Na początek kilka operacji do zapamiętania:

- Metoda `hstack()` pozwala „skleić" dwie tablice (lub więcej) poprzez ustawienie ich poziomo obok siebie (jest to analogia wizualna), np. z dwóch tablic jednowymiarowych można utworzyć nową jednowymiarową.
- Metoda `vstack()` pozwala „skleić" dwie tablice (lub więcej) poprzez ustawienie pionowo na sobie (jest to analogia wizualna), np. z dwóch tablic jednowymiarowych można utworzyć tablicę dwuwymiarową, której wiersze są złożone z tablic źródłowych.
- Metoda `transpose()` pozwala odwrócić układ tablicy poprzez zamianę wierszy na kolumny i odwrotnie [*numpy2.py* (fragment)].

```
import numpy as np
t1 = np.arange(1, 6)  # Od 1 do 5
print("t1: ",t1)
t2 = np.arange(6, 11) # Od 6 do 10
print("t2: ",t2)
t3 = np.vstack((t1, t2))
print("Kształt t3 to", t3.shape)
print("t3 po zastosowaniu vstack()\n",t3)
t3 = t3.transpose()
print("t3 po transpozycji:",t3)
print("Nowy kształt t3 to", t3.shape)
print("Liczba wymiarów tablicy t3 to", t3.ndim)
t4 = np.hstack((t1, t2))
print("t4=t1 sklejona horyzontalnie z t2:",t4)
```

Wynik będzie następujący (nieco przeformatowałem układ wyświetlanych informacji w książce w celu zwiększenia czytelności wydruku):

```
t1:  [1 2 3 4 5]
t2:  [6  7  8  9 10]
Kształt t3 to  (2, 5)
t3 po zastosowaniu vstack()
 [ [1  2  3  4  5]
   [6  7  8  9 10] ]
t3 po transpozycji: [[1  6]
                     [2  7]
                     [3  8]
                     [4  9]
                     [5 10] ]
Nowy kształt t3 to (5, 2)
Liczba wymiarów tablicy t3 to 2
t4=t1 sklejona horyzontalnie z t2: [1  2  3  4  5  6  7  8  9 10]
```

Wycinki w tablicach

W Pythonie czasami używane są karkołomne konstrukcje, które przyprawiają o ból głowy osoby początkujące. Na pewno należą do nich wycinki, o których wspominałem już szerzej w rozdziale 5., pokazując operacje na napisach.

Uogólniona składnia komponowania wycinków (stosowana w napisach, listach, tablicach NumPy) jest następująca:

możemy określić parametry *od*, *do* i tzw. *krok*, czyli przeskok kolejnych wartości. Wygląda to koszmarnie i na dodatek każdy z tych parametrów może być pominięty...

Nie chcę tu zatem wprowadzać formalnych zapisów, po prostu popatrzmy na kilka przykładów użycia pythonowego operatora zakresu w celu filtrowania podzakresów na przykładzie tabeli NumPy (fragment *numpy3.py*):

```
import numpy as np
t=np.arange(10) # Tablica będzie zawierała 10 elementów: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
t1=t[:3]        # Wycinek [0,2], tj. pierwsze trzy wartości, tj. [0 1 2]
t2=t[3:]        # Wycinek [2,9], tj. pozostałe siedem wartości, tj. [3 4 5 6 7 8 9]
t3=t[5:7]       # Wycinek [5,6], tj. zakres w środku, tj. [5 6]
t4=t[::3]       # Co trzeci element, tj. [0 3 6 9]
t5=t[4::2]      # Co drugi element, zaczynając od indeksu 4, tj. [4 6 8]
t6=t[::-1]      # Wszystkie wartości, ale od końca (odwracamy tablicę), tj. [9 8 7 6 5 4 3 2 1 0]
t7=t[4::-2]     # Wszystkie co drugie wartości, od końca i zaczynając od indeksu 4, tj. [4 2 0]
```

Bardzo podobnie można odfiltrować podzakresy tablic 2D (kontynuacja *numpy3.py*).

DEKLARACJE:	ZAWARTOŚĆ:

`v=np.array( [ [1,  2,  3,  4, 5],`
`[5,  6,  7,  8, 9],`
`[9,  10, 11, 12, 13],`
`[14, 15, 16, 17, 18],`
`] )`
	`[ [1  2  3  4  5]`
	`[5  6  7  8  9]`
	`[9 10 11 12 13]`
	`[14 15 16 17 18] ]`

# Wycinek 2x4 (2 wiersze i 4 kolumny) `v1=v[:2, :4]`	`[ [1 2 3 4]` `[5 6 7 8] ]`

# Wycinek (3 wiersze) x (co 2 kolumny) 	
`v2=v[:3, ::2]`
	`[ [1  3  5]`
	`[5  7  9]`
	`[9 11 13] ]`

`v3=v[:, 1:3]`
# Wycinek (Wszystkie wiersze) x (kolumny 1. i 2.) 	
	`[ [2  3]`
	`[6  7]`
	`[10 11]`
	`[15 16] ]`

Co ważne, wycinek tablicy NumPy dalej operuje na tym samym obszarze pamięci i modyfikacja komórki w wycinku także zmieni oryginalną wartość:

```
t=np.array([1, 5, 10, 15, 20])
t_wycinek= t[0:2]   # Pierwsze dwie wartości, czyli [1, 5]
t_wycinek[0]=-1
t_wycinek[1]=-5
print(t)            # Wypisze: [-1 -5 10 15 20]
```

Jest to istotna różnica w porównaniu z listami, gdzie metody zwracały kopie oryginalnych obiektów! Czy da się jednak jawnie uzyskać wycinek będący kopią oryginalnych danych? Na szczęście tak. W nawiązaniu do przykładu wyżej: taka modyfikacja spowodowałaby pozostawienie tablicy t w stanie nienaruszonym podczas operacji na wycinku:

```
t_wycinek= t[0:2].copy()
```