Programming (Cognitive Science), August 18, 2025

~~60 minutes →~~ 90 minutes

A dictionary of stations is given, where the keys are the names of Bicikelj stations and the values are dictionaries with the geographical coordinates of the stops, capacity (number of stands), and the current number of bikes at the station. (See the dictionary in the tests.)

A function razdalje(postaja1, postaja2) is also given, which takes the names of two stations and returns the distance between them.

1. Data Retrieval

Write a function **`prenesi()`** that reads data from the address **`https://ucilnica.fri.uni-lj.si/mod/resource/view.php?id=59478`** (for convenience, the address is stored in the variable URL in the tests), processes it into a dictionary in the same format as the given dictionary postaje, and returns it. (This is a copy of actual data from https://api.jcdecaux.com/vls/v3/stations?apiKey=frifk0jbxfefqqniqez09tw4jvk37wyf823b5j1i&contract=ljubljana, inaccessible during exam.)

2. Alternative Stations

The author of the exam is not a Bicikelj user (once was enough), but knows that at a certain station you sometimes cannot get or return a bike, so it is necessary to look for nearby stations. Write a function **`najblizje3(postaja)`** that, for a given station, finds the three closest stations and returns a list of (three) pairs (distance, name). The list should be sorted by distance. The call **`najblizje3("ŽIVALSKI VRT")`** returns **`[(0.73687236710574, 'VIŠKO POLJE'), (0.9639016462683163, 'TEHNOLOŠKI PARK'), (1.0760703913614853, 'SOSESKA NOVO BRDO')]`**.

3. Station Table

Write a function zapisi(ime_datoteke) that writes into the given file the station names (padded to 40 characters), followed by the number of available bikes and the number of stands. The output must be sorted alphabetically by station names (characters with diacritics will come last; use the default Python sorting) and in the format shown in the box excerpt.

```
KONGRESNI TRG-ŠUBIČEVA ULICA             9/20
KOPALIŠČE ILIRIJA                        0/20
KOPALIŠČE KOLEZIJA                      19/20
KOPRSKA ULICA                            3/8
KOSEŠKI BAJER                           14/20
```

4. Bike Returns

We have a NumPy matrix representing bike transfers between stations over a certain period: the element at index `[i][j]` indicates how many bikes were taken from the i-th station and returned to the j-th station. The matrix in the figure shows that at station 1, 3 bikes were taken and returned to station 1; 7 bikes were taken and returned to station 2; and 5 bikes were returned to station 4.

```
[[1, 2, 1, 0, 3, 0],
 [0, 3, 7, 0, 5, 0],
 [0, 1, 0, 2, 3, 0],
 [0, 3, 0, 0, 4, 2],
 [0, 1, 5, 0, 0, 3],
 [0, 0, 5, 1, 3, 0]]
```

- In total, 15 bikes were taken from station 1, and 10 were returned (where do we see this?). At station 1 there are five bikes fewer than before. Write a function `spremembe(prevozi)` that takes such an array and returns a vector with as many elements as there are stations. Each element indicates how many more (positive numbers) or fewer (negative) bikes are at the station than before. For the matrix in the figure, it returns [-6, -5, 12, -6, 9, -4].
- From station 1 to station 4, 5 bikes were returned, and from station 4 to 1 only 1 bike. The difference is 4. From station 4 to 5, 3 bikes were returned, and vice versa also 3, so the difference is 0. The largest difference is between stations 1 and 2: 1 bike went in one direction 1 and 7 in the other; the difference is 6. Write a function `asimetrija(prevozi)` that returns the pair of stations with the largest difference. For the example above, it should return [1, 2] or [2, 1]. If there are multiple pairs with the largest difference, it may return any of them.

You are supposed to solve the task using clever use of NumPy – in one or two lines, without Python loops.

5. Downhill

By some coincidence, the matrix in the figure also represents the altitudes of certain points on a grid. The cyclist only goes downhill: if they are at cell [1][1] (value 3), they can go up (to 2), left (to 0), or down (to 1), but not right, because there is 7 there. Diagonal moves are not allowed.

Write a function `spust_na_0(matrika, x, y)` that, for the given x (column) and y (row), returns how many zeros are reachable from that cell. If a zero can be reached in multiple ways, count it multiple times, because the cyclist is not smart and does not notice when they reach the same target by a different path.

```
[[1, 2, 1, 0, 3, 0],
 [0, 3, 7, 0, 5, 0],
 [0, 1, 0, 2, 3, 0],
 [0, 3, 0, 0, 4, 2],
 [0, 1, 5, 0, 0, 3],
 [0, 0, 5, 1, 3, 0]]
```

From the mentioned 3, five zeros can be reached: one directly, two via the 1 below, and if going up to the 2, one can then go left or right to the 1 in the first row and then to the zero next to them. The zero in the first row is counted twice because it can be reached in two ways.

Tip: don't worry about “multiple ways”: the task is designed this way to make it easier, not harder. If you tackle the task normally, you will already accidentally count multiple paths multiple times.