FOLDERS ARCHITECTURE

FILES EXPLANATION

SCRIPT.PY VARIABLES DESCRIPTION

VAN GRAB DROP ALGORITHM BREAKDOWN

VAN GRAB DROP ALGORITHM EXAMPLE

# FOLDERS ARCHITECTURE

○ data

| • **all\_tables.xlsx**

| | - hubs tab

| | | - hub\_id

| | | - hub\_name

| | | - hub\_latitude

| | | - hub\_longtitude

| | | - hub\_cap

| | - garms tab

| | | - garm\_id

| | | - garm\_name

| | | - hub\_id

| |

| |

| • **orders.json**

| • **route.json**

|

|

○ action

| • **print.py**

| • **script.py**

|

|

○ help

| • **readme.docx**

| • **script\_comments.txt**

| • **van\_algorithm\_example.jpg**

|

|

• **print**

• **STATIONTOSTATION.txt**

# FILES EXPLANATION

**orders.json** (necessary data file)

* ITEM\_HUB\_ORDER
* itemX must be at hubY
* “628”: 4

“all orders for single day”

Manually update. How?

* When order arrives see customer address and item
* Find closest laundrette to customer address
* Update the order.json file with “item\_id”: hub\_id
* Save file

**route.json** (necessary data file)

{1,2,3,2,1}

MANUALLY WRITE

ROUTE HUB TO HUB THAT THE VAN WILL FOLLOW EVERY DAY

DOES NOT CHANGE

**all\_tables.xlsx** (necessary data file)

LAUNDRETTE HUBS

LAUNDRETTE LATITUDE

LAUNDRETTE LONGTITUDE

LAUNDRETTE CAPACITY

CLOTHES (GARMS) IDs

CLOTHES (GARMS) NAME

CLOTHES (GARMS) CURRENT HUB ID

**script.py**

ALL CODE IN THIS ONE FILE

**route.json** (output file)

# SCRIPT.PY VARIABLES DESCRIPTION

**all\_hubs** dictionary

hub1 : [ "hub1\_name", capacity34, [2,4,19,35] ]

hub2 : [ "hub2\_name", capacity20, [1,3,10,16] ]

read all\_tables.xlsx > hubs/garms tables.s

**item\_hub\_order** dictionary

item1 : hub3

item2 : hub5

item3 : hub1

item6 : hub2

item "must be there"

item1 ordered by customer near hub3

> daily orders.

**item\_hub\_move** dictionary

item1 : hub3

item3 : hub1

item "must go there"

**all the items i will inner move**

**and their future hubs.**

> subset of item\_hub\_outer.

> item\_hub\_outer minus items

that are needed at hubs but

are already there.

**item\_hub\_stock** dictionary

item1 : hub2

item3 : hub2

> same length as item\_hub\_inner.

> **all the items i will inner move**

**and their current hubs**

**item\_hub\_hub** dictionary

item1 : [hub3, hub2]

item3 : [hub1, hub2]

> item\_hub\_inner + item\_hub\_stock

**hub\_route** array

[1,2,3,4,3,2,1]

**hub\_take\_leave** dictionary

hub1 : [ [item1, item4, item10], [item2, item3, item20] ]

hub2 : [ [item9], [item11, item13] ]

> at hub grab items and drop items

**station\_take\_leave** dictionary

hub1, #1 : [ [item1, item4, item10], ]

hub2, #1 : [ [item9], ]

...

hub2, #2 : [ [item11, item13] ]

hub1, #1 : [ [item2, item3, item20] ]

at hub1, pass #1

at hub2, pass #2

# VAN GRAB DROP ALGORITHM BREAKDOWN

i have **route** list

[1,2,3,4,3,2,1]

i have **hub\_take\_leave** dictionary

1 : [ [2,4,5,7], [1,13,20] ]

2 : [ [9,10], [5,7,19] ]

...

4 : [ ... ]

i make **station\_take\_leave** dictionary

kypseli take 2 3 4 7 leave null

glyfada take 9 10 leave null

...

glyfada take null leave 1 13 20

kypseli take null leave 5 7 19

1) van starts empty

2) each hub has

clothes that must be grabbed

clothes that must be dropped

3) for [1,2,3,4] everytime van passes

grab all clothes needed

update van

it checks if what they need is on van. if yes then drop (leave) it

update van

4) for [3,2,1] everytime van passes

(grab nothing)

check if what they need is on van. if yes drop (leave) it

update van

5) at the end

the van should be empty

the hubs should have what they need

# VAN GRAB DROP ALGORITHM EXAMPLE