

Pamiętamy również o dodaniu funkcji `sortuj`.

Oto kompletny kod programu:

```
1  def sortuj(wartoscDoWagi, indeksy, n):
2      for i in range(n - 1):
3          for j in range(n - i - 1):
4              if wartoscDoWagi[j] < wartoscDoWagi[j + 1]:
5                  wartoscDoWagi[j], wartoscDoWagi[j + 1] = wartoscD
6                  indeksy[j], indeksy[j + 1] = indeksy[j + 1], inde
7
8  def ogolnyPlecak(nazwy, wartosci, wagi, n , pojemnosc):
9      wartoscDoWagi = []
10     indeksy = []
11     for i in range(n):
12         wartoscDoWagi.append(wartosci[i] / wagi[i])
13         indeksy.append(i)
14
15     sortuj(wartoscDoWagi, indeksy, n)
16
17     wynik = 0
18     spakowane = []
19     for i in range(n):
20         spakowane.append(0)
21
22     for i in range(n):
23         indeks = indeksy[i]
24         while wagi[indeks] <= pojemnosc:
25             pojemnosc -= wagi[indeks]
26             wynik += wartosci[indeks]
27             spakowane[indeks] += 1
28
29     for i in range(n):
30         print("Liczba spakowanych sztuk przedmiotu " + nazwy[i] +
```

```
31
32      print("\nWartosc spakowanych przedmiotow:", wynik)
33
34  nazwy = ["anemon", "bławatek", "chaber", "dziurawiec", "eustoma",
35  wartosci = [21, 15, 4, 4, 14, 1, 3]
36  wagi = [11, 8, 5, 4, 7, 1, 7]
37  n = 7
38  pojemnosc = 41
39  ogolnyPlecak(nazwy, wartosci, wagi, n, pojemnosc)
```

Wynik działania programu:

```
1  Liczba spakowanych sztuk przedmiotu anemon (indeks 0): 0
2  Liczba spakowanych sztuk przedmiotu bławatek (indeks 1): 0
3  Liczba spakowanych sztuk przedmiotu chaber (indeks 2): 0
4  Liczba spakowanych sztuk przedmiotu dziurawiec (indeks 3): 1
5  Liczba spakowanych sztuk przedmiotu eustoma (indeks 4): 5
6  Liczba spakowanych sztuk przedmiotu forsycja (indeks 5): 2
7  Liczba spakowanych sztuk przedmiotu gipsówka (indeks 6): 0
8
9  Wartosc spakowanych przedmiotow: 76
```

Decyzyjny problem plecakowy

Specyfikacja problemu:

Dane:

- `nazwy` – tablica liczb łańcuchów znaków; nazwy poszczególnych przedmiotów
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów
- `n` – liczba naturalna; liczba rodzajów przedmiotów
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka

Wynik:

- spakowane przedmioty, których całkowita waga nie przekracza wartości przechowywanej w zmiennej `pojemnosc`, a których całkowita wartość jest największa wśród możliwych wypełnień plecaka rzeczami o takiej wadze, która nie przekracza wartości zmiennej `pojemnosc`; pakujemy po jednej sztuce przedmiotu każdego rodzaju
- całkowita wartość spakowanego plecaka

Zaprezentowane rozwiązania będziemy testować dla następujących danych:

- maksymalna pojemność plecaka wynosi **21**;
- do dyspozycji mamy **po jednej sztuce przedmiotu każdego rodzaju**;
- przedmioty mają następujące wartości oraz wagi:

Przedmiot – i	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Nazwa	**aparat**	**baterie**	**czajnik**	**dmuchany materac**	**ekologiczny prysznic**	**filtr**	**garnus**
Wartość	**4**	**15**	**6**	**13**	**5**	**12**	**3**
Waga	**8**	**3**	**2**	**6**	**15**	**16**	**2**

Pojemność plecaka to 21.

Już wiesz

Przypomnijmy, w jaki sposób działa algorytm, analizując podane dane.

Zaczniemy od obliczenia ilorazów wartości i wag: 4/8, 15/3, 6/2, 13/6, 5/15, 12/16, 3/2.

Sortujemy je nierosnąco.

Posortowane nierosnąco ilorazy: 15/3, 6/2, 13/6, 3/2, 12/16, 4/8, 5/15.

Daje to następującą kolejność posortowanych nierosnąco przedmiotów: `baterie`, `czajnik`, `dmuchany materac`, `garnuszek`, `filtr`, `aparat`, `ekologiczny prysznic` (indeksy: 1, 2, 3, 6, 5, 0, 4).

Dla każdego posortowanego przedmiotu sprawdzamy, czy można go spakować.

Wybieramy pierwszy przedmiot. To `baterie`, których waga to 3. Waga jest mniejsza od pojemności, zatem pakujemy je do plecaka. Pojemność plecaka po spakowaniu `baterii` wynosi 18.

Waga kolejnego przedmiotu (`czajnika`) wynosi 2, zatem można spakować ten przedmiot. Pojemność plecaka to 16.

Kolejnym przedmiotem jest `dmuchany materac`. Waga tego przedmiotu to 6. Jest mniejsza od pojemności plecaka, zatem przedmiot zostaje zapakowany. Pozostała pojemność plecaka wynosi 10.

Kolejny przedmiot, `garnuszek`, waży 2. Może zostać spakowany, a pojemność po jego spakowaniu wynosi 8.

> Waga `filtra`, następnego przedmiotu, wynosi 16. Jest większa od tego, jaka jest pozostała pojemność plecaka, zatem nie można go spakować.
>
> `Aparat` jest kolejnym przedmiotem, a jego waga wynosi 8. Może zostać spakowany do plecaka.
>
> Wypełniliśmy tym samym całe miejsce w plecaku.
>
> Obliczmy całkowitą wartość spakowanych przedmiotów:
>
> - `baterie`: 15
> - `czajnik`: 6
> - `dmuchany materac`: 13
> - `garnuszek`: 3
> - `aparat`: 4
>
> Całkowita wartość spakowanych przedmiotów wynosi 41. Wykorzystaliśmy całe miejsce.

Decyzyjny problem plecakowy wymaga, by każdy przedmiot był dodawany do plecaka maksymalnie jeden raz. W programie wymaga to zmiany niewielkiej części programu.

Konieczne jest pominięcie pętli `while` w tym fragmencie i wykorzystanie instrukcji warunkowej `if`:

```
for i in range(n):
    indeks = indeksy[i]
    if wagi[indeks] <= pojemnosc:
        pojemnosc -= wagi[indeks]
        wynik += wartosci[indeks]
        print("Spakowano przedmiot " + nazwy[indeks] + " (indeks
```

Zwróć uwagę, że w kodzie nie pojawia się również tablica `spakowane`, ponieważ w przypadku problemu decyzyjnego nie ma potrzeby liczenia, ile sztuk danego przedmiotu spakowano (do dyspozycji jest tylko jedna sztuka danego przedmiotu).

Oto kompletny kod programu:

```
def sortuj(wartoscDoWagi, indeksy, n):
    for i in range(n - 1):
        for j in range(n - i - 1):
            if wartoscDoWagi[j] < wartoscDoWagi[j + 1]:
                wartoscDoWagi[j], wartoscDoWagi[j + 1] = wartoscD
                indeksy[j], indeksy[j + 1] = indeksy[j + 1], inde
```

```
7
8  def decyzyjnyPlecak(nazwy, wartosci, wagi, n, pojemnosc):
9      wartoscDoWagi = []
10     indeksy = []
11     for i in range(n):
12         wartoscDoWagi.append(wartosci[i] / wagi[i])
13         indeksy.append(i)
14
15     sortuj(wartoscDoWagi, indeksy, n)
16
17     wynik = 0
18     spakowane = []
19     for i in range(n):
20         spakowane.append(0)
21
22     for i in range(n):
23         indeks = indeksy[i]
24         if wagi[indeks] <= pojemnosc:
25             pojemnosc -= wagi[indeks]
26             wynik += wartosci[indeks]
27             print("Spakowano przedmiot " + nazwy[indeks] + " (ind
28
29     print("\nWartosc spakowanych przedmiotow:", wynik)
30
31 nazwy = ["aparat", "baterie", "czajnik", "dmuchany materac", "eko
32 wartosci = [4, 15, 6,13, 5, 12, 3]
33 wagi = [8, 3, 2, 6, 15, 16, 2]
34 n = 7
35 pojemnosc = 21
36 decyzyjnyPlecak(nazwy, wartosci, wagi, n, pojemnosc)
```

Wynik działania programu

```
1 Spakowano przedmiot baterie (indeks 1)
2 Spakowano przedmiot czajnik (indeks 2)
3 Spakowano przedmiot dmuchany materac (indeks 3)
4 Spakowano przedmiot garnuszek (indeks 6)
5 Spakowano przedmiot aparat (indeks 0)
6
7 Wartosc spakowanych przedmiotow: 41
```

Słownik

algorytm zachłanny

algorytm wybierający na każdym kroku lokalnie najlepsze rozwiązanie, licząc, że doprowadzi to do globalnie optymalnego rozwiązania problemu

optymalizacja

(od łac. *optimus* – najlepszy) poprawa wydajności algorytmu uzyskana dzięki zmniejszeniu liczby operacji niezbędnych do otrzymania wyniku

Prezentacja multimedialna

Ciągły problem plecakowy

Jak już wiemy, **ciągły problem plecakowy** to taki wariant problemu plecakowego, w którym możemy pakować do plecaka fragmenty (części) przedmiotów.

W tej sekcji przedstawimy, w jaki sposób rozwiązać ciągły problem plecakowy, wykorzystując algorytm zachłanny.

Zaprezentowane rozwiązania będziemy testować dla następujących danych:

- maksymalna pojemność plecaka wynosi 27;
- możemy pakować części przedmiotów, ale do dyspozycji jest tylko **po jednej sztuce przedmiotu każdego rodzaju**;
- przedmioty, które mamy do dyspozycji, mają następujące wartości oraz wagi:

Przedmiot – i	**0**	**1**	**2**	**3**	**4**	
Nazwa	**akwarele**	**blok rysunkowy**	**cienkopisy**	**dzienniczek**	**encyklopedia**	**f**
Wartość	**3**	**17**	**2**	**1**	**5**	
Waga	**9**	**8**	**6**	**6**	**15**	

Już wiesz

Przypomnijmy, w jaki sposób działa algorytm, analizując podane dane.

Zaczniemy od obliczenia ilorazów wartości i wag.

Obliczone ilorazy: 3/9, 17/8, 2/6, 1/6, 5/15, 3/7, 4/2.

Następnie sortujemy je nierosnąco.

Posortowane nierosnąco ilorazy: 17/8, 4/2, 3/7, 3/9, 2/6, 5/15, 1/6.

Daje to następującą kolejność posortowanych nierosnąco przedmiotów:
`blok rysunkowy`, `gumki do mazania`, `flamastry`, `akwarele`, `cienkopisy`, `encyklopedia`, `dzienniczek` (indeksy: 1, 6, 5, 0, 2, 4, 3).

Dla każdego posortowanego przedmiotu sprawdzamy, czy można go spakować w całości, a jeśli nie, to w jakiej części.

Wybieramy pierwszy przedmiot. To `blok rysunkowy`, którego waga to 8. Jego waga jest mniejsza od pojemności plecaka, zatem możemy spakować ten przedmiot. Pojemność plecaka po spakowaniu tego przedmiotu wynosi 19.

Waga kolejnego przedmiotu (`gumek do mazania`) wynosi 2, zatem można spakować ten przedmiot. Pojemność plecaka po jego spakowaniu to 17.

Kolejnym przedmiotem są `flamastry`. Waga tego przedmiotu to 7. Jest mniejsza od pojemności plecaka, zatem zostaje zapakowany. Pozostała pojemność plecaka wynosi 10.

Kolejnym przedmiotem są `akwarele`, które ważą 9. Mogą zostać spakowane, a pojemność plecaka po ich spakowaniu wynosi 1.

Waga `cienkopisów`, następnego przedmiotu, wynosi 6. Jest większa od tego, jaka jest pozostała pojemność plecaka, zatem nie można ich spakować w całości.

Należy zatem obliczyć, jaką część można spakować. By to zrobić, dzielimy pozostałą pojemność plecaka przez wagę przedmiotu. Wynik to 1/6. By obliczyć, jaką wartość ma zapakowana część przedmiotu, uzyskany wynik mnożymy przez wartość całego przedmiotu.

Uzyskana wartość to 2 × 1/6, co daje 2/6. Po zaokrągleniu do trzeciego miejsca po przecinku wynik wynosi 0,333.

Obliczmy całkowitą wartość spakowanych przedmiotów:

- `blok rysunkowy`: 17
- `gumki do mazania`: 4
- `flamastry`: 3
- `akwarele`: 3
- `cienkopisy`: 0,333

Całkowita wartość spakowanych przedmiotów wynosi 27,333. Wykorzystaliśmy całe miejsce.

Polecenie 1

Zapisz program, który rozwiąże **ciągły problem plecakowy**. Do dyspozycji masz po **jednej sztuce** każdego przedmiotu.

Wynik zaokrąglij do trzech miejsc po przecinku.

Specyfikacja problemu:

Dane:

- `nazwy` – tablica łańcuchów znaków; nazwy poszczególnych przedmiotów
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów
- `n` – liczba naturalna; liczba rodzajów przedmiotów
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka

Wynik:

- spakowane przedmioty (w całości lub części), których całkowita waga nie przekracza wartości przechowywanej w zmiennej `pojemnosc`, a których całkowita wartość jest największa wśród możliwych wypełnień plecaka rzeczami o takiej wadze, która nie przekracza wartości zmiennej `pojemnosc`; pakujemy po jednej sztuce przedmiotu każdego rodzaju
- całkowita wartość spakowanego plecaka zaokrąglona do trzech miejsc po przecinku

Działanie programu przetestuj dla następujących danych:

```
nazwy = ["akwarele", "blok rysunkowy",
         "cienkopisy", "dzienniczek",
         "encyklopedia", "flamastry",
         "gumki do mazania"]
```

```
wartosci = [3, 17, 2, 1, 5, 3, 4]
wagi = [9, 8, 6, 6, 15, 7, 2]
n = 7
pojemnosc = 27
```

Przykładowe wyniki dla podanych danych:

```
Spakowano przedmiot blok rysunkowy (indeks 1)
Spakowano przedmiot gumki do mazania (indeks 6)
Spakowano przedmiot flamastry (indeks 5)
Spakowano przedmiot akwarele (indeks 0)
Spakowano 0.167 przedmiotu cienkopisy (indeks 2)

Wartosc spakowanych przedmiotow: 27.333
```

```
def sortuj(wartoscDoWagi, indeksy, n):
    for i in range(n - 1):
        for j in range(n - i - 1):
            if wartoscDoWagi[j] < wartoscDoWagi[j + 1]:
                wartoscDoWagi[j], wartoscDoWagi[j + 1] = wartoscDoWagi[j + 1], wartoscDoWagi[j]
                indeksy[j], indeksy[j + 1] = indeksy[j + 1], indeksy[j]

def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):

# Uzupełnij kod

nazwy = ["akwarele", "blok rysunkowy", "cienkopisy",
```

```

```

Polecenie 2

Zapoznaj się z prezentacją przedstawiającą rozwiązanie ciągłego problemu plecakowego.

1

Problem plecakowy należy do tzw. problemów NP-trudnych, czyli jest problemem obliczeniowym, którego rozwiązanie jest co najmniej tak trudne, jak rozwiązanie każdego problemu z klasy NP (czyli takiego, dla którego czas potrzebny na rozwiązanie rośnie bardzo szybko). Stephen Cook jako pierwszy wykazał istnienie problemu NP-zupełnego.
Źródło: Jiří Janíček, pl.wikipedia.org, CC BY-SA 3.0

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

By upewnić się, że w plecaku nie pozostanie niewykorzystane miejsce, możemy pakować ułamkowe części przedmiotów, które zachowają **proporcjonalną** wartość. Taki wariant problemu plecakowego nazywamy **ciągłym problemem plecakowym**.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

W programie wykorzystamy dwie funkcje:

- `ciaglyPlecak` — funkcja mająca na celu znalezienie maksymalnej wartości przedmiotów, które można zmieścić w plecaku o określonej pojemności, używając podejścia zachłannego;
- `sortuj` — funkcja pomocnicza.

W rozwiązaniu wykorzystamy algorytm sortowania bąbelkowego.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

Przechodzimy do zapisania funkcji właściwej. Zapisujemy jej nagłówek. Funkcja przyjmie pięć argumentów:

- `nazwy` — tablica łańcuchów znaków; nazwy poszczególnych przedmiotów;
- `wartosci` — tablica liczb naturalnych; wartości poszczególnych przedmiotów;
- `wagi` — tablica liczb naturalnych; wagi poszczególnych przedmiotów;

- n — liczba naturalna; liczba rodzajów przedmiotów;
- `pojemnosc` — liczba naturalna; maksymalna pojemność plecaka.

```
def ciaglyPlecak(nazwy, wartości, wagi, pojemnosc, n):
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

W pierwszym kroku tworzone są dwie kolejne tablice:

- `wartoscDoWagi` — tablica liczb rzeczywistych;
- `indeksy` — tablica liczb naturalnych.

Obie tablice składają się z n elementów. Zapisujemy pętlę `for`, która będzie iterować po kolejnych elementach tablic, by je zapełnić.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

Każdy `i`-ty element tablicy `wartosci` dzielimy przez `i`-ty element tablicy `wagi`. Wynik zapisujemy do tablicy `wartoscDoWagi`. Następnie do tablicy `indeksy` zapisujemy wartość licznika, czyli zmiennej `i`.

```
def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):
    wartoscDoWagi = []
```

```
    indeksy = []
    for i in range(n):

 wartoscDoWagi.append((wartosci[i]) / wagi[i])
        indeksy.append(i)
```

Po wyjściu z pętli wywołujemy funkcję `sortuj`. Przyjmuje ona trzy argumenty:

- `wartoscDoWagi` — tablica liczb rzeczywistych;
- `indeksy` — tablica liczb naturalnych;
- n — liczba naturalna.

```
def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):

 wartoscDoWagi.append((wartosci[i]) / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

Po wykonaniu sortowania tworzymy zmienną `wynik` i przypisujemy jej wartość 0. Zmienna ta będzie przechowywała sumę wartości wybranych przedmiotów.

W kolejnym kroku zapisujemy pętlę `for`. Będzie ona iterowała przez posortowane przedmioty. W pętli utworzymy również zmienną `indeks`,

której z każdą iteracją będziemy przypisywać wartość `i`-tego elementu tablicy `indeksy` — pobieramy w ten sposób indeks bieżącego przedmiotu.

```
def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):

 wartoscDoWagi.append((wartosci[i]) / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)

    wynik = 0
    for i in range(n):
        indeks = indeksy[i]
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

W pętli zapisujemy instrukcję warunkową. Sprawdzamy, czy przedmiot o danym indeksie zmieści się w plecaku.

```
def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):

 wartoscDoWagi.append((wartosci[i]) / wagi[i])
```

```
6           indeksy.append(i)
7
8       sortuj(wartoscDoWagi,
    indeksy, n)
9
10      wynik = 0
11      for i in range(n):
12          indeks =
    indeksy[i]
13          if wagi[indeks] <=
    pojemnosc:
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

Jeśli przedmiot o danym indeksie zmieści się w plecaku (zatem instrukcja zwraca wartość logiczną `true`), waga przedmiotu odejmowana jest od dostępnej pojemności plecaka (element tablicy `wagi` o indeksie `indeks` jest odejmowany od wartości zmiennej `pojemnosc`). Równocześnie dodajemy wartość tego przedmiotu do wyniku (element tablicy `wartosci` o indeksie `indeks` jest dodawany do wartości zmiennej `wynik`). Wyświetlamy komunikat o dodanym w całości przedmiocie.

Jeśli zaś przedmiot o danym indeksie nie zmieści się w plecaku, obliczamy, jaka część przedmiotu może zostać zapakowana. Zatem do zmiennej `wynik` dodajemy iloraz pozostałej pojemności oraz elementu tablicy `wagi` o indeksie `indeks` pomnożony razy element tablicy przechowującej wartości o indeksie `indeks`. W ten sposób dodajemy częściową wartość oraz wagę przedmiotu. Wyświetlamy komunikat o dodanym częściowo przedmiocie. Pamiętajmy o zaokrągleniu.

W kolejnym kroku przerywamy działanie pętli `for`.

Wyświetlamy komunikat o całkowitej wartości spakowanych przedmiotów.

```
def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):

 wartoscDoWagi.append((wartosci[i]) / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)

    wynik = 0
    for i in range(n):
        indeks = indeksy[i]
        if wagi[indeks] <= pojemnosc:
            pojemnosc -= wagi[indeks]
            wynik += wartosci[indeks]

 print("Spakowano przedmiot " + nazwy[indeks] + " (indeks " + str(indeks) + ")")
        else:
            wynik += pojemnosc / wagi[indeks] * wartosci[indeks]

 print("Spakowano " + str(round(pojemnosc / wagi[indeks], 3)) + " przedmiotu " +
```

```
            nazwy[indeks] + " (indeks
            " + str(indeks) + ")")
20                  break
```


Jedną ze strategii rozwiązywania problemu plecakowego jest wykorzystanie programowania dynamicznego. Jego podstawy stworzył Richard Bellman, matematyk.
Źródło: Wikipedia, en.wikipedia.org, tylko do użytku edukacyjnego.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBdZ2Nqkn

Dodajemy wywołanie funkcji.

Pamiętamy o dodaniu funkcji `sortuj`.

Bierzemy również pod uwagę wymóg, by wynik został zaokrąglony do trzech miejsc po przecinku.

Oto kompletny kod programu:

```
1 def sortuj(wartoscDoWagi,
  indeksy, n):
2     for i in range(n - 1):
```

```
        for j in range(n - i - 1):
            if wartoscDoWagi[j] < wartoscDoWagi[j + 1]:
                wartoscDoWagi[j], wartoscDoWagi[j + 1] = wartoscDoWagi[j + 1], wartoscDoWagi[j]
                indeksy[j], indeksy[j + 1] = indeksy[j + 1], indeksy[j]

def ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n):
    wartoscDoWagi = []
    indeksy = []
    for i in range(n):
        wartoscDoWagi.append((wartosci[i]) / wagi[i])
        indeksy.append(i)

    sortuj(wartoscDoWagi, indeksy, n)

    wynik = 0
    for i in range(n):
        indeks = indeksy[i]
        if wagi[indeks] <= pojemnosc:
            pojemnosc -= wagi[indeks]
            wynik += wartosci[indeks]
            print("Spakowano przedmiot " + nazwy[indeks] + " (indeks " + str(indeks) + ")")
        else:
```

```
            wynik += pojemnosc / wagi[indeks] * wartosci[indeks]

 print("Spakowano " + str(round(pojemnosc / wagi[indeks], 3)) + " przedmiotu " + nazwy[indeks] + " (indeks " + str(indeks) + ")")
            break

    print("\nWartosc spakowanych przedmiotow:", round(wynik, 3))

nazwy = ["akwarele", "blok rysunkowy", "cienkopisy", "dzienniczek", "encyklopedia", "flamastry", "gumki do mazania"]
wartosci = [3, 17, 2, 1, 5, 3, 4]
wagi = [9, 8, 6, 6, 15, 7, 2]
pojemnosc = 27
n = 7

ciaglyPlecak(nazwy, wartosci, wagi, pojemnosc, n)
```

Wynik działania programu:

```
Spakowano przedmiot blok rysunkowy (indeks 1)
Spakowano przedmiot gumki do mazania (indeks 6)
Spakowano przedmiot flamastry (indeks 5)
Spakowano przedmiot akwarele (indeks 0)
Spakowano 0.167 przedmiotu cienkopisy (indeks 2)
```

```
6
7 Wartosc spakowanych
  przedmiotow: 27.333
```

Prezentacja multimedialna przedstawiająca rozwiązanie ciągłego problemu plecakowego

Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje dotyczące rozwiązywania różnych wariantów problemu plecakowego.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który rozwiąże **ogólny problem plecakowy** i wyświetli komunikat dotyczący tego, ile sztuk kolejnych przedmiotów spakowano, oraz jaka jest całkowita wartość spakowanego plecaka.

Specyfikacja problemu:

Dane:

- `nazwy` – tablica łańcuchów znaków
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów
- `n` – liczba naturalna; liczba rodzajów przedmiotów
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka

Wynik:

- liczby sztuk poszczególnych przedmiotów (również równe 0), których całkowita waga nie przekracza wartości przechowywanej w zmiennej `pojemnosc`, a których całkowita wartość jest największa wśród możliwych wypełnień plecaka rzeczami o takiej wadze, która nie przekracza wartości zmiennej `pojemnosc`
- całkowita wartość spakowanego plecaka

Działanie programu przetestuj dla następujących danych:

```
nazwy = ["apaszka", "beret", "chusteczki", "dres", "espadryle"]
wartosci = [1, 5, 4, 1, 9]
wagi = [9, 1, 2, 8, 7]
pojemnosc = 11
n = 5
```

Przykładowe wyniki dla podanych danych:

```
Liczba spakowanych sztuk przedmiotu apaszka (indeks 0): 0
Liczba spakowanych sztuk przedmiotu beret (indeks 1): 11
Liczba spakowanych sztuk przedmiotu chusteczki (indeks 2): 0
Liczba spakowanych sztuk przedmiotu dres (indeks 3): 0
Liczba spakowanych sztuk przedmiotu espadryle (indeks 4): 0

Wartosc spakowanych przedmiotow: 55
```

Twoje zadania

1. Program wyświetla informacje na temat tego, ile sztuk kolejnych przedmiotów spakowano, oraz jaka jest całkowita wartość spakowanego plecaka.

Ćwiczenie 2

Napisz program, który rozwiąże **decyzyjny problem plecakowy** i wyświetli komunikat dotyczący tego, które przedmioty spakowano, oraz jaka jest całkowita wartość spakowanego plecaka.

Specyfikacja problemu

Dane:

- `nazwy` – tablica łańcuchów znaków
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów
- `n` – liczba naturalna; liczba rodzajów przedmiotów
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka

Wynik:

- spakowane przedmioty, których całkowita waga nie przekracza wartości przechowywanej w zmiennej `pojemnosc`, a których całkowita wartość jest największa wśród możliwych wypełnień plecaka rzeczami o takiej wadze, która nie przekracza wartości zmiennej `pojemnosc`; pakujemy po jednej sztuce przedmiotu każdego rodzaju
- całkowita wartość spakowanego plecaka

Działanie programu przetestuj dla następujących danych:

```
nazwy = ["apaszka", "beret", "chusteczki", "dres", "espadryle"]
wartosci = [1, 5, 4, 1, 9]
wagi = [9, 1, 2, 8, 7]
pojemnosc = 11
n = 5
```

Przykładowe wyniki dla podanych danych:

```
Spakowano przedmiot apaszka (indeks 1)
Spakowano przedmiot beret (indeks 2)
Spakowano przedmiot chusteczki (indeks 4)

Wartosc spakowanych przedmiotow: 18
```

Twoje zadania

1. Napisz program, który rozwiąże decyzyjny problem plecakowy i wyświetli nazwy spakowanych przedmiotów, ich wartości i wagi oraz łączną wartość wszystkich spakowanych przedmiotów.

Ćwiczenie 3

Napisz program, który rozwiąże **ciągły problem plecakowy** i wyświetli komunikat dotyczący tego, które przedmioty spakowano w całości, które częściowo (oraz w jakiej części), oraz jaka jest całkowita wartość spakowanego plecaka.

Wyniki należy zaokrąglić do trzech miejsc po przecinku.

Specyfikacja problemu:

Dane:

- `nazwy` – tablica łańcuchów znaków
- `wartosci` – tablica liczb naturalnych; wartości poszczególnych przedmiotów
- `wagi` – tablica liczb naturalnych; wagi poszczególnych przedmiotów
- `n` – liczba naturalna; liczba rodzajów przedmiotów
- `pojemnosc` – liczba naturalna; maksymalna pojemność plecaka

Wynik:

- spakowane przedmioty, których całkowita waga nie przekracza wartości przechowywanej w zmiennej `pojemnosc`, a których całkowita wartość jest największa wśród możliwych wypełnień plecaka rzeczami o takiej wadze, która nie przekracza wartości zmiennej `pojemnosc`; pakujemy po jednej sztuce przedmiotu każdego rodzaju
- całkowita wartość spakowanego plecaka

Działanie programu przetestuj dla następujących danych:

```
nazwy = ["apaszka", "beret", "chusteczki", "dres", "espadryle"]
wartosci = [1, 5, 4, 1, 9]
wagi = [9, 1, 2, 8, 7]
pojemnosc = 11
```

```
n = 5
```

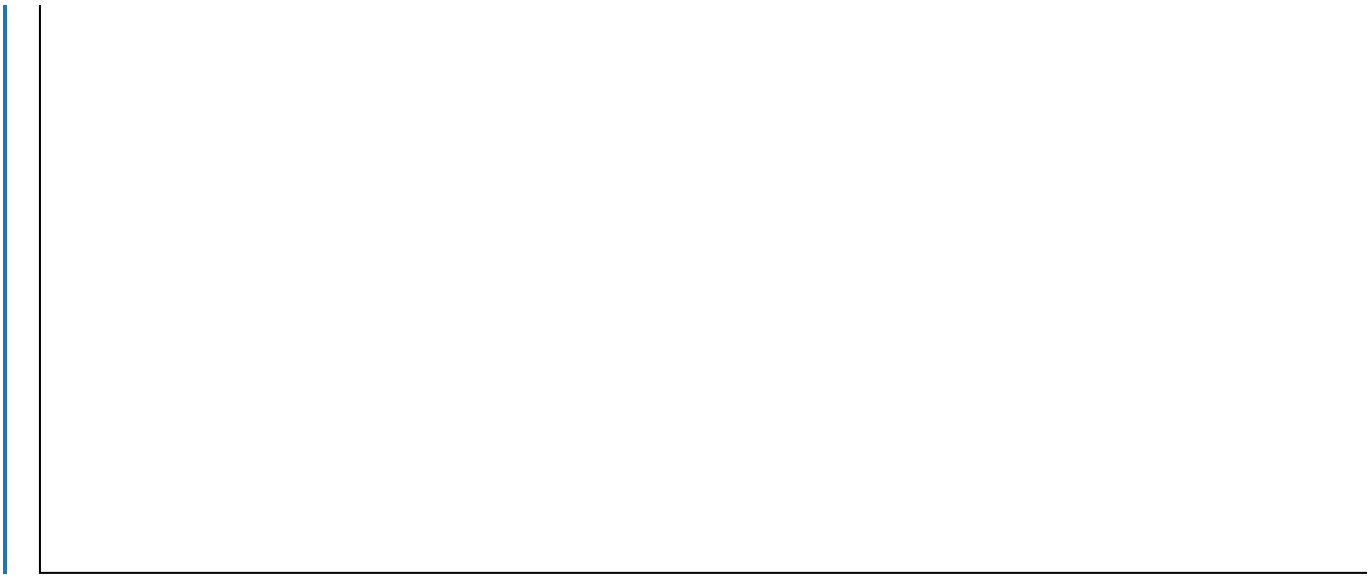
Przykładowe wyniki dla podanych danych:

```
Spakowano przedmiot beret (indeks 1)
Spakowano przedmiot chusteczki (indeks 2)
Spakowano przedmiot espadryle (indeks 4)
Spakowano 0.125 przedmiotu dres (indeks 3)

Wartosc spakowanych przedmiotow: 18.125
```

Twoje zadania

1. Napisz program, który rozwiąże ogólny problem plecakowy i wyświetli nazwy spakowanych przedmiotów, ich wartości, wagi oraz łączną wartość wszystkich spakowanych przedmiotów.



Dla nauczyciela

Autor: zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Problem plecakowy w języku Python

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

d) podejście zachłanne (do wydawania reszty, pakowania plecaka, szukania najkrótszej drogi),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz zapisane w języku Python rozwiązania trzech wariantów problemu plecakowego, które wykorzystują algorytmy zachłanne.
- Przypomnisz, czym charakteryzują się ogólne, decyzyjne oraz ciągłe problemy plecakowe.
- Zaimplementujesz rozwiązanie problemu plecakowego w języku Python.
- Rozwiążesz ćwiczenia wymagające znajomości problemu plecakowego.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;

- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. Chętna lub wybrana osoba przygotowuje krótką prezentację, w której przedstawia najważniejsze informacje dotyczące problemu plecakowego i algorytmów zachłannych.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Problem plecakowy w języku Python". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia na podstawie informacji na platformie stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. W kolejnym kroku **pracy z tekstem** na forum klasy uczniowie testują zaprezentowane programy dla różnych danych, a następnie dyskutują na temat uzyskanych wyników.
3. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie indywidualnie realizują polecenie nr 1. Zapisują ewentualne problemy i pytania. Następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
4. Uczniowie wykonują polecenie nr 2 z sekcji „Prezentacja multimedialna".
5. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1–3 z sekcji „Sprawdź się", a następnie omawiają rozwiązania na forum.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Python.

Praca domowa:

1. Uczniowie wykonują polecenie nr 3 z sekcji „Prezentacja multimedialna”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

Treści w sekcji „Prezentacja multimedialna” można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.