

# Python: a faster way to solve non-trivial problems

Yasser Bashir  
PyCon Pakistan 2017

# whoami

Programmer for 20 years

- C++
- Java
- Python
- Javascript

Builder of successful (and failed) products

Founder of companies

(Also amateur squash player)

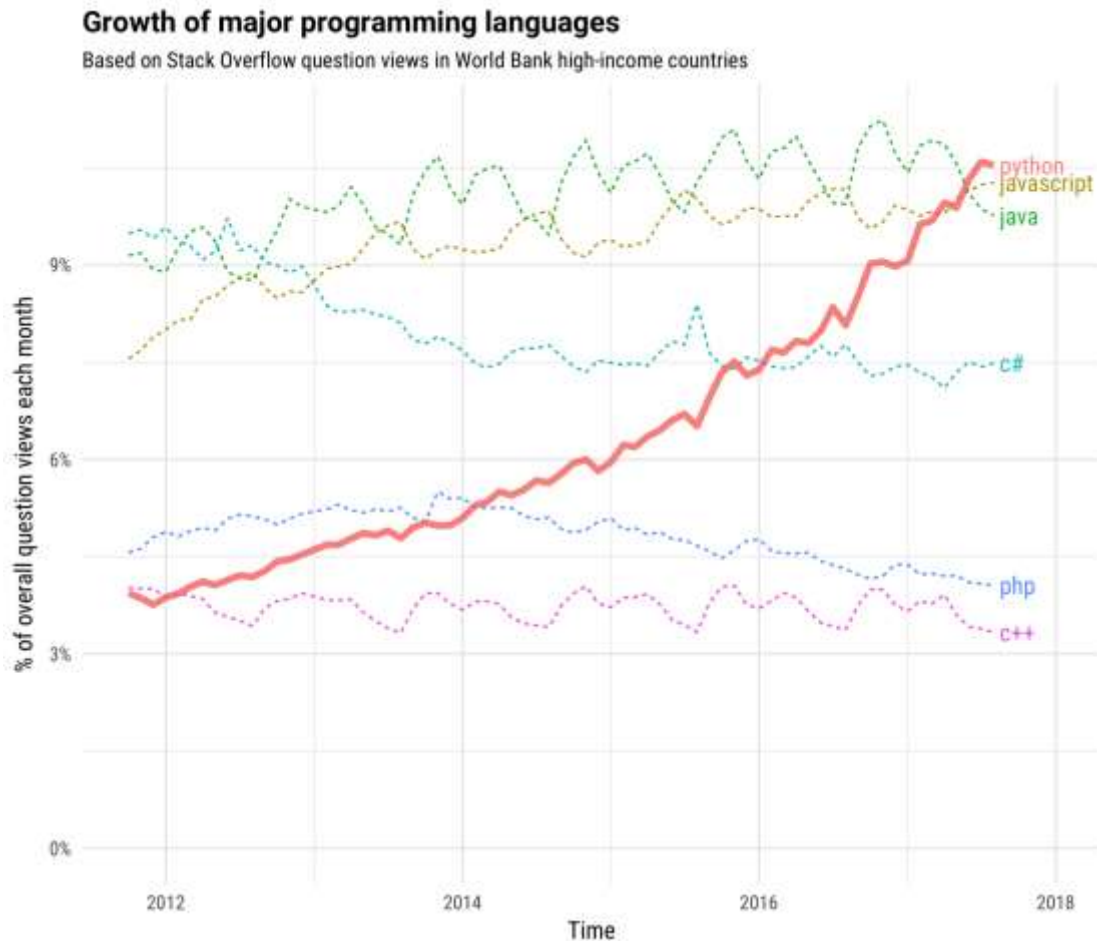
# Why Python?

Language of the fourth industrial wave.

# Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.



# Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.

Because it is practised by contributors in both industrial and scientific communities.

# Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.

Because it is practised by contributors in both industrial and scientific communities.

Because it is easy to understand.

# Why Python?

Language of the fourth industrial wave.

Because it is very popular and its popularity is increasing.

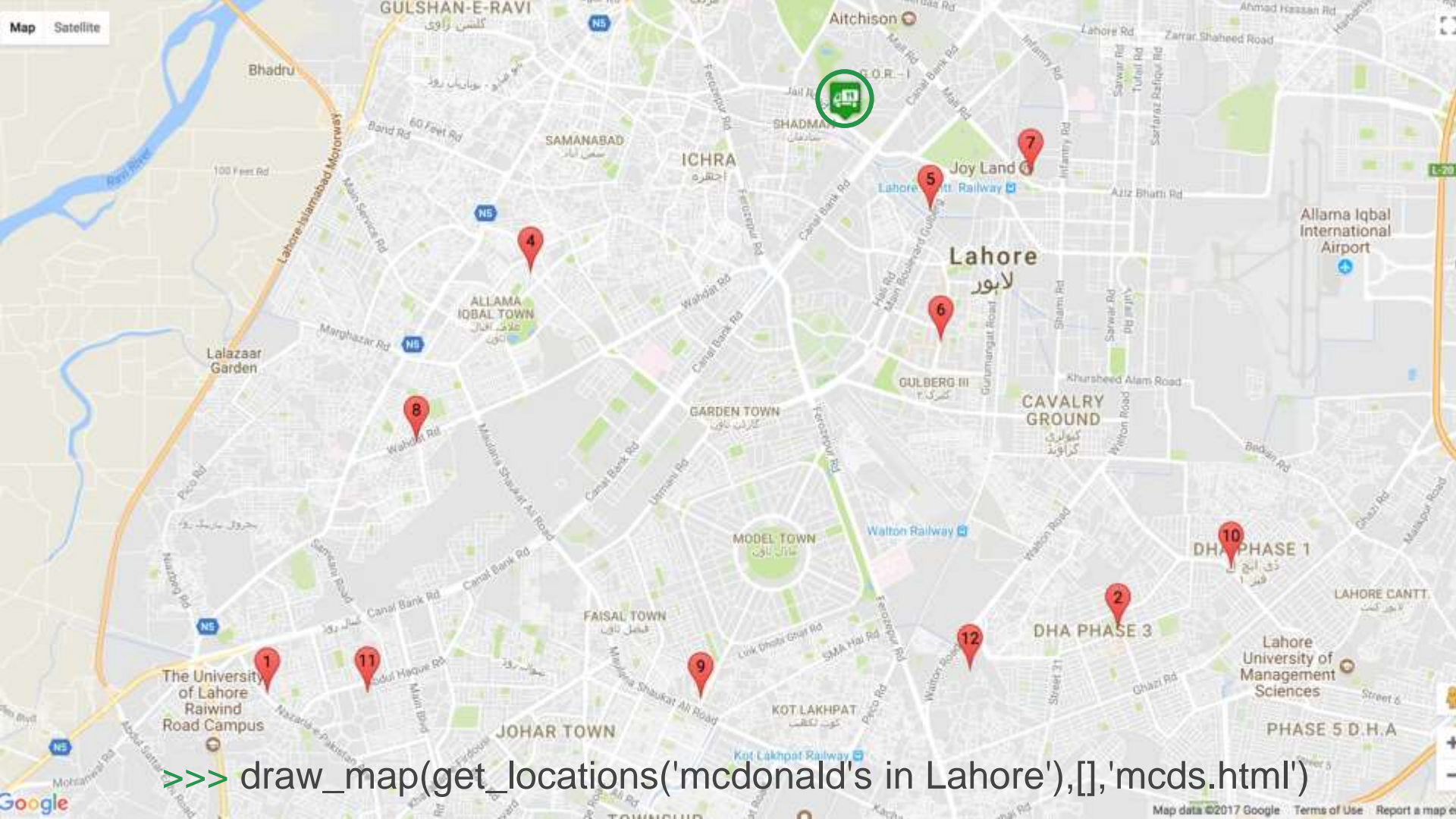
Because it is practised by contributors in both industrial and scientific communities.

Because it is easy to understand.

Because there is very little syntax to learn

Python allows you to solve non-trivial problems faster





>>> draw\_map(get\_locations('mcdonald's in Lahore'),[],'mcds.html')

# The Vehicle Routing Problem (VRP)

How to minimize the time (or distance) required to visit a set of locations given that:

- Vehicles have unlimited capacity

- There may be multiple vehicles

- Vehicles depart from one depot

- Visit each location once only

- Come back to the depot

## Input

*Locations:* [0,1,2,3,4,5,6,7,8,9]

*Vehicles:* 3 *Distances:* [0→1=4, 0→2=5, 2→3=7 etc]

....

Assume 0th location is the depot

Find all partitions of size k for the input set where k is the number of vehicles

Minimize the route for each subset in each partition

Find the partition with the smallest cost

....

## Output

Optimal route: [ [0,7,2] , [0,3,5,9], [0,4,1,8,6] ]

Optimal time: 72.3 minutes



GULSHAN-E-RAVI  
گلشن راوی

Aitchison

Bhadru

SAMANABAD  
سمن آباد

ICHRA  
احقرہ

SHADMAN  
شادمان

Lahore  
لاہور

Allama Iqbal  
International  
Airport

Lalazaar  
Garden

ALLAMA  
IQBAL TOWN  
علامہ اقبال  
ٹاؤن

GARDEN TOWN  
گاردن ٹاؤن

CAVALRY  
GROUNDS  
کراؤنڈ  
گراؤنڈ

MODEL TOWN  
مڈل ٹاؤن

DHA PHASE 1  
ڈی ایچ  
فیز

LAHORE CANTT.  
لاہور کینٹ

The University  
of Lahore  
Raiwind  
Road Campus

FARJAL TOWN  
فجرال ٹاؤن

KOT LAKHPAT  
کوت لاکھپت

DHA PHASE 3

Lahore  
University of  
Management  
Sciences

PHASE 5 D.H.A

# Inputs

Id	Address	Lat	Lon
0	Al-Hussain Road· Lahore, Pakistan	31.4669244	74.25186409999999
1	Lahore, Pakistan	31.3743935	74.17818310000001
2	Sheeba Park, Y Block Market, DHA Phase 3· Lahore, Pakistan	31.47498589999999	74.3770639
3	13 Main Boulevard Allama Iqbal Town· Lahore 54000, Pakistan	31.5197581	74.29066159999999
4	89-B, Jail Rd· Lahore, Pakistan	31.5386987	74.3369092
5	M.M. Alam Road, Block B2· Lahore 54660, Pakistan	31.5111079	74.3509436
6	63 Main Gulberg· Lahore, Pakistan	31.5274972	74.34939969999999
7	Plot #14 and Plot # 15, Hadayatullah Block, Mustafa Town, Wahdat Road· Lahore, Pakistan	31.4984994	74.2737308
8	22-Phase II Govt. Employees Cooperative Housing Society Ltd. Model Town, Link Road· Lahore 54600, Pakistan	31.4662861	74.31567249999999

---

start	end	distance	duration
0	1	17.79	32.80
0	2	17.29	34.83
0	3	9.76	22.00
0	4	13.69	23.82
-	-	-	-

# The Pythonic Solution

**6**

functions

**26**

lines of (non-spaghetti) code using core libraries

**50**

seconds running time

**3**

hours coding time

<https://github.com/ybashir/vrpfun>

```
1 def distance(x, y):
2     return distances[x][y]
3
4 def route_length(route):
5     return sum(distance(route[i], route[i - 1]) for i, v in enumerate(route))
6
7 def all_routes(seq):
8     return [[seq[0]] + list(rest) for rest in itertools.permutations(seq[1:])]
9
10 def all_partitions(collection):
11     if len(collection) == 1:
12         yield [collection]
13         return
14     first = collection[0]
15     for smaller in all_partitions(collection[1:]):
16         for n, subset in enumerate(smaller):
17             yield smaller[:n] + [[first] + subset] + smaller[n + 1:]
18             yield [[first]] + smaller
19
20 def k_partitions_with_shortest_routes(ids, k=3):
21     for p in filter(lambda x: len(x) == k, all_partitions(ids[1:])):
22         yield [min(all_routes([ids[0]] + q), key=route_length) for q in p]
23
24 def shortest_partition(ids, k=3):
25     return min(k_partitions_with_shortest_routes(ids, k),
26               key=lambda x: max(route_length(x[i]) for i in range(k)))
```

Python

```
1 def route_length(route):  
2     return sum(distance(route[i], route[i - 1]) for i, v in enumerate(route))
```

```
>>> route_length([0,3,5,2])  
134.85
```

Java

```
1 private float routeLength(List<Integer> route) {  
2     float sum = 0.0f;  
3     for (int i = 1; i < route.size(); i++) {  
4         sum += getDistance(route.get(i), route.get(i - 1));  
5     }  
6     sum += getDistance(route.get(0), route.get(route.size() - 1));  
7     return sum;  
8 }  
9
```

Kotlin

```
1 fun routeLength(route: List<Int>): Double {  
2     var sum = 0.0  
3     for (i in 1 until route.size)  
4         sum += distance(route[i], route[i - 1])  
5     sum += distance(route[0], route[route.size - 1])  
6     return sum  
7 }  
8
```



Python

```
1 def all_routes(seq):  
2     return [[seq[0]] + list(rest) for rest in itertools.permutations(seq[1:])]
```

```
>>> all_routes([0,3,5,2])  
[[0, 3, 5, 2], [0, 3, 2, 5], [0, 5, 3, 2], [0, 5, 2, 3], [0, 2, 3, 5], [0, 2, 5, 3]]
```

Java

```
1 private List<List<Integer>> allRoutes(List<Integer> seq) {  
2     Integer depot = seq.get(0);  
3     List<List<Integer>> allPossibleRoutes = new ArrayList<>();  
4     Collection<List<Integer>> permutations = Collections2.permutations(seq.subList(1, seq.size()));  
5     for (List<Integer> permutation : permutations) {  
6         List<Integer> route = new ArrayList<>();  
7         route.add(depot);  
8         route.addAll(permutation);  
9         allPossibleRoutes.add(route);  
10    }  
11    return allPossibleRoutes;  
12 }
```

Kotlin

```
1 fun allRoutes(original: ArrayList<Int>): ArrayList<ArrayList<Int>> {  
2     if (original.size < 2) {  
3         return arrayListOf(original)  
4     } else {  
5         val firstElement = original.removeAt(0)  
6         return permutations(original).map {  
7             it.add(0, firstElement)  
8             return@map it  
9         } as ArrayList<ArrayList<Int>>  
10    }  
11 }
```

```

1 def all_partitions(collection):
2     if len(collection) == 1:
3         yield [collection]
4         return
5     first = collection[0]
6     for smaller in all_partitions(collection[1:]):
7         for n, subset in enumerate(smaller):
8             yield smaller[:n] + [[first] + subset] + smaller[n + 1:]
9             yield [[first]] + smaller

```

```

>>> for p in (all_partitions([0,3,5,2])):
...     print(p)

```

```

[[0, 3, 5, 2]]
[[0], [3, 5, 2]]
[[0, 3], [5, 2]]
[[3], [0, 5, 2]]
[[0], [3], [5, 2]]
[[0, 3, 5], [2]]
[[3, 5], [0, 2]]
[[0], [3, 5], [2]]
[[0, 5], [3, 2]]
[[5], [0, 3, 2]]
[[0], [5], [3, 2]]
[[0, 3], [5], [2]]
[[3], [0, 5], [2]]
[[3], [5], [0, 2]]
[[0], [3], [5], [2]]

```

<https://stackoverflow.com/questions/19368375/set-partitions-in-python>

```

1 public static List<List<List<Integer>>> getAllPartitions(List<Integer> inputList) throws Exception {
2     int[] array = inputList.stream().mapToInt(i -> i).toArray();
3     int[][][] partitionArrays = getAllPartitions(array);
4     List<List<List<Integer>>> partitionsList = new ArrayList<>();
5     for (int[][] partition : partitionArrays) {
6         List<List<Integer>> partitionSets = new ArrayList<>();
7         for (int[] set : partition) {
8             List<Integer> singleSet = Ints.asList(set);
9             partitionSets.add(singleSet);
10        }
11        partitionsList.add(partitionSets);
12    }
13    return partitionsList;
14}
15
16 private static int[][][] getAllPartitions(int[] array) throws Exception {
17     int[][][] res = new int[0][][];
18     int n = 1;
19     for (int i = 0; i < array.length; i++) {
20         n = 2;
21     }
22     for (int i = 1; i < n; i += 2) {
23         boolean[] contains = new boolean[array.length];
24         int length = 0;
25         int k = 1;
26         for (int j = 0; j < array.length; j++) {
27             contains[j] = k % 2 == 1;
28             length += k % 2;
29             k /= 2;
30         }
31         int[] firstPart = new int[length];
32         int[] secondPart = new int[array.length - length];
33         int p = 0;
34         int q = 0;
35         for (int j = 0; j < array.length; j++) {
36             if (contains[j]) {
37                 firstPart[p++] = array[j];
38             } else {
39                 secondPart[q++] = array[j];
40             }
41         }
42         int[][][] partitions;
43         if (length == array.length) {
44             partitions = new int[1][][1] {{firstPart}};
45         } else {
46             partitions = getAllPartitions(secondPart);
47             for (int j = 0; j < partitions.length; j++) {
48                 int[] partition = new int[partitions[j].length + 1];
49                 partition[0] = firstPart;
50                 System.arraycopy(partitions[j], 0, partition, 1, partitions[j].length);
51                 partitions[j] = partition;
52             }
53         }
54         int[][][] newRes = new int[res.length + partitions.length][][];
55         System.arraycopy(res, 0, newRes, 0, res.length);
56         System.arraycopy(partitions, 0, newRes, res.length, partitions.length);
57         res = newRes;
58     }
59     return res;
60 }

```

```
20 def k_partitions_with_shortest_routes(ids, k=3):
21     for p in filter(lambda x: len(x) == k, all_partitions(ids[1:])):
22         yield [min(all_routes([ids[0]] + q), key=route_length) for q in p]
23
```

```
>>> for p in k_partitions_with_shortest_routes([0,3,5,2],k=2):
...     print(p)
...
[[0, 3], [0, 5, 2]]
[[0, 3, 5], [0, 2]]
[[0, 5], [0, 3, 2]]
```

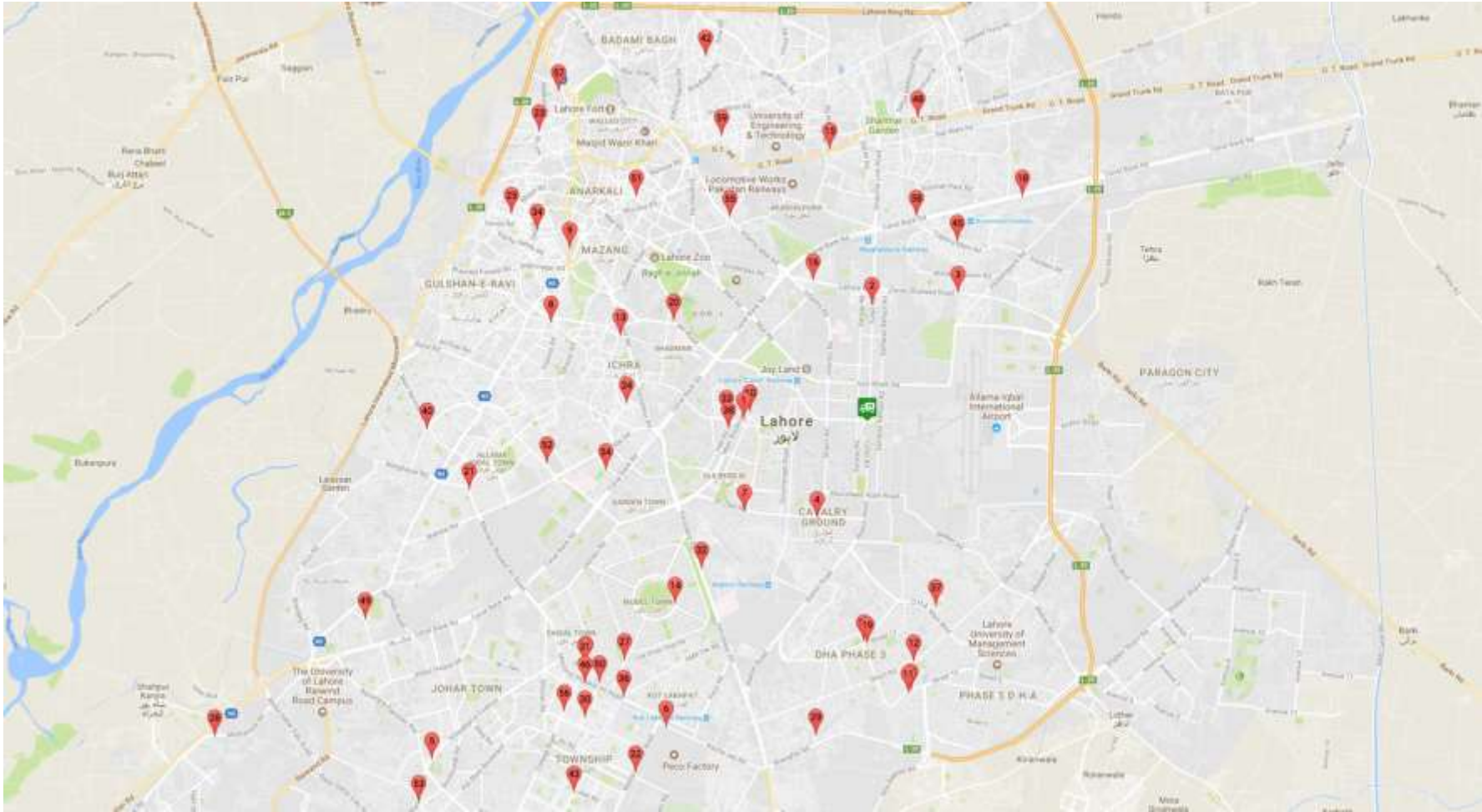
```
23
24 def shortest_partition(ids, k=3):
25     return min(k_partitions_with_shortest_routes(ids, k),
26               key=lambda x: max(route_length(x[i]) for i in range(k)))
```

```
>>> shortest_partition([0,3,5,2],k=2)
[[0, 3], [0, 5, 2]]
```

```
1 def distance(x, y):
2     return distances[x][y]
3
4 def route_length(route):
5     return sum(distance(route[i], route[i - 1]) for i, v in enumerate(route))
6
7 def all_routes(seq):
8     return [[seq[0]] + list(rest) for rest in itertools.permutations(seq[1:])]
9
10 def all_partitions(collection):
11     if len(collection) == 1:
12         yield [collection]
13         return
14     first = collection[0]
15     for smaller in all_partitions(collection[1:]):
16         for n, subset in enumerate(smaller):
17             yield smaller[:n] + [[first] + subset] + smaller[n + 1:]
18             yield [[first]] + smaller
19
20 def k_partitions_with_shortest_routes(ids, k=3):
21     for p in filter(lambda x: len(x) == k, all_partitions(ids[1:])):
22         yield [min(all_routes([ids[0]] + q), key=route_length) for q in p]
23
24 def shortest_partition(ids, k=3):
25     return min(k_partitions_with_shortest_routes(ids, k),
26               key=lambda x: max(route_length(x[i]) for i in range(k)))
```

	Python (PyPy)	Java	Kotlin
<b>Execution time</b>	50 secs	25 secs	19 secs
<b>Coding time</b>	3 hours	12 hours	8 hours
<b>Lines of code</b>	30	150+	100
<b>Memory footprint</b>	136M	346M	887 MB

>>> draw\_map(get\_locations('gourmets in lahore'),[],'gourmet.html')



# Thank you

Special thanks to:

Aitzaz Mumtaz Khan (for contributing java code)

[https://github.com/aitzaz/vehicle\\_routing\\_problem](https://github.com/aitzaz/vehicle_routing_problem)

Shahroz Khan (for contributing Kotlin code)

<https://github.com/Shahroz16/VRP>