**Temat/y: Wyszukiwanie liniowe/sekwencyjne z wartownikiem, zliczanie według kryterium, wyszukiwanie wartości maksymalnej i/lub minimalnej w zbiorze.**

**Autor: Piotr Świder**

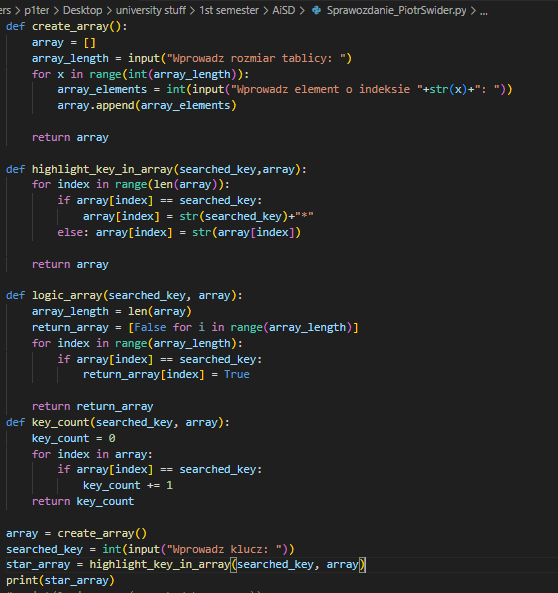
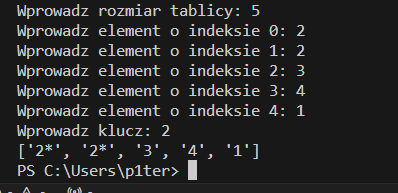
**Grupa laboratoryjna: 8**

**Data: 08/11/2023**

**Zadanie #1.**

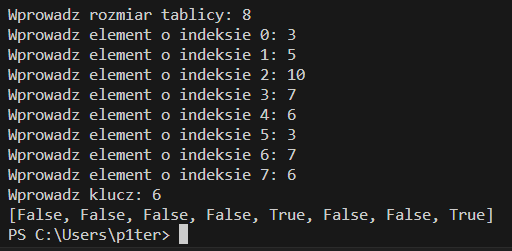
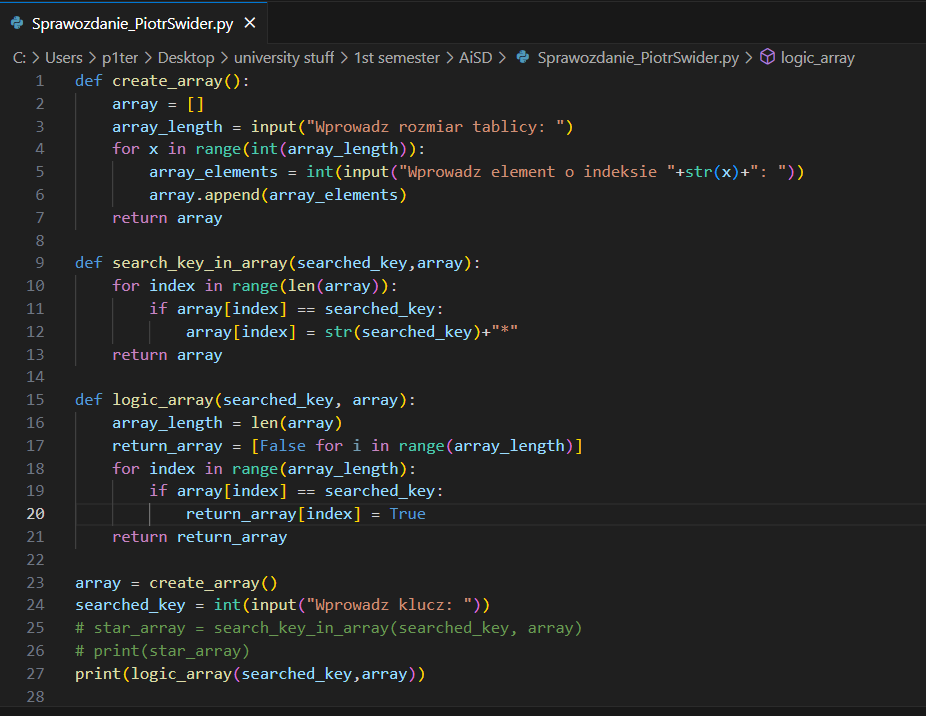
1. Przeanalizować kod powyższego programu i sprawdzić jego działanie z użyciem funkcji;
2. Zakomentować wywołanie funkcji oraz odkomentować instrukcje wyszukiwania liniowego w programie głównym. Przeanalizować działanie programu;
3. Zmodyfikować program, tak aby wyświetlał elementy tablicy (wiersz lub kolumnie) oraz oznaczył wartość klucza za pomocą gwiazdki „\*”;
4. Zmodyfikować program, tak aby zwracał wszystkie pozycje klucza z pomocą tablicy logicznej.

**Rozwiązanie zadania #1.**

1. Na początku w kodzie definiujemy funkcję nazwaną WySzukaj, która jest odpowiedzialna za wyszukiwanie elementów w tablicy. Użytkownik podając kolejno: rozmiar, elementy zbioru oraz klucz, którego ma szukać wywoływana funkcja, otrzymuje pierwszą znalezioną pozycję, na której stoi określony element lub też wartość -1 w przypadku gdy wartość danego klucza nie jest równa żadnemu z elementów.
2. Efekt po odrzuceniu działania wcześniej zdefiniowanej funkcji wyszukaj i odkomentowania algorytmu wyszukiwania liniowego jest niezmienny. Program działa w bardzo podobny sposób – jednakże znajduje się tutaj zastosowanie wartownika, który jest wstawiany na sam koniec tablicy, a jego wartość jest równoważna wartości klucza podanego przez użytkownika. Gdy program wykryje klucz na ostatniej pozycji tablicy(i na żadnej innej), czyli tej, na której został dodany wartownik – zwróci wartość -1.
3. Kod oraz wartość zwrotna programu do podpunktu c;  

Komentarz do podpunktu c; tworzymy funkcję, która stworzy tablicę o podanym rozmiarze oraz elementach wprowadzonych przez użytkownika. Następnie tworzymy funkcję, która ma za zadanie dodawać gwiazdki do poszczególnych wartości, które zostały określone przez klucz – który również został wcześniej wprowadzony przez użytkownika. Na sam koniec wywołujemy te funkcję i otrzymujemy tablice z już podpisanymi elementami poprzez „\*”.

1. Kod oraz wartość zwrotna programu do podpunktu d;



Komentarz do podpunktu d; definiujemy funkcje, w której tworzymy dodatkową tablicę bazowaną na tej stworzonej pierwotnie i wypełniamy ją wartościami – False. Następnie przechodzimy przez całą pierwotną tablicą sprawdzając jej wartości i porównując do szukanego klucza. Jeśli wartość jest równa kluczowi, zamieniamy wartość False w tablicy return\_array na wartość True (na tym samym indeksie, na którym została znaleziona poszukiwana wartość).

**Wnioski:**

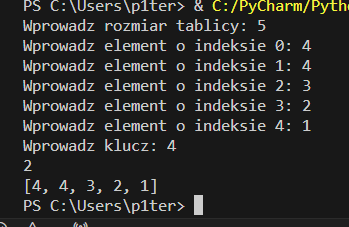
c) Wyszukując i porównując wartości elementu do klucza wystarczy dodać string „\*”.

d) W tworzeniu tablicy logicznej warto stworzyć tablicę dodatkową - pomocniczą, która będzie przechowywać określone wartości, które kolejno będziemy zamieniać.

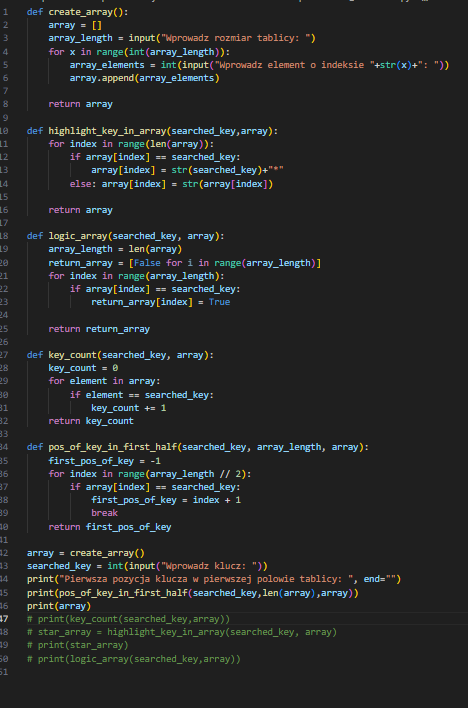
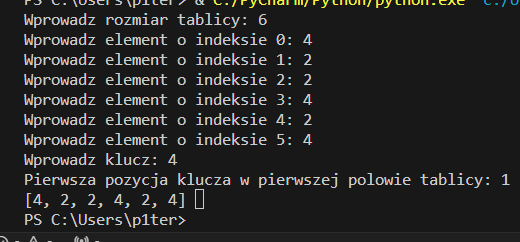
**Zadanie #2.**

1. Zmodyfikować program z zadania 1, tak aby zliczał ilość wystąpienia klucza w tablicy;
2. Zmodyfikować program, tak aby zwracał pierwsze wystąpienie (pierwszą pozycję) klucza w pierwszej połowie tablicy;
3. Zmodyfikować program, tak aby zwracał pierwsze wystąpienie (pierwszą pozycję) klucza w drugiej połowie tablicy.

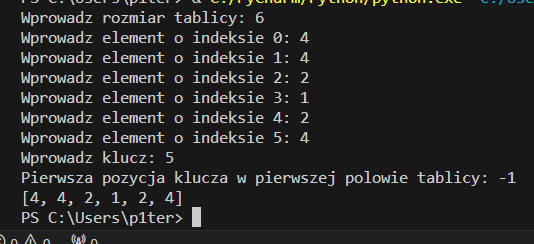
**Rozwiązanie zadania #2.**

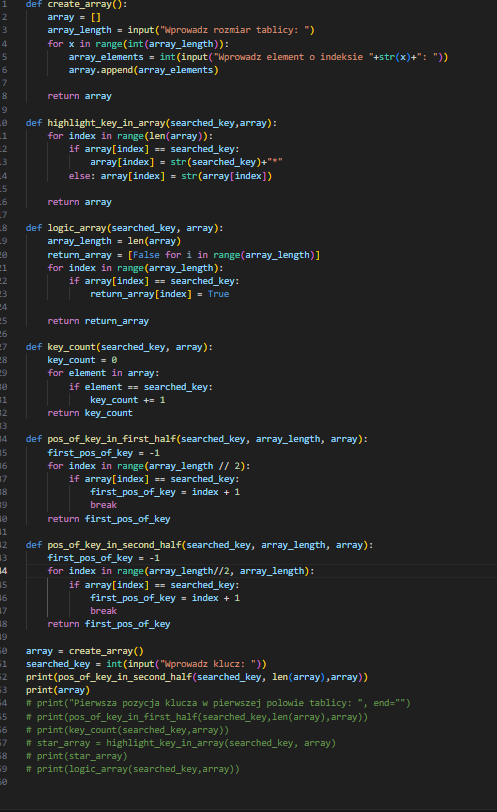
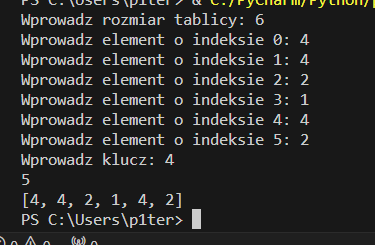
1. Kod oraz wartość zwrotna programu do podpunktu a; ****

Komentarz do podpunktu a; Tworzymy funkcję, która przechodzi po każdym elemencie tablicy i jeśli dany element jest równoważny naszemu szukanemu kluczowi dodajemy do wartości key\_count, która początkowo jest równa 0 jeden co każde spełnienie warunku.

1. Kod oraz wartość zwrotna programu do podpunktu b;  

Komentarz do podpunktu b; Tworzymy funkcje, która przechodzi po każdym elemencie tablicy array, którą ograniczamy przez podanie warunku array\_length//2 tak, aby uzyskać jedynie pierwszą połowę tablicy. Następnie sprawdzamy czy dana wartość na danym indeksie jest równa naszemu kluczowi i nadpisujemy nasz klucz oraz kończymy działanie pętli break’iem tak, aby nie uzyskać wartości kolejnej. Jeśli klucz nie występuje funkcja zwraca nam wartość -1.



c) Kod oraz wartość zwrotna programu do podpunktu c;  

Komentarz do podpunktu c; Definiujemy taką samą funkcję jak w podpunkcie b), jedynie zmieniając range(), w którym mamy wyszukiwać danego klucza z pierwszej połowy na drugą czyli przechodzimy z pętli:  na pętle: .

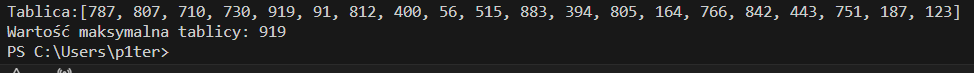
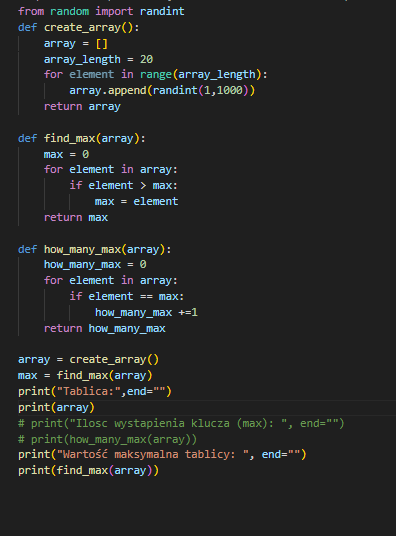
**Wnioski:**

1. Aby zliczyć wystąpienia klucza najlepiej posłużyć się pomocniczą zmienną w tym przypadku key\_count.
2. Aby sprawdzić pierwszą połowę tablicy wystarczy określić jej długość oraz podzielić ją przez dwa.
3. Podobna sytuacja jak w podpunkcie b) – jedyna zmiana w konstrukcji pętli.

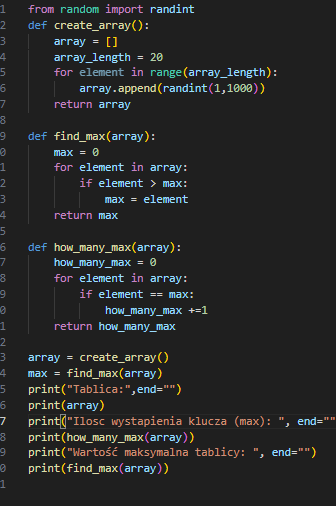
**Zadanie #3.**

1. Przeanalizować i uzupełnić poniższy program. Sprawdzić działanie programu;
2. Wyszukać pozycję klucza. Kluczem jest wartość maksymalna (max);
3. Zliczyć ile razy wystąpiła wartość max;
4. Wyszukać wszystkie pozycje wartości max;
5. Wyszukać pozycję klucza. Kluczem jest min\_dod (najmniejsza wartość dodatnia);
6. Wyszukać pozycję klucza. Kluczem jest min.

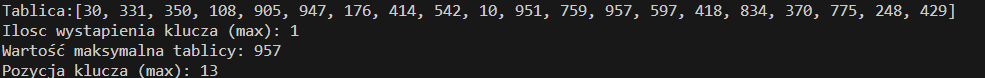
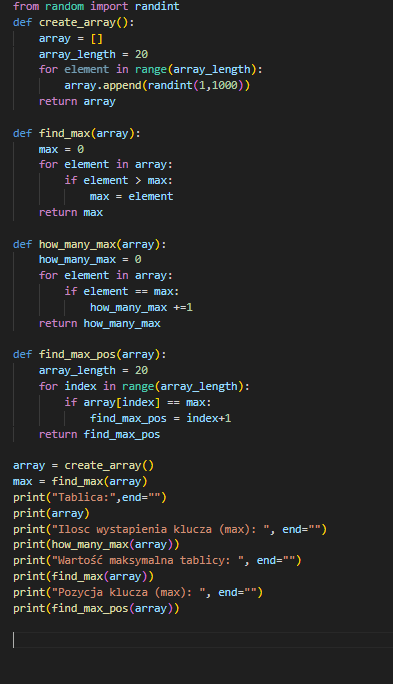
**Rozwiązanie zadania #3.**

1. Poniższy program wykonuje wyszukiwanie wartości maksymalnej czy też jeśli odkomentujemy poszczególne linijki wartości minimalnej.
2. Kod oraz wartość zwrotna programu do podpunktu b; 

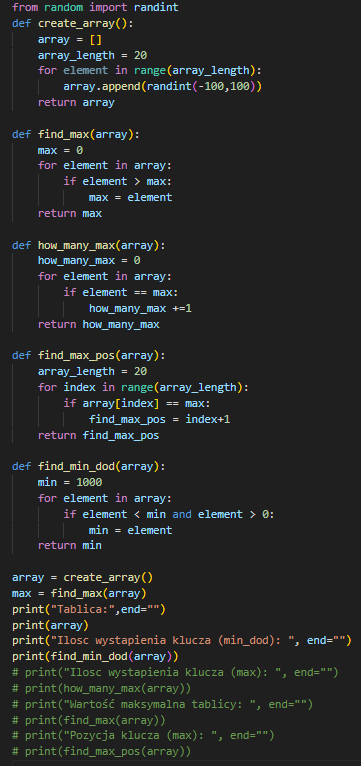
Komentarz do podpunktu b; Definiujemy funkcję tworzącą tablicę 20 elementową, której wartości wachają się od 1 do 1000, zwracamy stworzoną tablicę. Następnie definiujemy funkcję find\_max, która przechodzi przez każdy element stworzonej wcześniej tablicy i nadpisuje wartość max, która początkowo wynosi 0 – na coraz to większe jeśli one występują.

1. Kod oraz wartość zwrotna programu do podpunktu c; 

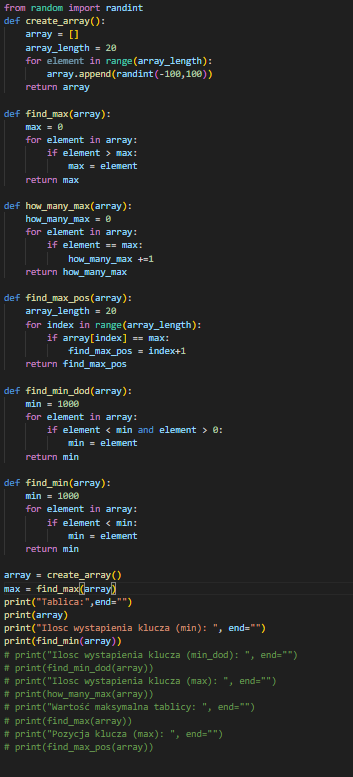
Komentarz do podpunktu c; Tworzymy kolejną funkcję how\_many\_max, która ma za zadanie zliczać ilość wystąpień wartości maksymalnej.

1. Kod oraz wartość zwrotna programu do podpunktu d; 

Komentarz do podpunktu d; Tworzymy funkcję find\_max\_pos, która przechodzi przez długość całej tablicy a następnie sprawdza, która z wartości jest równa wartości maksymalnej. Następnie pod zmienną find\_max\_pos zapisujemy indeks i dodajemy jedynkę, aby uniknąć błędnej pozycji, która określana jest w ten sposób przez indeksowanie od 0.

1. Kod oraz wartość zwrotna programu do podpunktu e; 

Komentarz do podpunktu e; Zmieniony został randint – a w zasadzie zakres generowania liczb z (1,1000) na (-100,100) tak, aby sens funkcji find\_min\_dod porównywał również wartości ujemne.

1. Kod oraz wartość zwrotna programu do podpunktu f; 

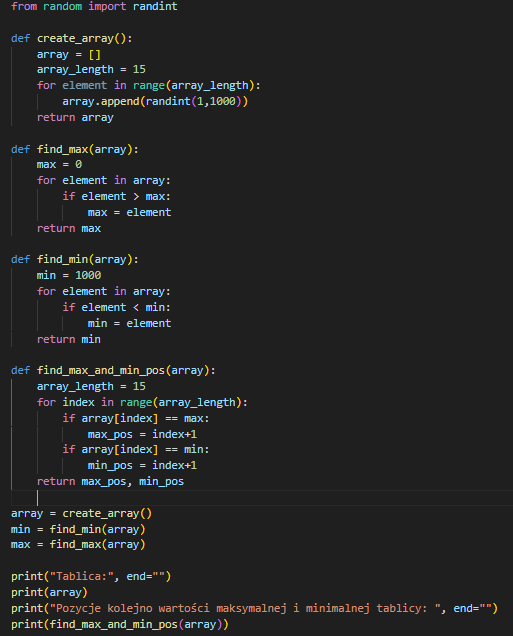
Komentarz do podpunktu f; Tworzymy taką samą funkcję jak w podpunkcie e), jedyna zmiana to warunek – w którym bierzemy pod uwagę również liczby dodatnie, czyli nie dodajemy warunku, że wartość elementu w zbiorze musi być większa niż zero.

**Wnioski z zadania #3**: Program ma na celu wyszukiwania konkretnych wartości, które zostały określone przez funkcję.

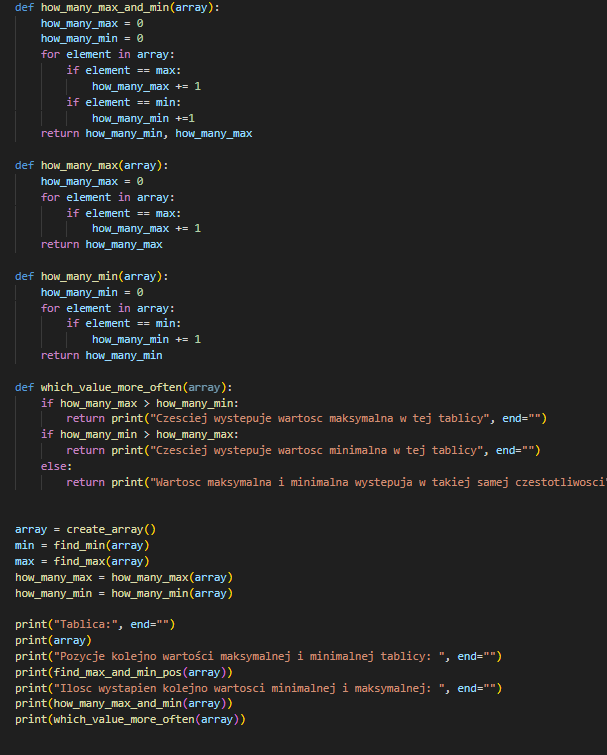
**Zadanie #4.**

1. Przeanalizować i uzupełnić poniższy program. Sprawdzić działanie programu;
2. Wyszukać pozycję kluczy. Kluczem jest wartość maksymalna (max/min). (pierwsze wystąpienie);
3. Zliczyć ile razy wystąpiła wartość max oraz min. Podać która wartość wystąpiła częściej;
4. Podać która z wartości wystąpiła wcześniej;
5. Obliczyć różnicę pozycji między min i max;
6. Zmodyfikować program tak aby rozpocząć wyszukiwanie pozycji min i max od kończ tablicy.

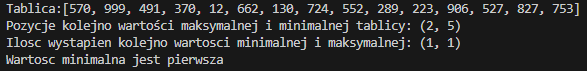
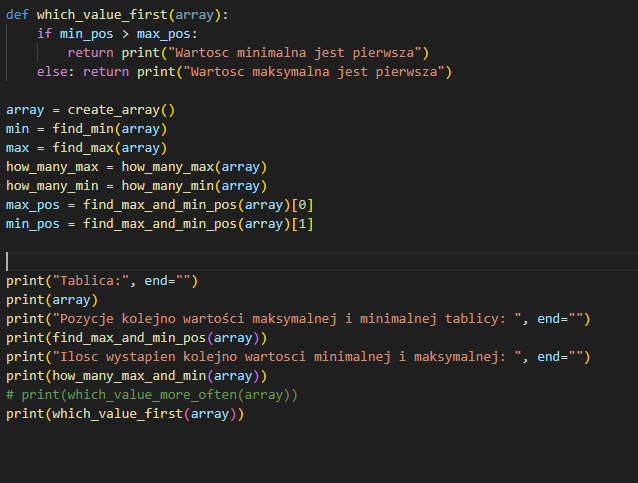
Rozwiązanie zadania #4.

1. Program wypisuje w kolumnie tablicę oraz wypisuję wartość maksymalną i minimalną.
2. Kod oraz wartość zwrotna programu do podpunktu b; 

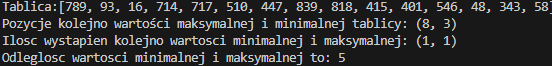
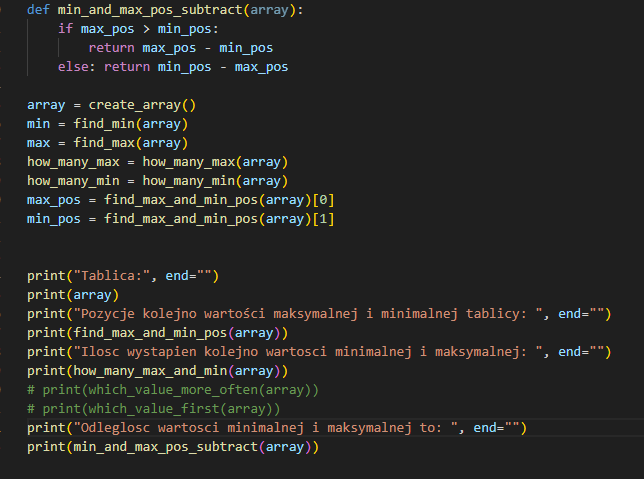
Komentarz do podpunktu b; Tworzymy funkcję find\_max oraz find\_min i korzystając z nich wyszukujemy pozycji wartości maksymalnej oraz minimalnej tak jak robiliśmy to w poprzednim zadaniu.

1. Kod oraz wartość zwrotna programu do podpunktu c; 

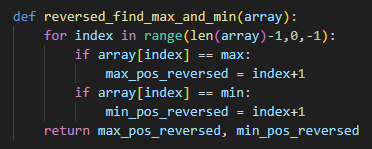
Komentarz do podpunktu c; Tworzymy funkcję, która zliczają poszczególne wartości w tablicy.

1. Kod oraz wartość zwrotna programu do podpunktu d; 

Komentarz do podpunktu d; Tworzymy funkcję, która porównuje indeks jednego i drugiego, a następnie printuje, która jest większa.

1. Kod oraz wartość zwrotna programu do podpunktu e; 

Komentarz do podpunktu e; Tworzymy warunek, w którym sprawdzamy, która z odległości jest większa, a następnie odejmujemy ją od tej mniejszej.

1. Kod programu do podpunktu f; 

Komentarz do podpunktu f; Tworzymy tą samą funkcję co find\_max\_and\_min (co wcześniej), lecz zmieniamy rozpoczynanie pętli od indeksowania tablicy od końca czyli 15 elementu – to samo co len(array)-1.

**Wnioski do zadania #4.**

Program wyszukuje wartości minimalne i maksymalne ze zbioru (tablicy) jednocześnie.