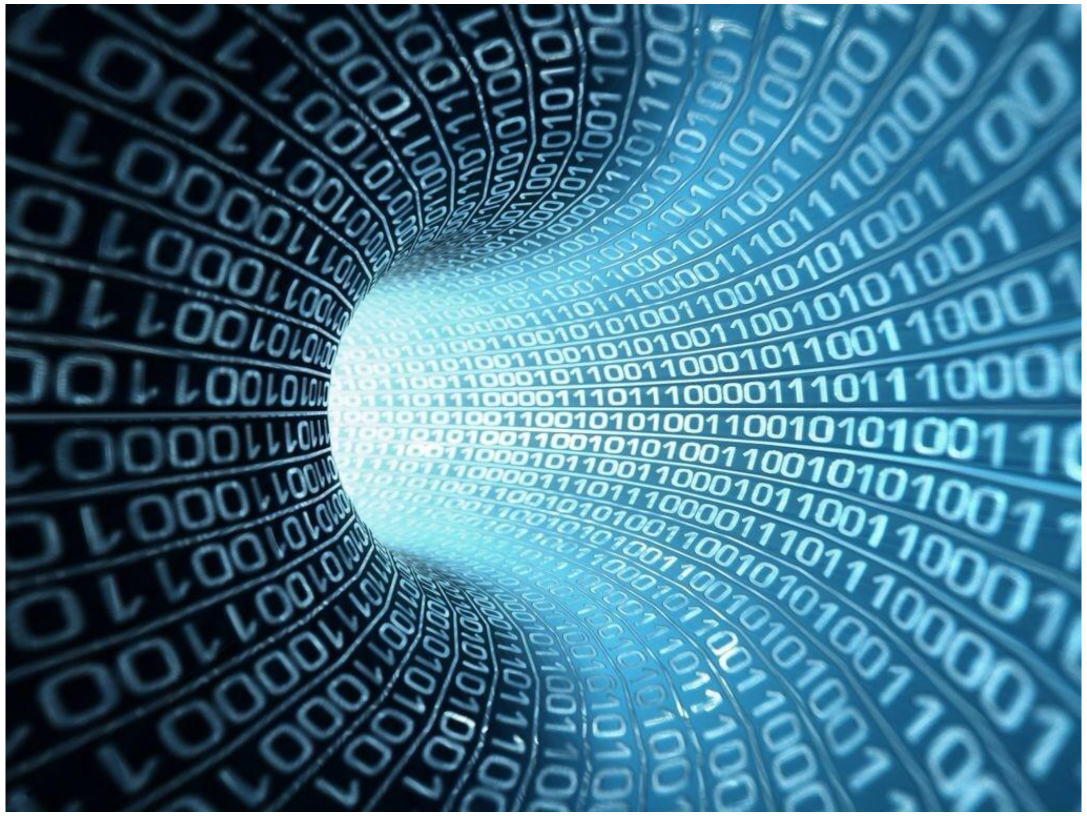
FIT1045: Algorithms and Programming Fundamentals in Python

Lecture 6

Data

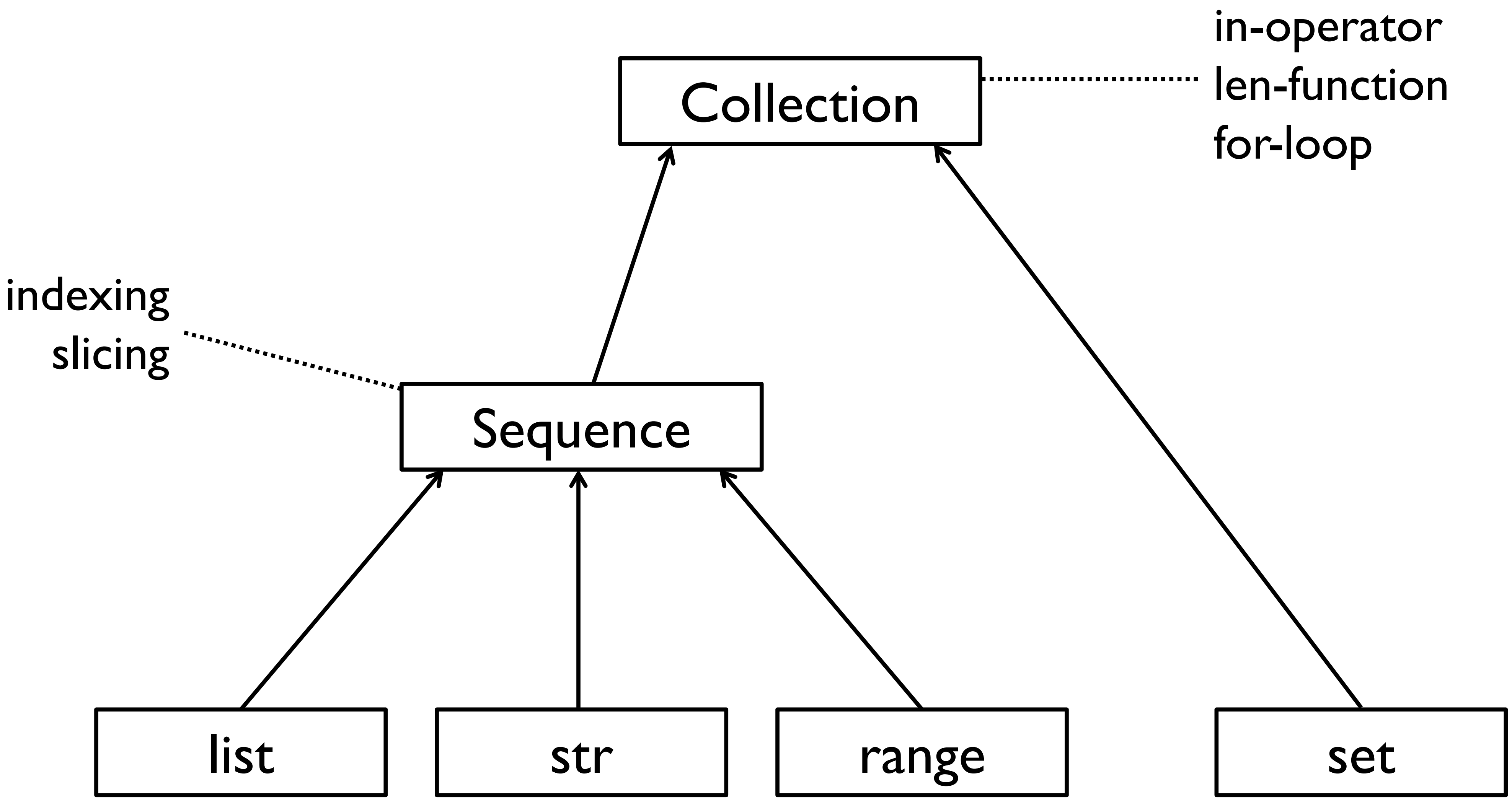


https://phys.org/news/2013-05-big-datafor-worse.html

Additional PASS Session

- PASS sessions frequented a lot
- high-level of discussions
- add an additional PASS session
- find best timeslot using google forms:
  - https://forms.gle/zNJ1ESYFgDgT94D66

Recap: Collections type hierarchy



Recap: for-loops and ranges

```
def have_common_element(c1, c2):
    for a in c1:
        for b in c2:
            if a==b:
                return True
    return False
```

```
def quantity_eaten(food, eaten_foods, eaten_quantities):
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

Goal this week: use Python to track macro-nutrients

Input:

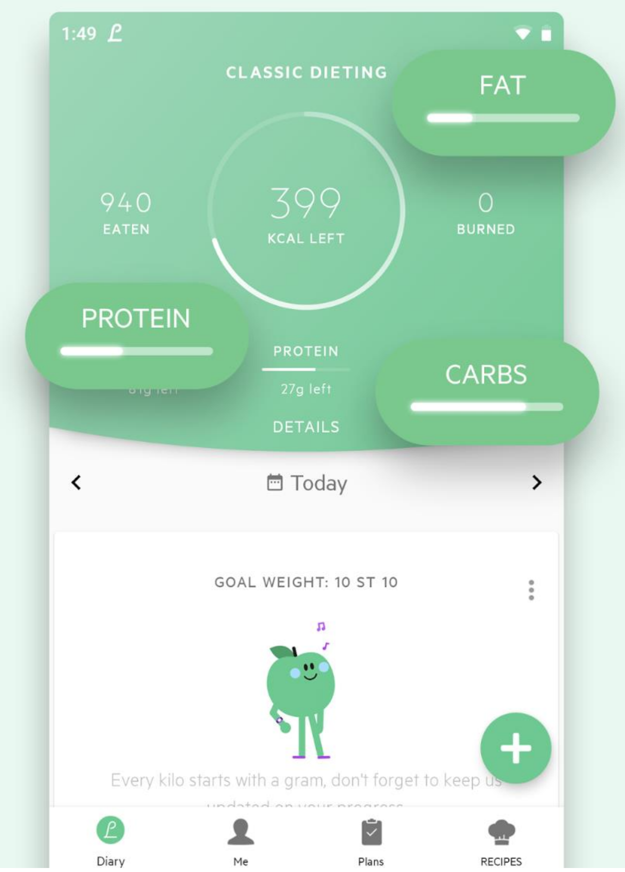
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
10	3	carrot	120
11	3	eggplant	150
12	3	coconut cream	160
13	3	apple	110

food diary

food	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12	11.8	0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
broccoli	124	89.6	3.2	2	2	0.1	4.1
beef	613	70	22.8	0.2	0	6	0
lamb	1057	60.2	18.6	0	0	20.2	0
bread	1446	37.6	8.4	43.5	1.5	2.6	6.9
potato	346	77.4	2	17	0	0.1	2.5
tofu	510	74	12	1.5	0.5	6.5	5
tomato	81	93.3	1	2.9	0.9	0.2	1
eggplant	107	91.6	1.2	3.5	1.5	0.2	2.5
carrot	116	90.6	0.8	4.7	4.4	0	2.9
coco. cream	872	73	1.5	3	0	21.5	0
rice	403	75.3	2.5	20	0	0.4	0.8

database of nutrition values

Output:



Every kilo starts with a gram, don't forget to keep us updated on your progress

https://play.google.com/store/apps/details?id=com.sillens.shapeupclub

nutritional intake per day

Goal this week: use Python to track macro-nutrients

Input:

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
10	3	carrot	120
11	3	eggplant	150
12	3	coconut cream	160
13	3	apple	110

food diary

food	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12	11.8	0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
broccoli	124	89.6	3.2	2	2	0.1	4.1
beef	613	70	22.8	0.2	0	6	0
lamb	1057	60.2	18.6	0	0	20.2	0
bread	1446	37.6	8.4	43.5	1.5	2.6	6.9
potato	346	77.4	2	17	0	0.1	2.5
tofu	510	74	12	1.5	0.5	6.5	5
tomato	81	93.3	1	2.9	0.9	0.2	1
eggplant	107	91.6	1.2	3.5	1.5	0.2	2.5
carrot	116	90.6	0.8	4.7	4.4	0	2.9
coco. cream	872	73	1.5	3	0	21.5	0
rice	403	75.3	2.5	20	0	0.4	0.8

database of nutrition values

Output:

day	energy	water	protein	carbs	sugars	fat	fibres
1	3354	705.7	81.2	67.6	15.8	18.5	18
2	1868	553.2	21.8	62.1	14.2	8.45	16.55
3	2833.4	621.31	11.1	72.89	20.51	35.58	11.52

nutritional intake per day

Objectives

Being able to read in, process, and write out data

- representing structured input data as tables
- *update* and transform data to solve problems
- reading data from and writing data to files

Learning outcomes

- 1 (translate between problem descriptions and program design with appropriate input/output representations)
- 2 (choose and implement appropriate problem solving strategies in Python)

Concrete goal: nutrition app

Where am I?

1. Reading from files
2. Tables and Multiple Assignments
3. Computing nutritional intake

How to read content of a file to list?

foods.txt

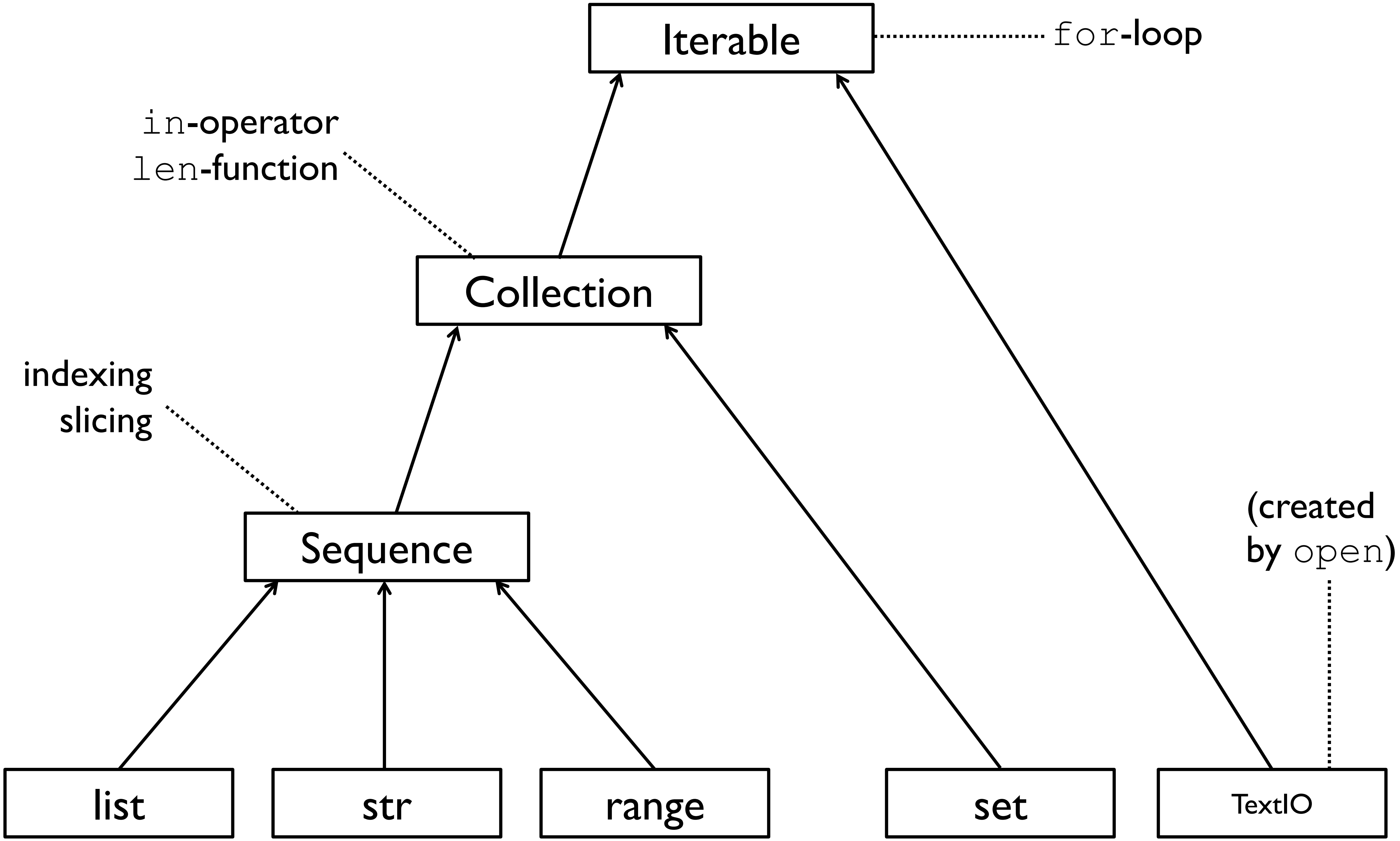
```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

- Built-in function *open* provides object that represents file
- Object can be used in for-loop to iterate over content (of text file)
- Each element corresponds to one line in file (terminated by new line character '\n')

```
>>> open('foods.txt')
<_io.TextIOWrapper name='foods.txt' mode='r' encoding='UTF-8'>
>>> file = open('foods.txt')
>>> type(file)
<class '_io.TextIOWrapper'>
>>> content = []
>>> for line in file:
...     content = content + [line]
...
>>> content
['apple\n', 'orange\n', 'broccoli\n', 'beef\n', 'lamb\n',
'bread\n', 'potato\n', 'tofu\n', 'tomato']
```

special character symbolising "new line"

Iterable type hierarchy



Attributes: named parts of an object

```
>>> file = open('foods.txt')
>>> file.readline()
'beef\n'
>>> file.name
'foods.txt'
>>> file.closed
False
>>> file.close()
>>> file.closed
True
```

- Objects can have named "parts" (other objects)
- Accessed via the dot-notation (just like functions in modules)
- If function with name f is part of object x then f is called "a method of x"
- Other things are usually referred to as "attributes"

This is not special to file-like objects

```
>>> x = 0.5
>>> x.is_integer()
False
>>> x.is_integer
<built-in method is_integer of float object at
0x10f45d870>
>>> x.as_integer_ratio()
(1, 2)
>>> y = 8
>>> y.bit_length()
4
```

Some useful string methods

```
>>> ' line with whitespace \n'.strip()
'line with whitespace'
>>> '_and_'.join(['dogs','cats','horses'])
'dogs_and_cats_and_horses'
>>> 'I now know strings!'.split()
['I', 'now', 'know', 'strings!']
>>> 'dogs,cats,horses'.split(',')
['dogs','cats','horses']
```

Let’s define a reusable function

foods.txt

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

```
def list_from_file(filename):
    file = open(filename)
    res = []
    for line in file:
        res = res + [line.strip()]
    file.close()
    return res
```

```
>>> foods = list_from_file('foods.txt')
>>> foods
['apple', 'broccoli', 'beef', 'lamb', 'bread', 'potato', 'tofu',
'tomato’]
>>>
```

Now we can apply our quantity computation to user data…

foods.txt

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

quantities.txt

```
300
300
200
100
250
100
120
200
```

```
def quantity_eaten(food, foods, quant):
    res = 0
    for i in range(len(foods)):
        if foods[i] == food:
            res = res + quant[i]
    return res
```

```
>>> foods = list_from_file('foods.txt')
>>> foods
['apple', 'broccoli', 'beef', 'lamb', 'bread', 'potato', 'tofu',
'tomato’]
>>> quantities = list_from_file('quantities.txt')
>>> quantity_eaten('apple', foods, quantities)
?
```

https://flux.qa

Clayton: AXXULH
Malaysia: LWERDE

…or not yet

foods.txt

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

quantities.txt

```
300
300
200
100
250
100
120
200
```

```
def quantity_eaten(food, foods, quant):
    res = 0
    for i in range(len(foods)):
        if foods[i] == food:
            res = res + quant[i]
    return res
```

```
>>> foods = list_from_file('foods.txt')
>>> foods
['apple', 'broccoli', 'beef', 'lamb', 'bread', 'potato', 'tofu',
'tomato’]
>>> quantities = list_from_file('quantities.txt')
>>> quantity_eaten('apple', foods, quantities)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/mbol0005/Google Drive Monash/FIT1045/FIT1045-S1-
2020/Lectures/Lecture05/lecture5.py", line 63, in quantity_eaten
    res = res + quant [i]
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Our function creates only string lists

foods.txt

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

quantities.txt

```
300
300
200
100
250
100
120
200
```

```
def quantity_eaten(food, foods, quant):
    res = 0
    for i in range(len(foods)):
        if foods[i] == food:
            res = res + quant[i]
    return res
```

```
>>> foods = list_from_file('foods.txt')
>>> foods
['apple', 'broccoli', 'beef', 'lamb', 'bread', 'potato', 'tofu',
'tomato’]
>>> quantities = list_from_file('quantities.txt')
>>> quantities
['300', '300', '200', '100', '250', '100', '120', '200']
```

Add numeric type conversion

foods.txt

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

quantities.txt

```
300
300
200
100
250
100
120
200
```

```
def list_from_file(fname, num=False):
    file = open(fname)
    rs = []
    for l in file:
        if num:
            rs = rs+[float(l.strip())]
        else:
            rs = rs+[l.strip()]
    file.close()
    return rs
```

```
>>> foods = list_from_file('foods.txt')
>>> foods
['apple', 'broccoli', 'beef', 'lamb', 'bread', 'potato', 'tofu',
'tomato’]
>>> quantities = list_from_file('quantities.txt’, True)
>>> quantities
[300.0, 300.0, 200.0, 100.0, 250.0, 100.0, 120.0, 200.0]
>>>
```

We’ll see nicer way to do this in later lecture

This works for now

foods.txt

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

quantities.txt

```
300
300
200
100
250
100
120
200
```

```
def quantity_eaten(food, foods, quant):
    res = 0
    for i in range(len(foods)):
        if foods[i] == food:
            res = res + quant[i]
    return res
```

```
>>> foods = list_from_file('foods.txt')
>>> foods
['apple', 'broccoli', 'beef', 'lamb', 'bread', 'potato', 'tofu',
'tomato’]
>>> quantities = list_from_file('quantities.txt’, True)
>>> quantities
[300.0, 300.0, 200.0, 100.0, 250.0, 100.0, 120.0, 200.0]
>>> quantity_eaten('apple', foods, quantities)
200.0
```

Where am I?

1. Reading from files
2. Tables and Multiple Assignments
3. Computing nutritional intake

Tables

Two-dimensional structured information (e.g., nutrition table)

	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12.0	11.8	0.0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
broccoli	124	89.6	3.2	2.0	2.0	0.1	4.1
beef	613	70	22.8	0.2	0.0	6.0	0.0
lamb	1057	60.2	18.6	0.0	0.0	20.2	0.0
bread	1446	37.6	8.4	43.5	1.5	2.6	6.9

How to represent in Python?

```
cols = ['energy', …,'carbs', 'sugars', 'fat', 'fibres’]
ids = ['apple', …,'beef', 'lamb', 'bread’]
```

Tables

Two-dimensional structured information (e.g., nutrition table)

	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12.0	11.8	0.0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
broccoli	124	89.6	3.2	2.0	2.0	0.1	4.1
beef	613	70	22.8	0.2	0.0	6.0	0.0
lamb	1057	60.2	18.6	0.0	0.0	20.2	0.0
bread	1446	37.6	8.4	43.5	1.5	2.6	6.9

How to represent in Python?

```
cols = ['energy', …,'carbs', 'sugars', 'fat', 'fibres’]
ids = ['apple', …,'beef', 'lamb', 'bread']

nutr_vals = [[229, 84.3, 0.4, 12.0, 11.8, 0.0, 2.3],
             [186, 84.3, 1, 9.5, 8.3, 0.2, 2.1],
                                …
             [1446, 37.6, 8.4, 43.5, 1.5, 2.6, 6.9]]
```

https://flux.qa

Clayton: AXXULH
Malaysia: LWERDE

Tables

Two-dimensional structured information (e.g., nutrition table)

	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12.0	11.8	0.0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
broccoli	124	89.6	3.2	2.0	2.0	0.1	4.1
beef	613	70	22.8	0.2	0.0	6.0	0.0
lamb	1057	60.2	18.6	0.0	0.0	20.2	0.0
bread	1446	37.6	8.4	43.5	1.5	2.6	6.9

How to represent in Python?

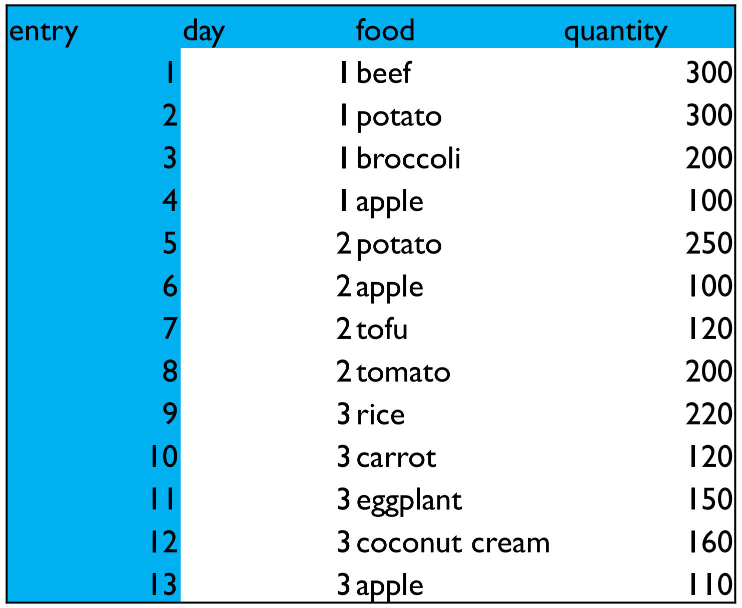
```
cols = ['energy', …,'carbs', 'sugars', 'fat', 'fibres’]
ids = ['apple', …,'beef', 'lamb', 'bread']

nutr_vals = [[229, 84.3, 0.4, 12.0, 11.8, 0.0, 2.3],
             [186, 84.3, 1, 9.5, 8.3, 0.2, 2.1],
                               …
             [1446, 37.6, 8.4, 43.5, 1.5, 2.6, 6.9]]

fat_broccoli = nutr_vals[2][5]  #value: 0.1
```

Representation as table can simplify code in some cases

```
def quantity_eaten(food, eaten_foods, eaten_quantities):
    """Input : specific food, list of eaten foods,
               list of eaten quantities
       Output: total quantity of specific food eaten"""
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

```
beef
potato
broccoli
apple
potato
apple
tofu
tomato
```

```
300
300
200
100
250
100
120
200
```



entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
10	3	carrot	120
11	3	eggplant	150
12	3	coconut cream	160
13	3	apple	110

Representation as table can simplify code in some cases

```
def quantity_eaten(food, eaten_foods, eaten_quantities):
    """Input : specific food, list of eaten foods,
               list of eaten quantities
      Output: total quantity of specific food eaten"""
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

```
def quantity_eaten(food, food_diary):
    """Input : specific food, food diary table with
               1st col food eaten, 2nd col quantity
      Output: total quantity of specific food eaten"""
    res = 0
    for row in food_diary:
        f = row[0]
        q = row[1]
        if f == food:
            res = res + q
    return res
```

instead of data series indices, now column indices

Representation as table can simplify code in some cases

```
def quantity_eaten(food, eaten_foods, eaten_quantities):
    """Input : specific food, list of eaten foods,
               list of eaten quantities
       Output: total quantity of specific food eaten"""
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

```
def quantity_eaten(food, food_diary):
    """Input : specific food, food diary table with
               1st col food eaten, 2nd col quantity
       Output: total quantity of specific food eaten"""
    res = 0
    for row in food_diary:
        f, q = row[0], row[1]

        if f == food:
            res = res + q
    return res
```

multiple assignment statement

Representation as table can simplify code in some cases

```
def quantity_eaten(food, eaten_foods, eaten_quantities):
    """Input : specific food, list of eaten foods,
               list of eaten quantities
       Output: total quantity of specific food eaten"""
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

```
def quantity_eaten(food, food_diary):
    """Input : specific food, food diary table with
               1st col food eaten, 2nd col quantity
       Output: total quantity of specific food eaten"""
    res = 0
    for row in food_diary:
        f, q = row

        if f == food:
            res = res + q
    return res
```

multiple assignment with sequence "unpacking"

Representation as table can simplify code in some cases

```python
def quantity_eaten(food, eaten_foods, eaten_quantities):
    """Input : specific food, list of eaten foods,
               list of eaten quantities
       Output: total quantity of specific food eaten"""
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

```python
def quantity_eaten(food, food_diary):
    """Input : specific food, food diary table with
               1st col food eaten, 2nd col quantity
       Output: total quantity of specific food eaten"""
    res = 0
    for f, q in food_diary:


        if f == food:
            res = res + q
    return res
```

multiple assignment
directly in for loop
assignment

Representation as table can simplify code in some cases

```
def quantity_eaten(food, eaten_foods, eaten_quantities):
    """Input : specific food, list of eaten foods,
               list of eaten quantities
       Output: total quantity of specific food eaten"""
    res = 0
    for i in range(len(eaten_foods)):
        if eaten_foods[i] == food:
            res = res + eaten_quantities[i]
    return res
```

```
def quantity_eaten(food, food_diary):
    """Input : specific food, food diary table with
               1st col food eaten, 2nd col quantity
       Output: total quantity of specific food eaten"""
    res = 0
    for f, q in food_diary:
        if f == food:
            res = res + q
    return res
```

No indices, clearer representation

Multiple assignment statement

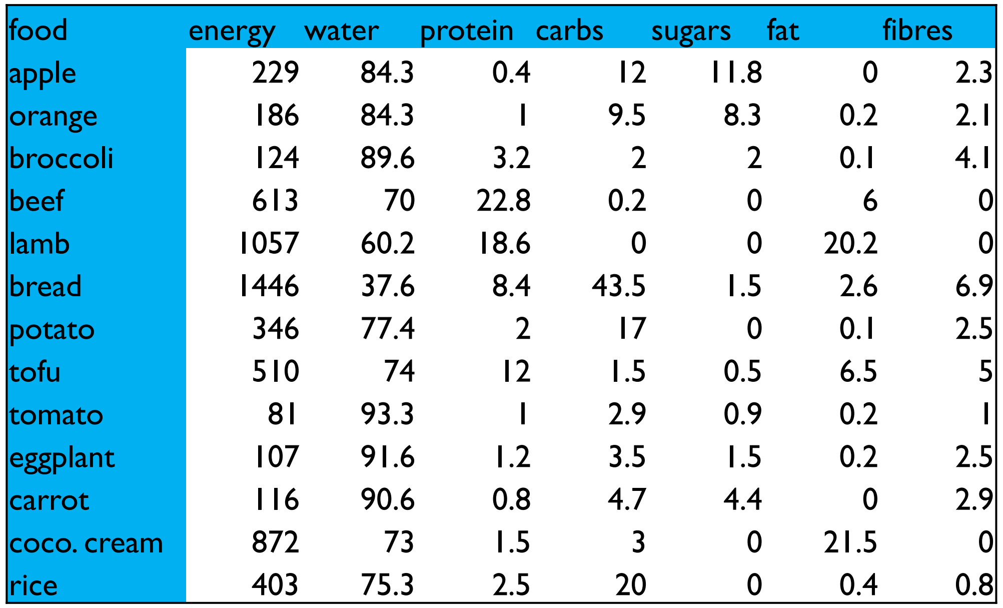
```
>>> x, y = 10, 21
>>> x
10
>>> y
21
>>> x, y = [10, 21]
>>> x, y, z = 'abc'
>>> x
'a'
>>> z
'c'
>>> x, y = 'abc'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)
```

number of variable
names (left) has to
match length of
sequence (right)

Reading table with type conversion

nutr_tab.csv

```
food,energy,water,protein,carbs,…
apple,229,84.3,0.4,12,11.8,0,2.3
orange,186,84.3,1,9.5,8.3,0.2,2.1
broccoli,124,89.6,3.2,2,2,0.1,4.1
beef,613,70,22.8,0.2,0,6,0
lamb,1057,60.2,18.6,0,0,20.2,0
bread,1446,37.6,8.4,43.5,1.5,2.6,…
potato,346,77.4,2,17,0,0.1,2.5
…
```



food	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12	11.8	0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
broccoli	124	89.6	3.2	2	2	0.1	4.1
beef	613	70	22.8	0.2	0	6	0
lamb	1057	60.2	18.6	0	0	20.2	0
bread	1446	37.6	8.4	43.5	1.5	2.6	6.9
potato	346	77.4	2	17	0	0.1	2.5
tofu	510	74	12	1.5	0.5	6.5	5
tomato	81	93.3	1	2.9	0.9	0.2	1
eggplant	107	91.6	1.2	3.5	1.5	0.2	2.5
carrot	116	90.6	0.8	4.7	4.4	0	2.9
coco. cream	872	73	1.5	3	0	21.5	0
rice	403	75.3	2.5	20	0	0.4	0.8

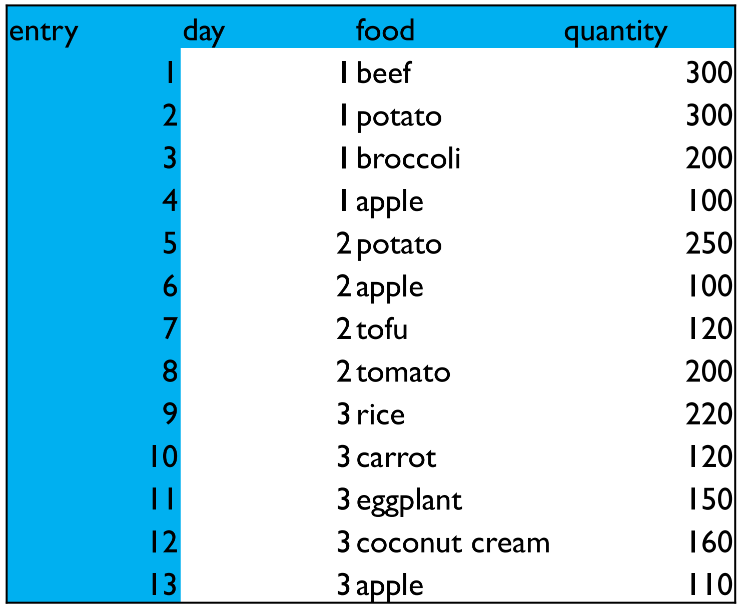
```python
def table_from_file(filename, num_cols=[]):
    lines = list_from_file(filename)
    cols = lines[0].split(',')[1:]
    ids, tab = [], []
    for i in range(1, len(lines)):
        entries = lines[i].split(',')
        ids = ids + [entries[0]]
        row = as_float(num_cols, entries[1:])
        tab = tab + [row]
    return tab, cols, ids

nvals, nutr_cols, foods = table_from_file('nutr_tab.csv', range(7))
```

multiple return values picked up in multiple assignment statement

Reading table with type conversion

food_diary.csv

```
id,day,food,quantity
1,1,beef,300
2,1,potato,300
3,1,broccoli,200
4,1,apple,100
5,2,potato,250
6,2,apple,100
7,2,tofu,120
…
```



entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
10	3	carrot	120
11	3	eggplant	150
12	3	coconut cream	160
13	3	apple	110

```python
nvals, nutr_cols, foods = table_from_file('nutr_tab.csv', range(7))
food_diary, _, _ = table_from_file('food_diary.csv', [3])
```

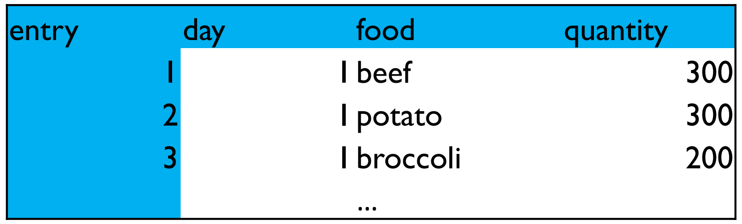
underscore as variable name to indicate “don’t care” for value

```
>>> food_diary
[['1', 'beef', 300.0], ['1', 'potato', 300.0], ['1', 'broccoli',
200.0], ['1', 'apple', 100.0], ['2', 'potato', 250.0], ['2',
'apple', 100.0], ['2', 'tofu', 120.0], ['2', 'tomato', 200.0],
['3', 'rice', 220.0], ['3', 'carrot', 120.0], ['3', 'eggplant',
150.0], ['3', 'coconut cream', 160.0], ['3', 'apple', 110.0]]
```

Where am I?

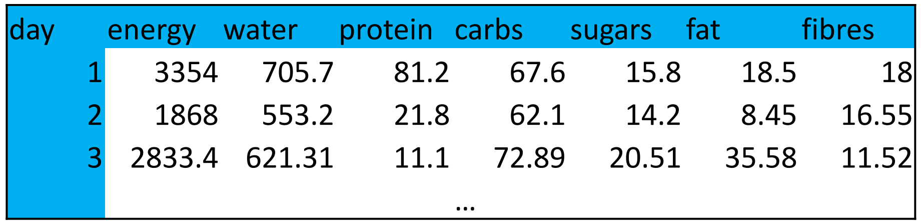
1. Reading from files
2. Tables and Multiple Assignments
3. Computing nutritional intake

Computing nutrient intake per day



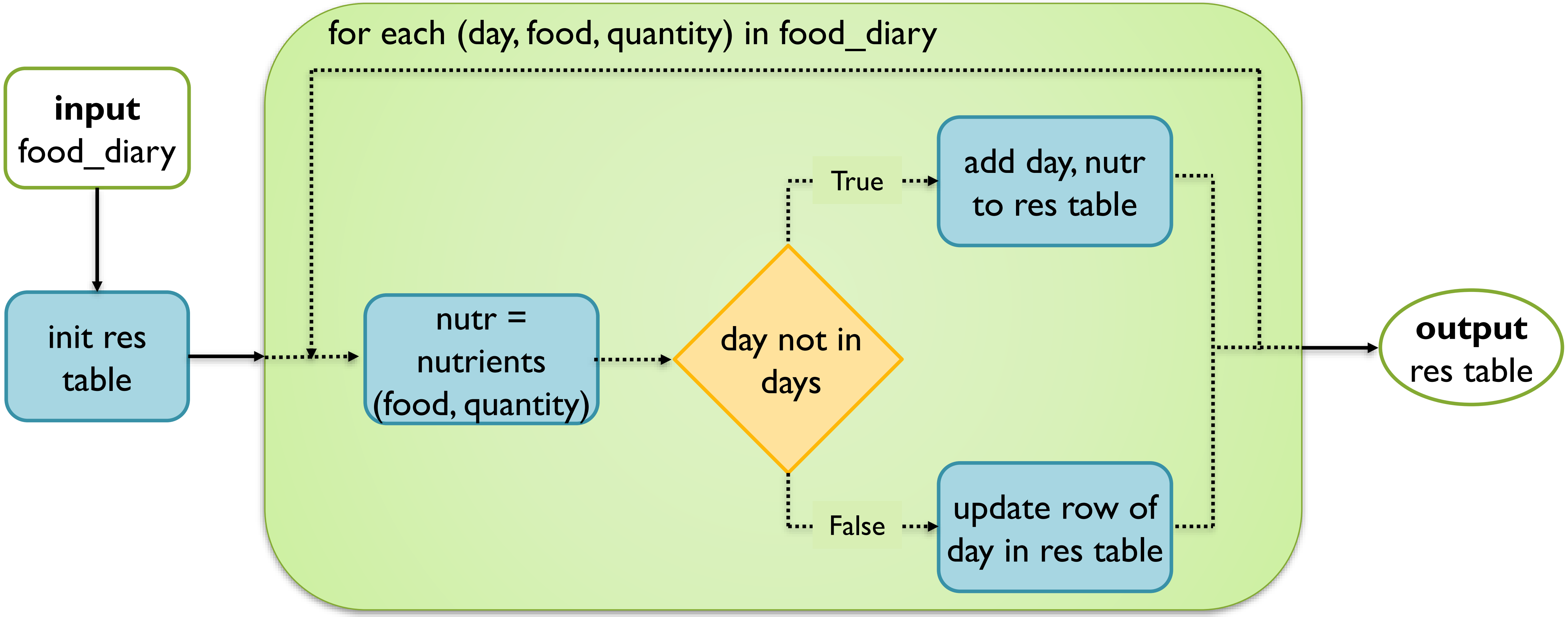
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
		...	

food diary



day	energy	water	protein	carbs	sugars	fat	fibres
1	3354	705.7	81.2	67.6	15.8	18.5	18
2	1868	553.2	21.8	62.1	14.2	8.45	16.55
3	2833.4	621.31	11.1	72.89	20.51	35.58	11.52
				...			

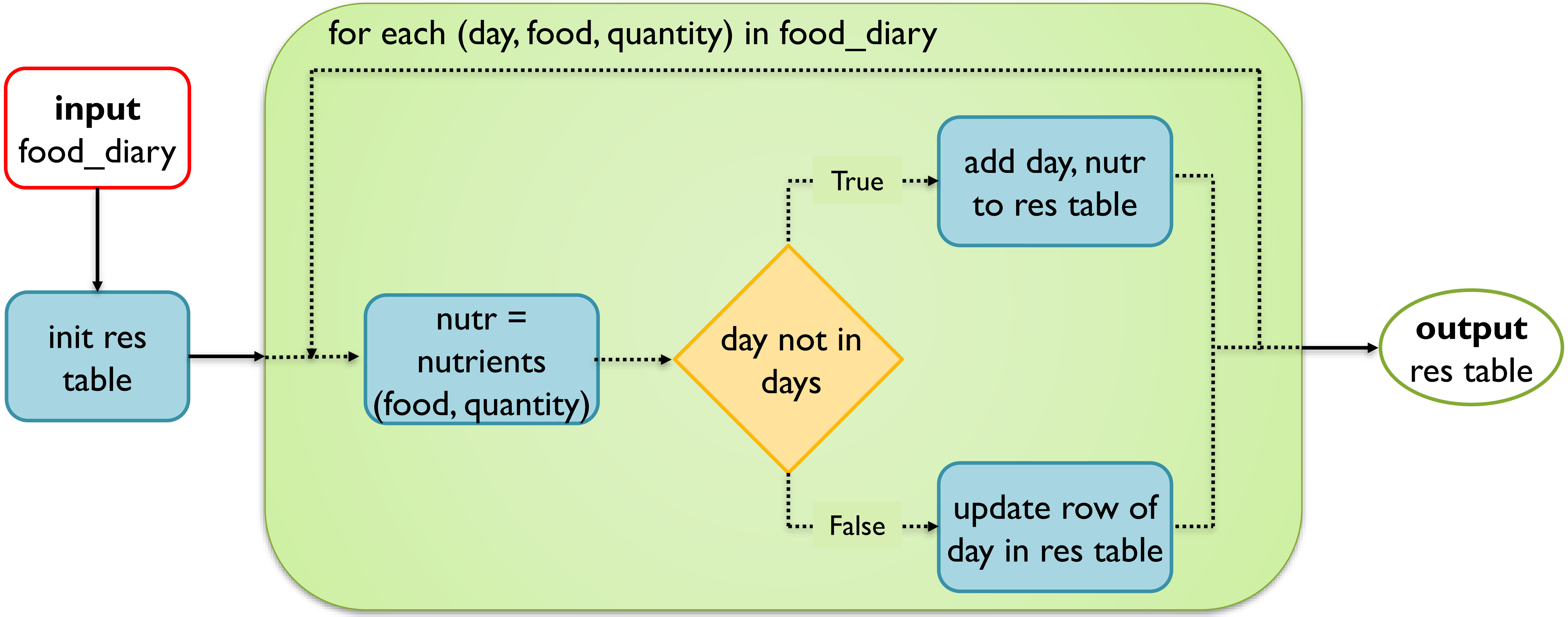
nutritional intake per day


for each (day, food, quantity) in food_diary
input
food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output
res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

food diary



input
food_diary

init res table

for each (day, food, quantity) in food_diary

nutr = nutrients (food, quantity)

day not in days

True

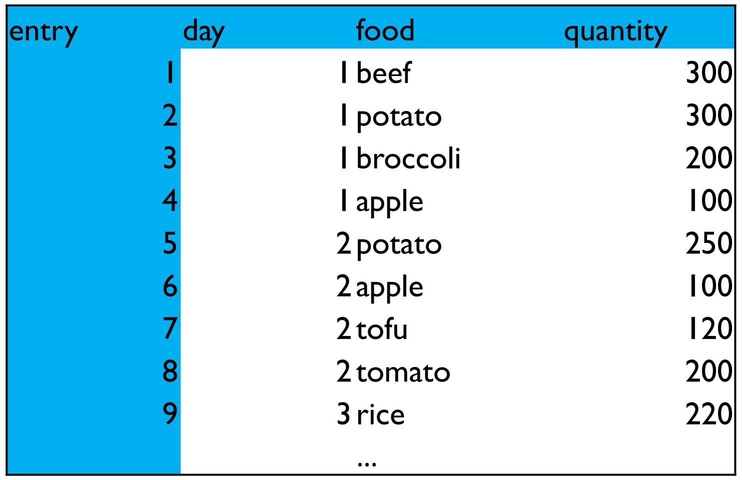
add day, nutr to res table

False

update row of day in res table

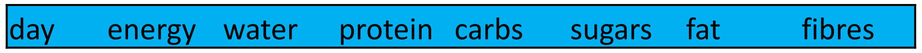
output
res table

Computing nutrient intake per day



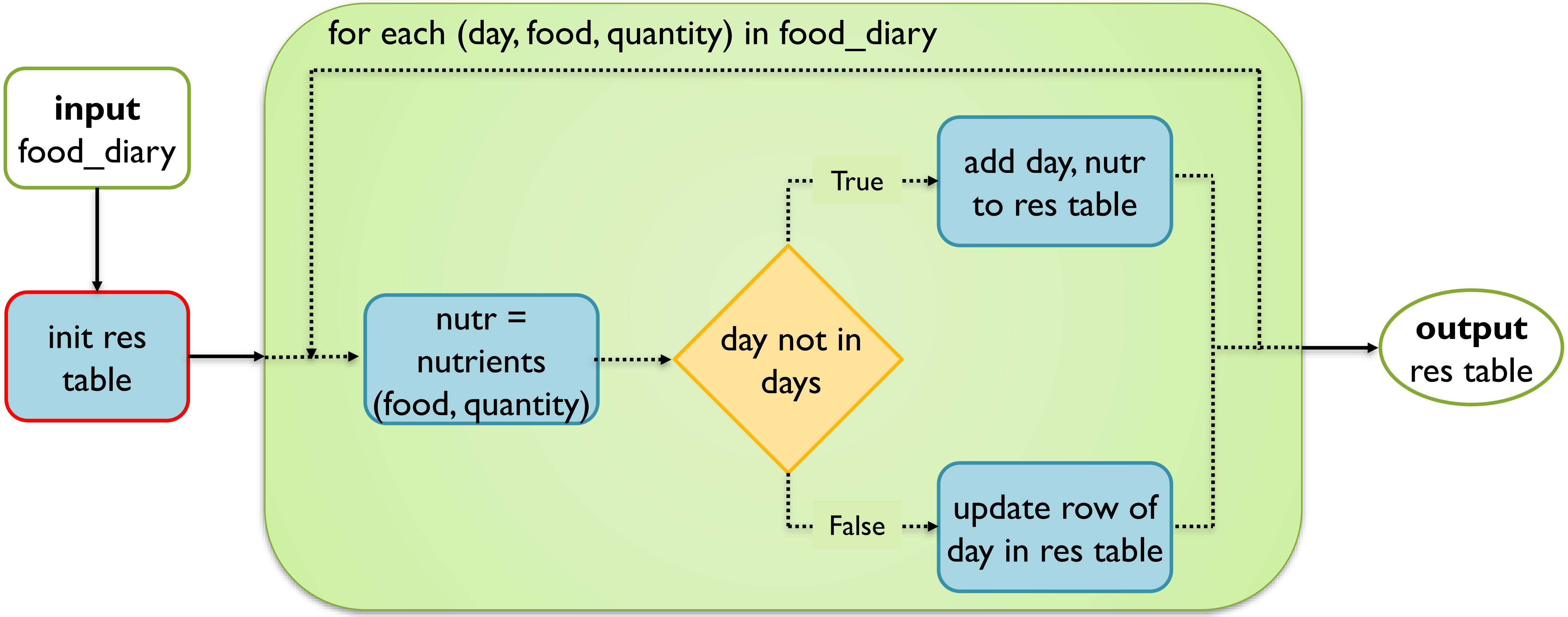
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

food diary



day	energy	water	protein	carbs	sugars	fat	fibres

res table



Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

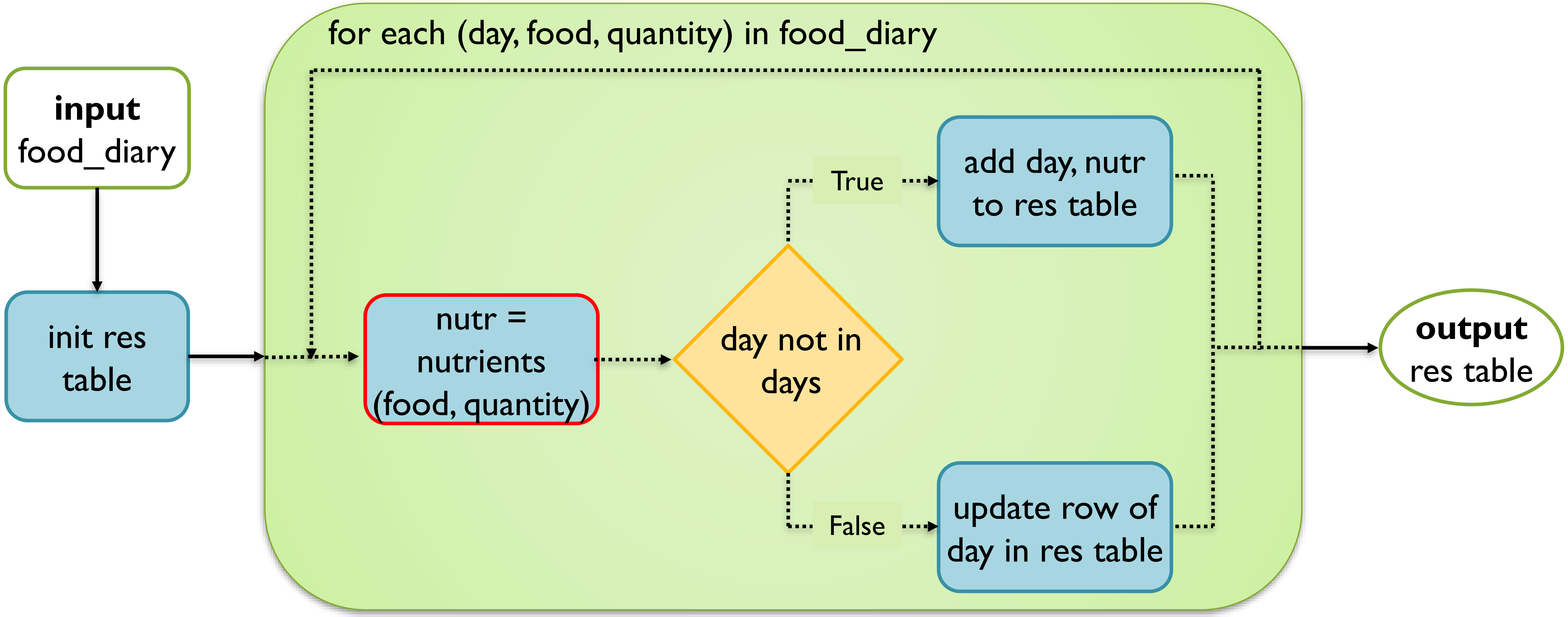
food diary

day	energy	water	protein	carbs	sugars	fat	fibres

res table

nutr

1839	210	68.4	0.6	0	18	0


for each (day, food, quantity) in food_diary
input food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

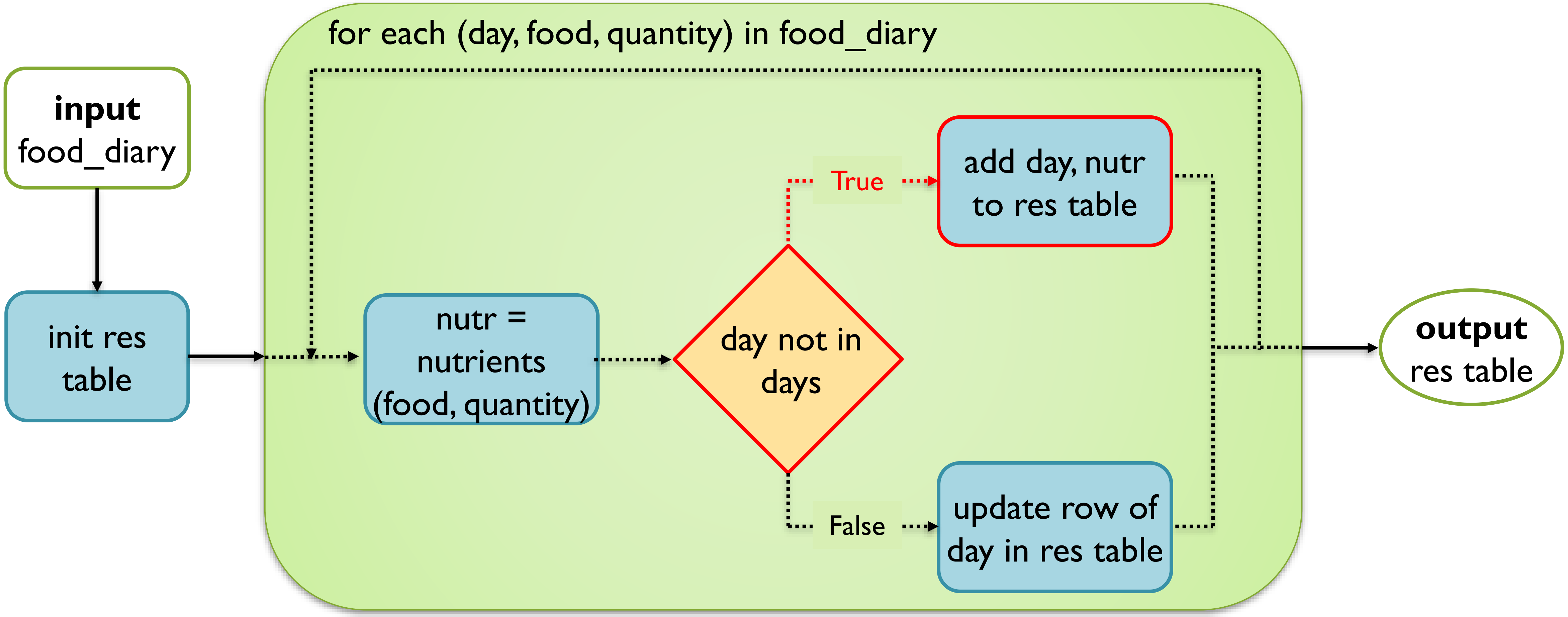
food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	1839	210	68.4	0.6	0	18	0

res table

nutr

1839	210	68.4	0.6	0	18	0



for each (day, food, quantity) in food_diary

input food_diary

init res table

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

False

update row of day in res table

output res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

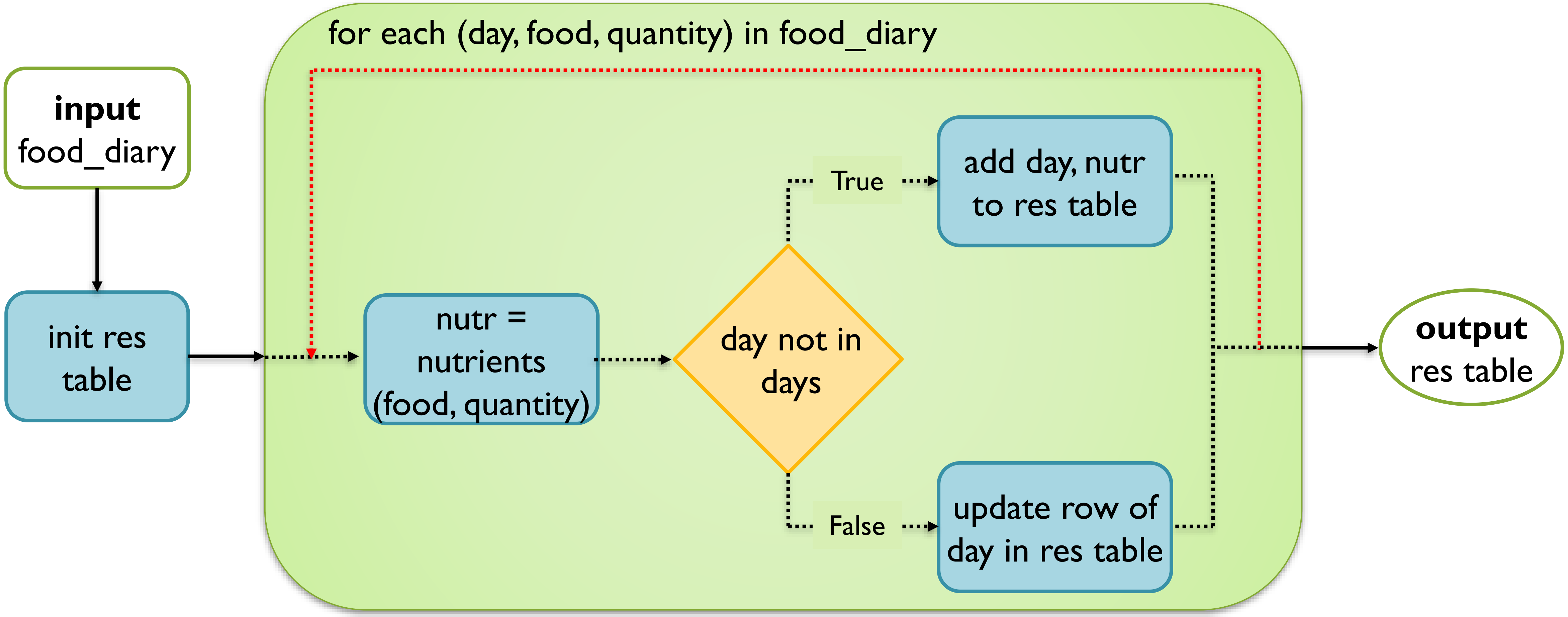
food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	1839	210	68.4	0.6	0	18	0

res table

nutr

1839	210	68.4	0.6	0	18	0


input
food_diary
init res table
for each (day, food, quantity) in food_diary
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output
res table

Computing nutrient intake per day

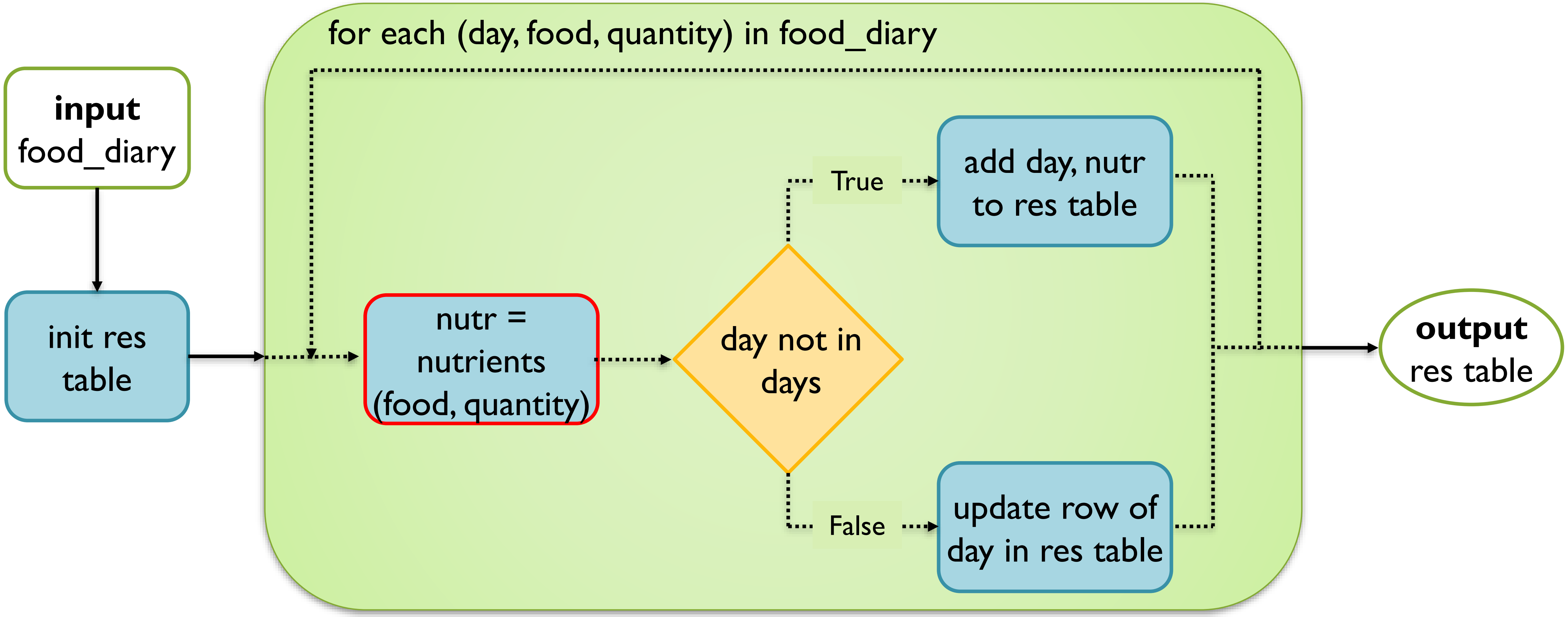
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	1839	210	68.4	0.6	0	18	0

res table

nutr							
	1038	232.2	6	51	0	0.3	7.5



for each (day, food, quantity) in food_diary
input food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

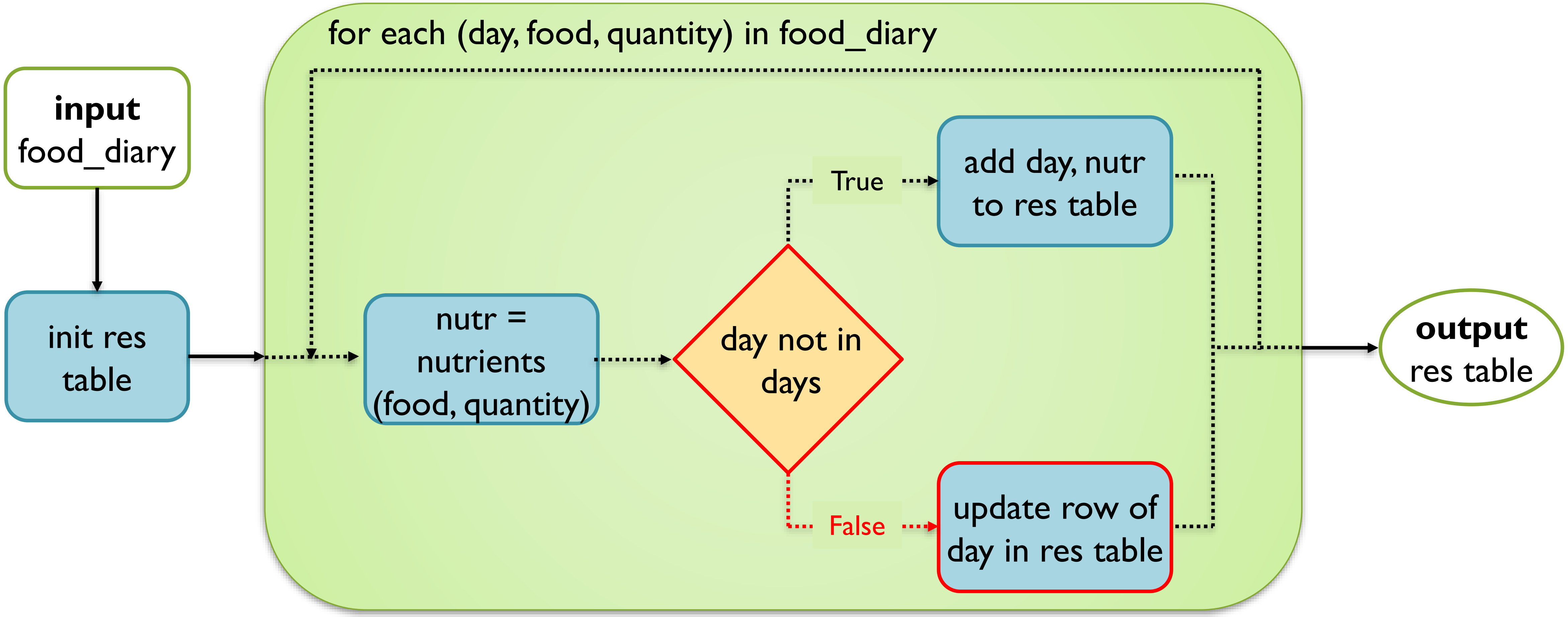
food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	2877	442.2	74.4	51.6	0.0	18.3	7.5

res table

nutr

1038	232.2	6	51	0	0.3	7.5



for each (day, food, quantity) in food_diary

input food_diary

init res table

nutr = nutrients (food, quantity)

day not in days

True

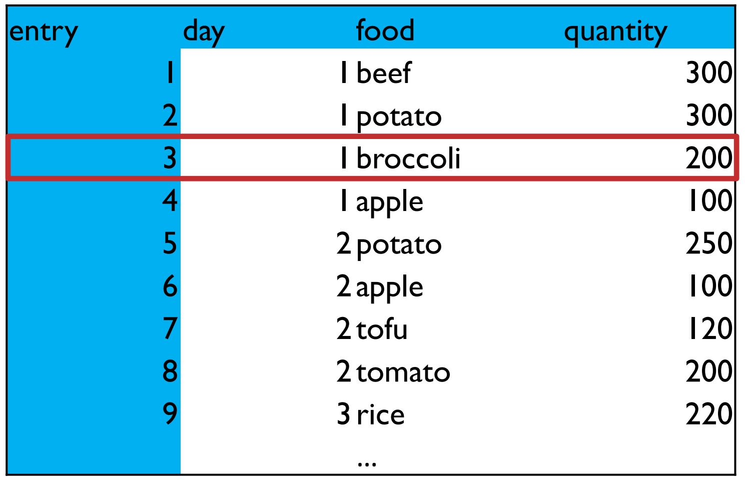
add day, nutr to res table

False

update row of day in res table

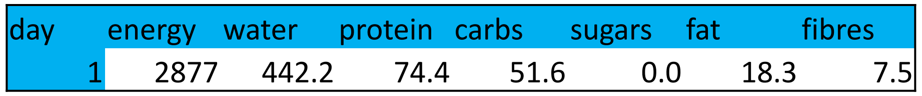
output res table

Computing nutrient intake per day



entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

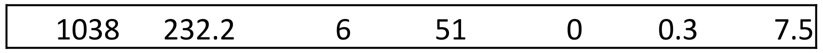
food diary



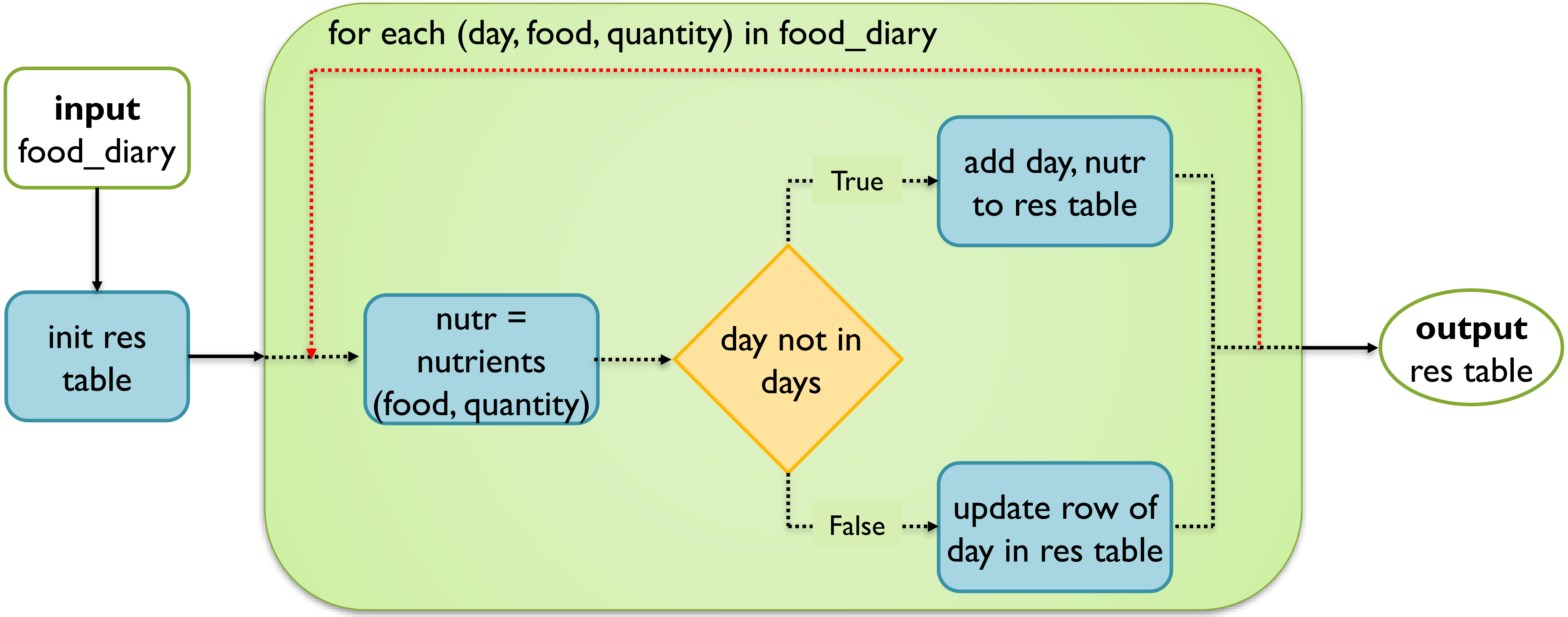
day	energy	water	protein	carbs	sugars	fat	fibres
1	2877	442.2	74.4	51.6	0.0	18.3	7.5

res table

nutr



1038	232.2	6	51	0	0.3	7.5



for each (day, food, quantity) in food_diary

input food_diary

init res table

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

False

update row of day in res table

output res table

Computing nutrient intake per day

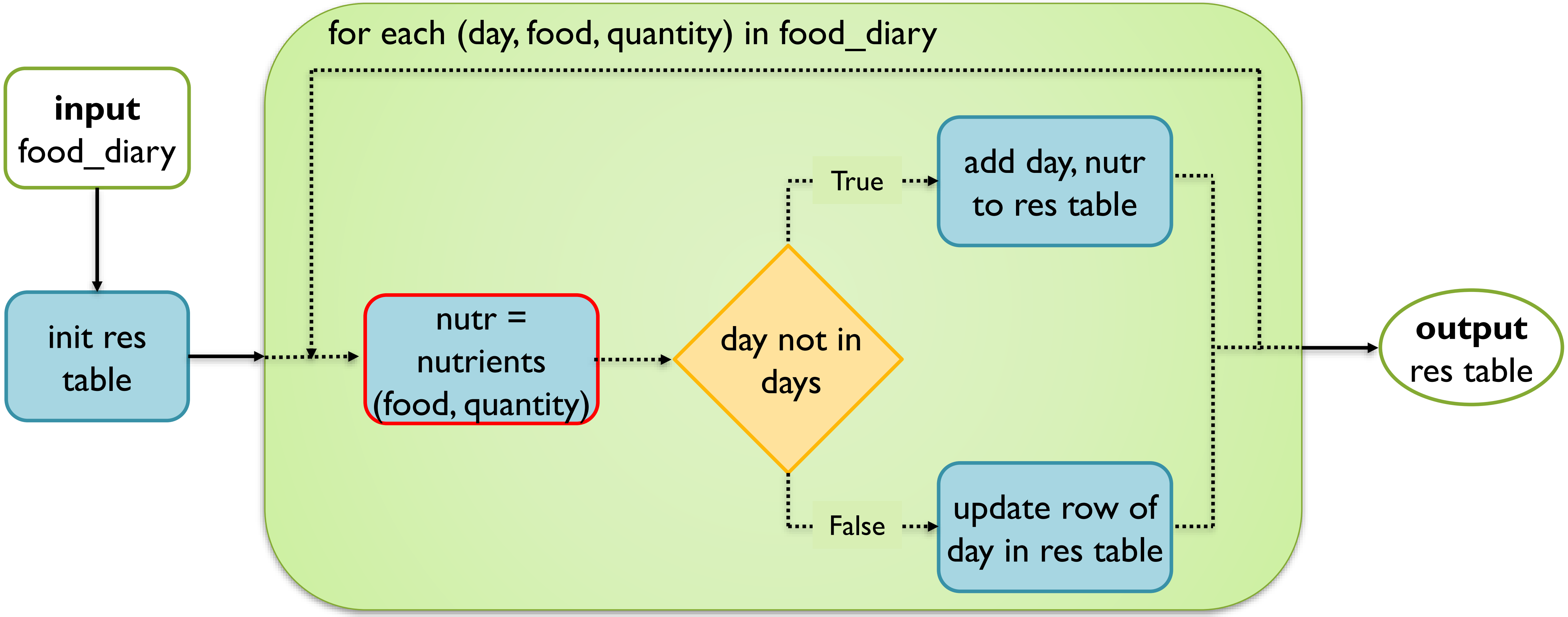
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	2877	442.2	74.4	51.6	0.0	18.3	7.5

res table

nutr	248	179.2	6.4	4.0	4.0	0.2	8.2



input food_diary

init res table

for each (day, food, quantity) in food_diary

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

False

update row of day in res table

output res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

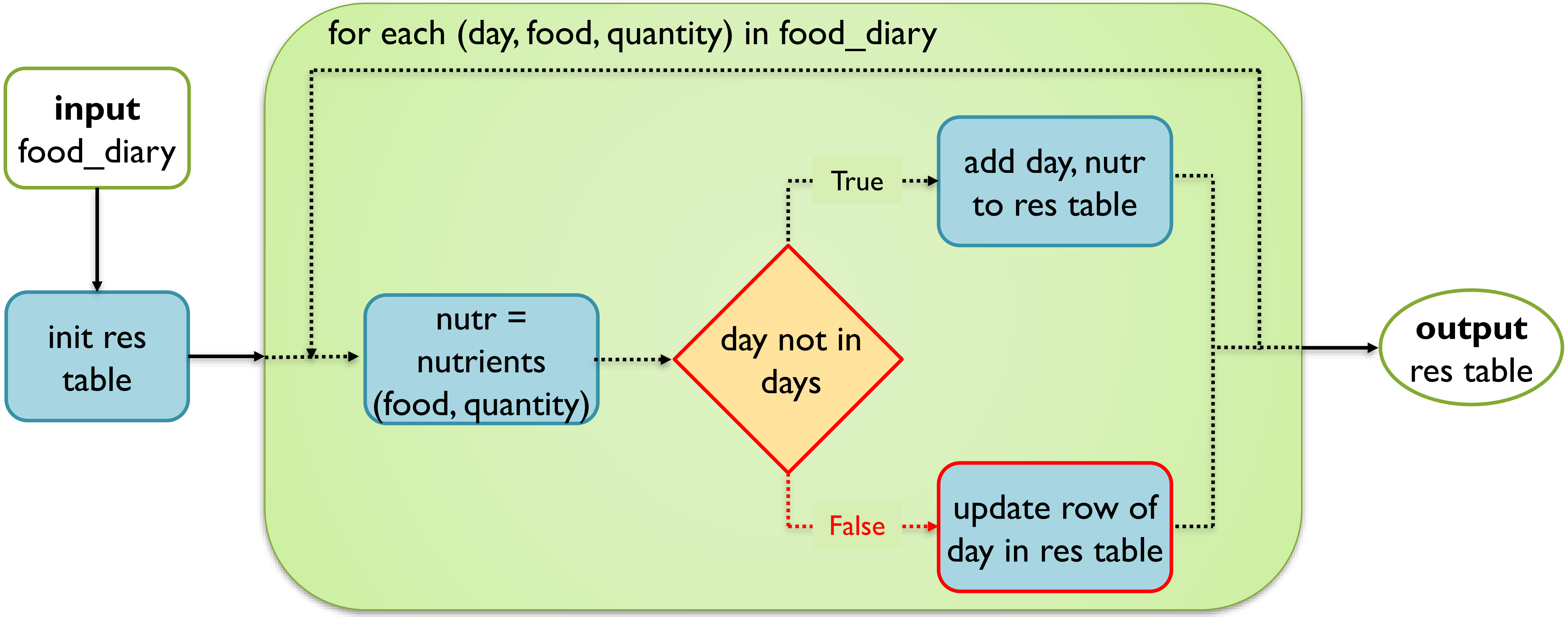
food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	3125	621.4	80.8	55.6	4.0	18.5	15.7

res table

nutr

248	179.2	6.4	4.0	4.0	0.2	8.2



input food_diary

init res table

for each (day, food, quantity) in food_diary

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

False

update row of day in res table

output res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

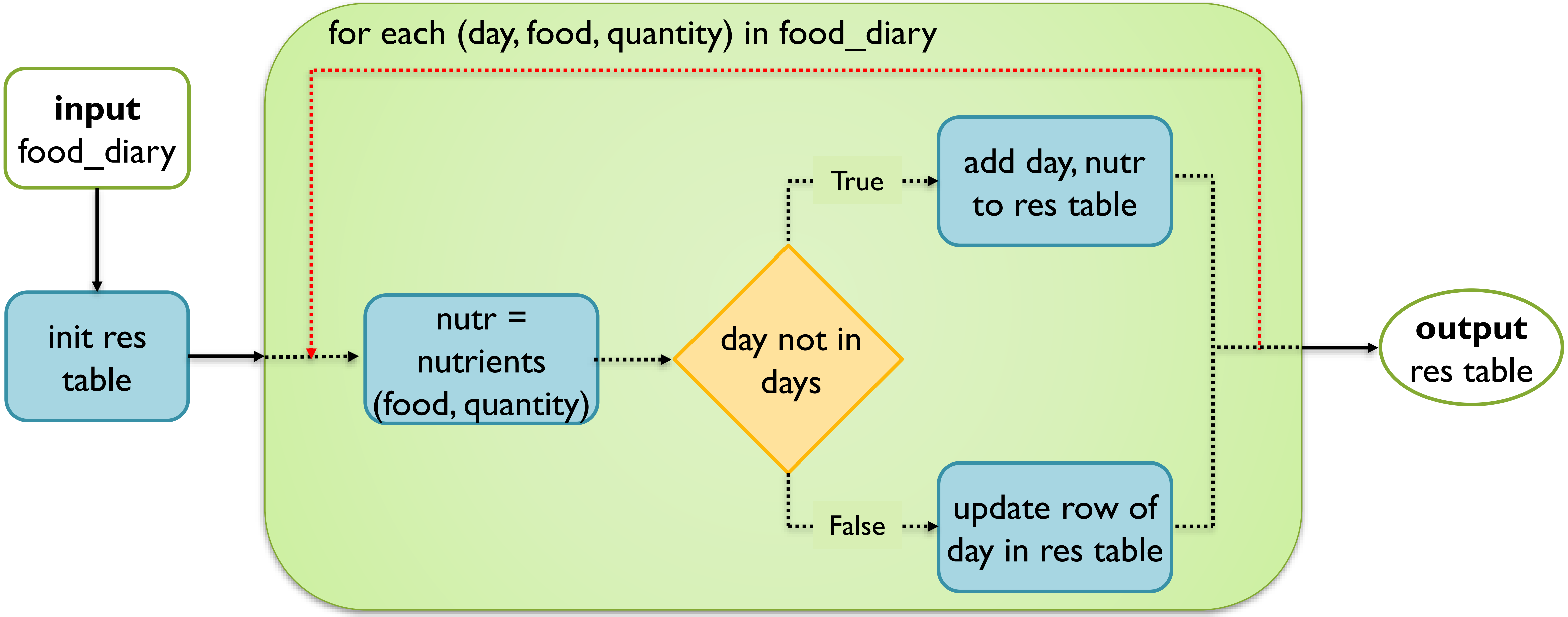
food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	3125	621.4	80.8	55.6	4.0	18.5	15.7

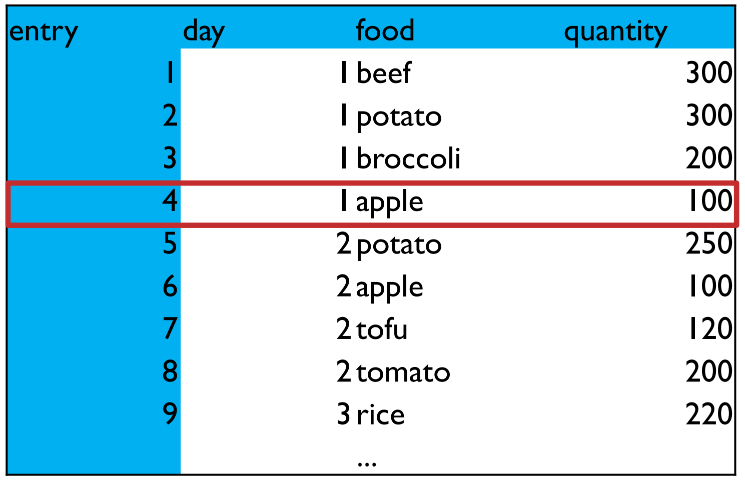
res table

nutr

248	179.2	6.4	4.0	4.0	0.2	8.2

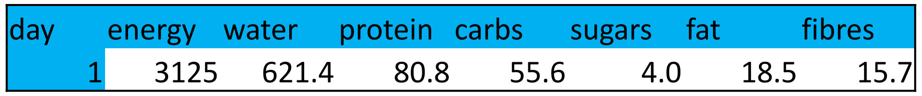

for each (day, food, quantity) in food_diary
input food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output res table

Computing nutrient intake per day



entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

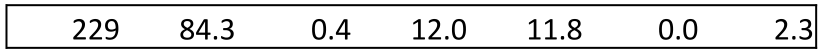
food diary



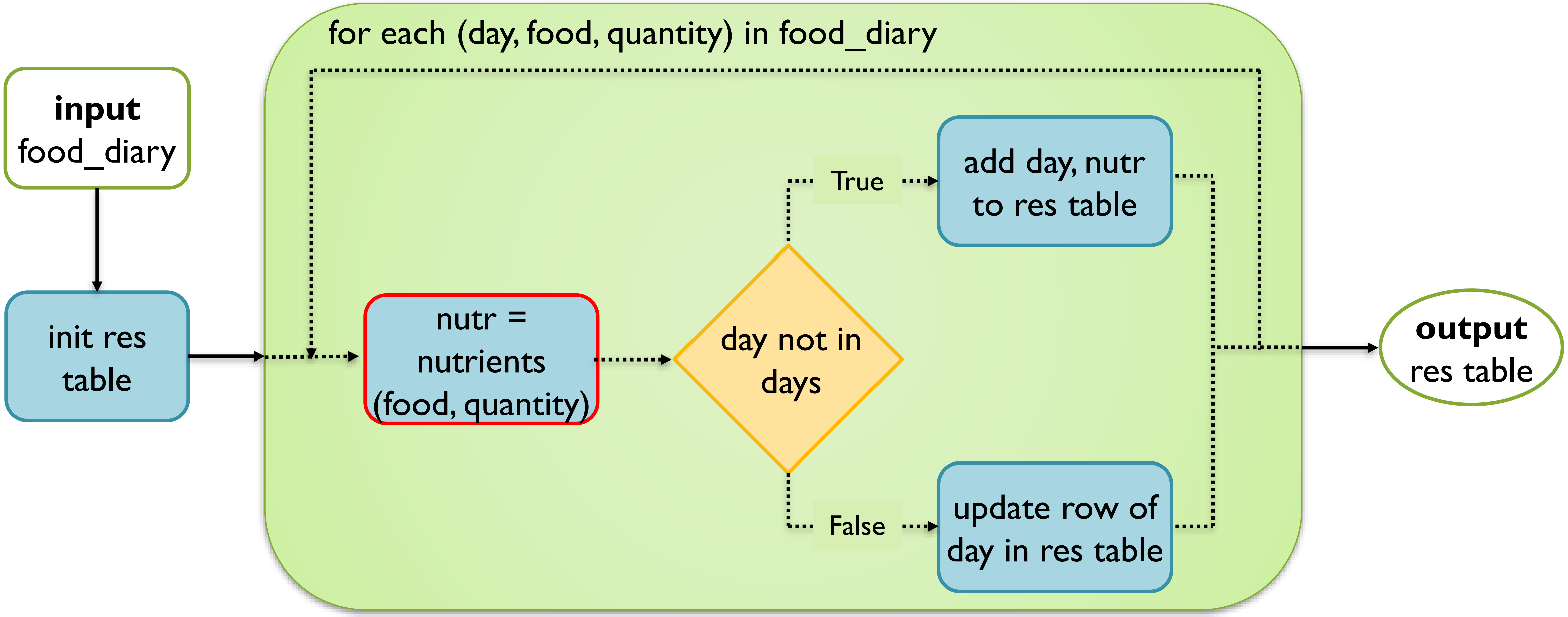
day	energy	water	protein	carbs	sugars	fat	fibres
1	3125	621.4	80.8	55.6	4.0	18.5	15.7

res table

nutr



229	84.3	0.4	12.0	11.8	0.0	2.3


for each (day, food, quantity) in food_diary
input food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output res table

Computing nutrient intake per day

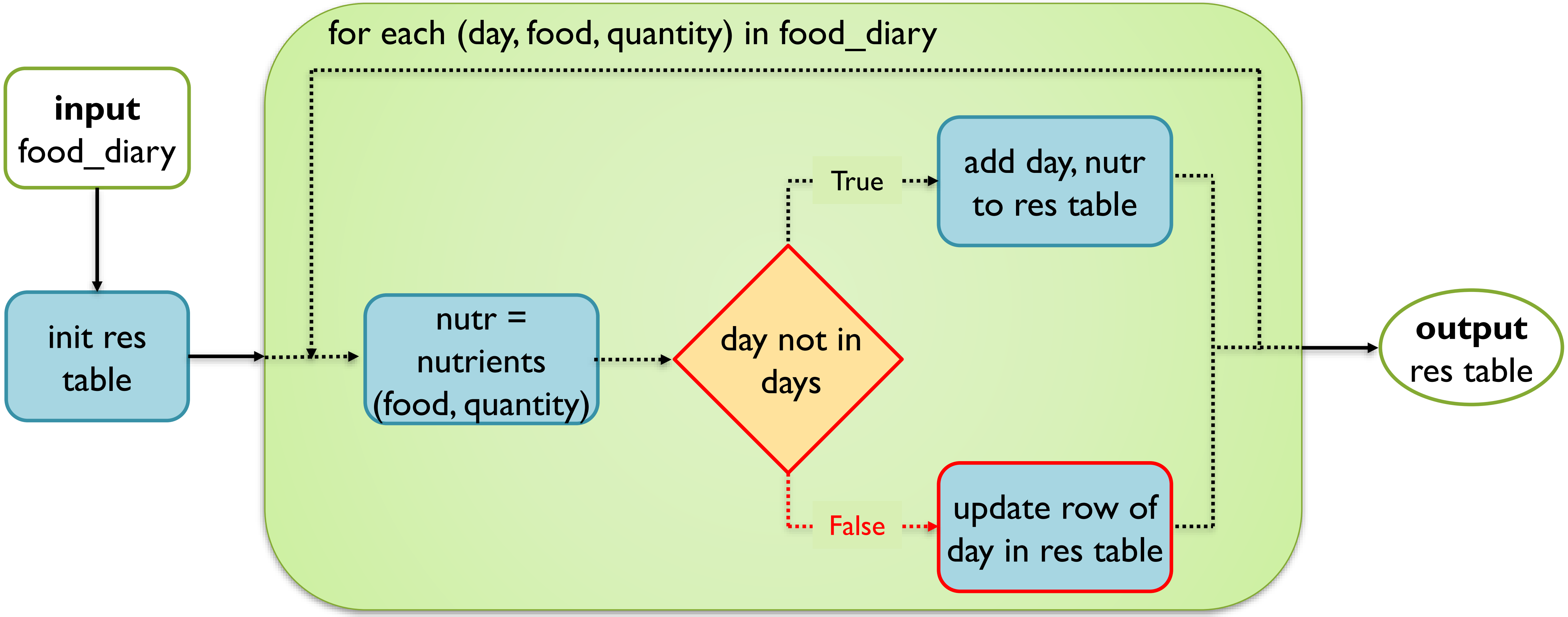
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	3354	705.7	81.2	67.6	15.8	18.5	18

res table

nutr	248	179.2	6.4	4.0	4.0	0.2	8.2



for each (day, food, quantity) in food_diary
input food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output res table

Computing nutrient intake per day

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

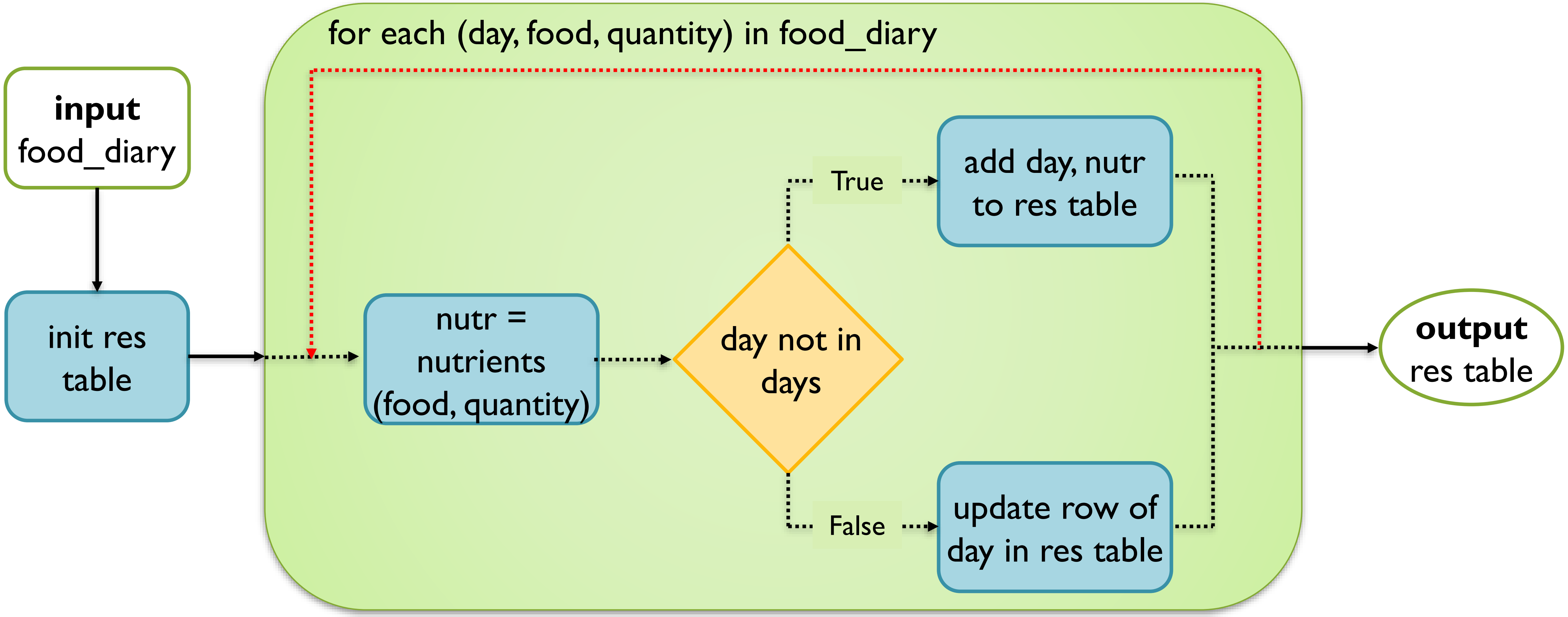
food diary

day	energy	water	protein	carbs	sugars	fat	fibres
1	3354	705.7	81.2	67.6	15.8	18.5	18

res table

nutr

248	179.2	6.4	4.0	4.0	0.2	8.2



input
food_diary

init res table

for each (day, food, quantity) in food_diary

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

False

update row of day in res table

output
res table

Computing nutrient intake per day

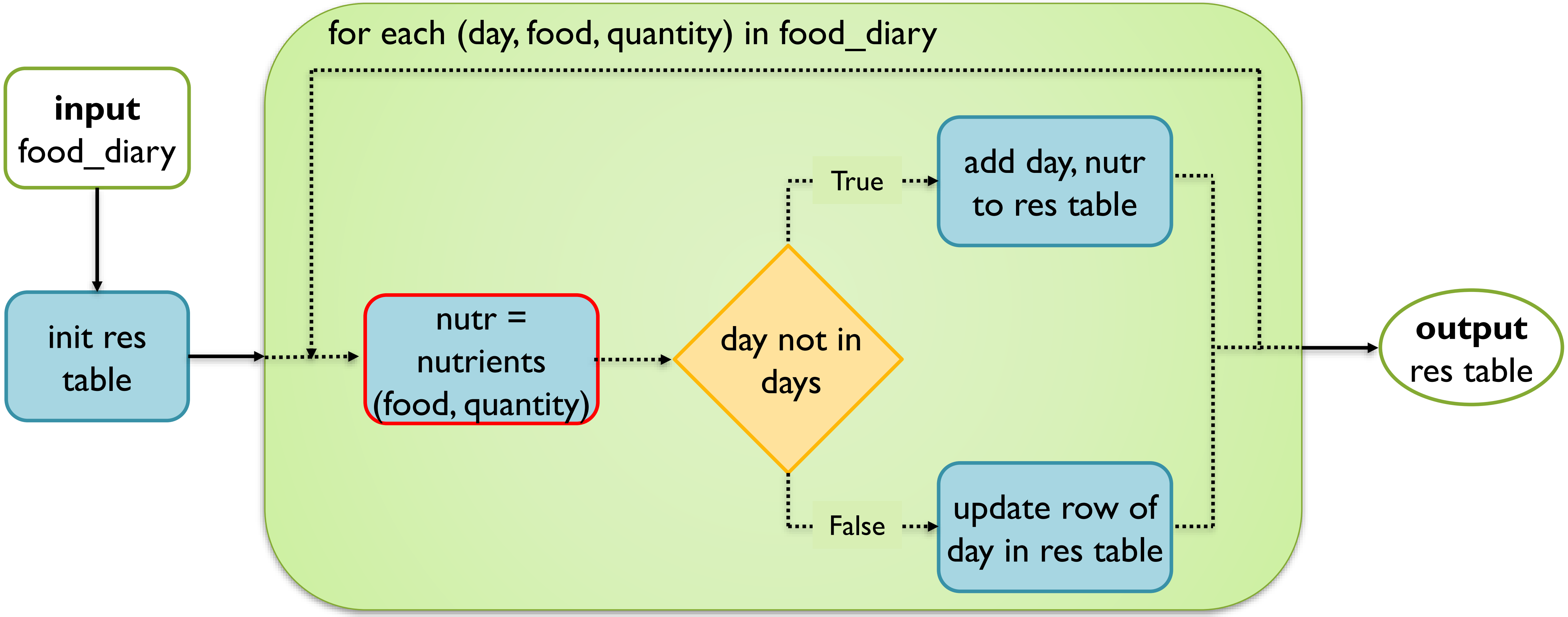
entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

food diary

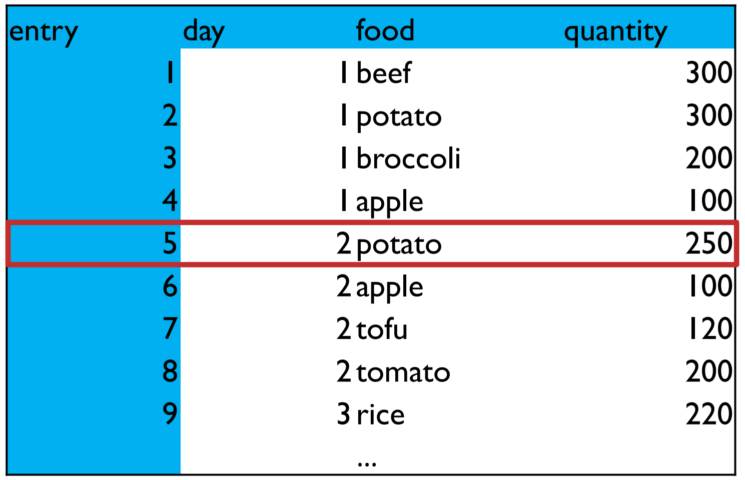
day	energy	water	protein	carbs	sugars	fat	fibres
1	3354	705.7	81.2	67.6	15.8	18.5	18

res table

nutr							
	865	193.5	5	42.5	0	0.25	6.25

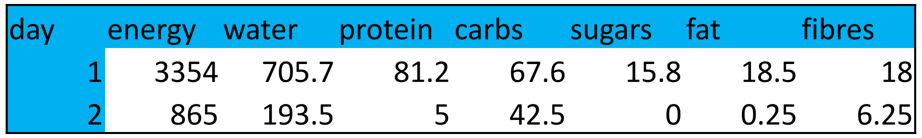

for each (day, food, quantity) in food_diary
input
food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output
res table

Computing nutrient intake per day



entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
5	2	potato	250
6	2	apple	100
7	2	tofu	120
8	2	tomato	200
9	3	rice	220
		...	

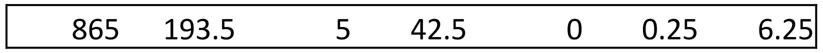
food diary



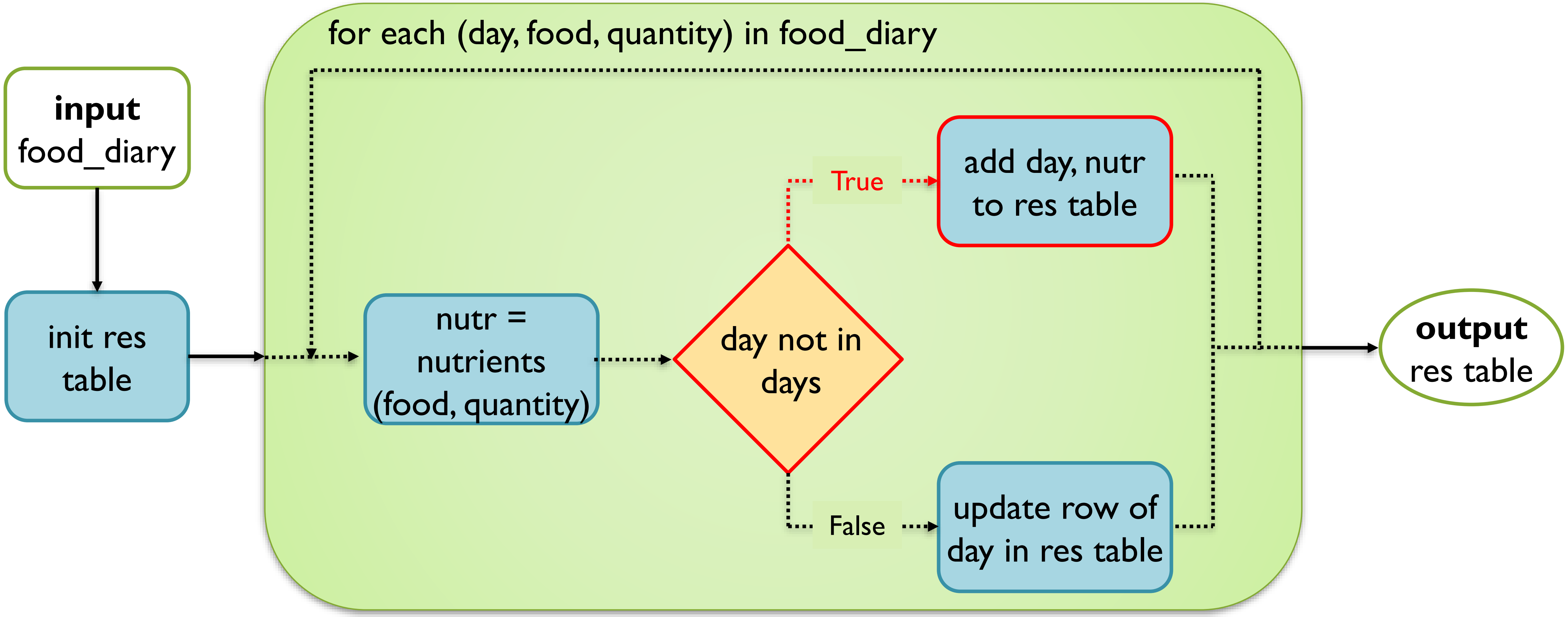
day	energy	water	protein	carbs	sugars	fat	fibres
1	3354	705.7	81.2	67.6	15.8	18.5	18
2	865	193.5	5	42.5	0	0.25	6.25

res table

nutr

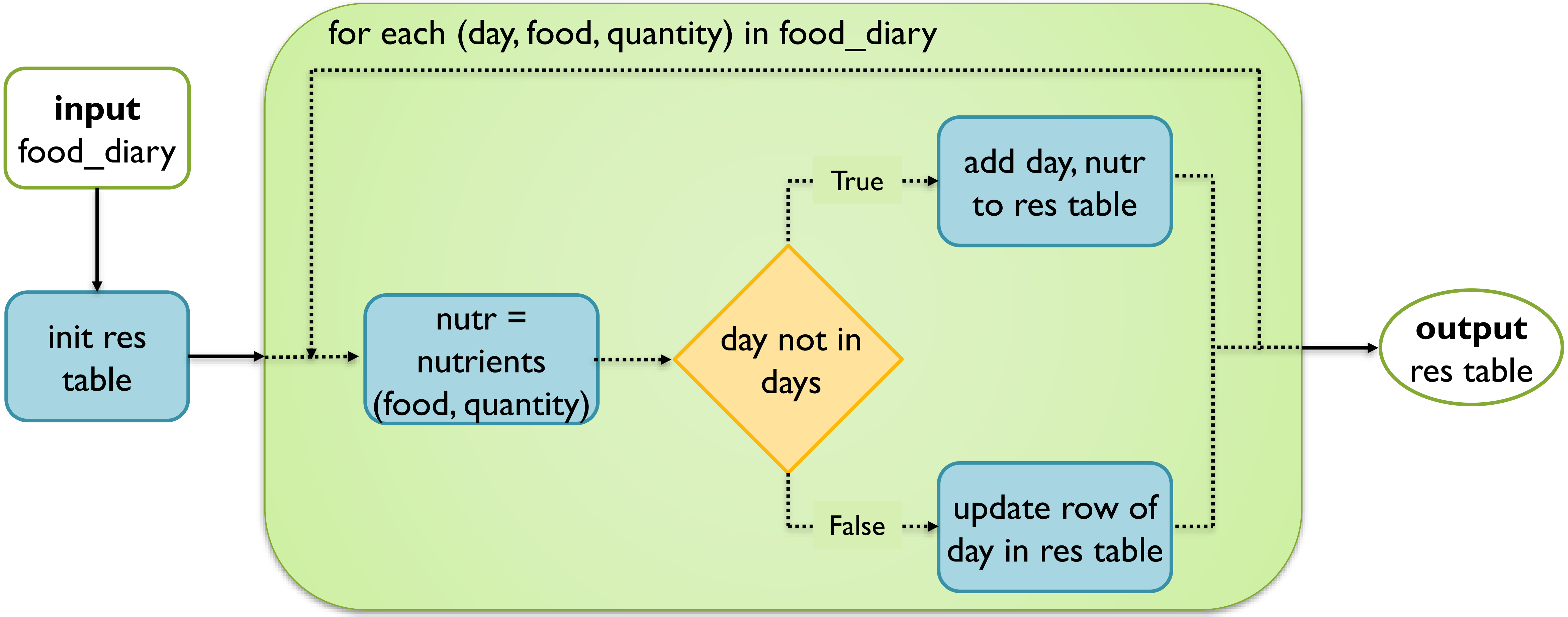


865	193.5	5	42.5	0	0.25	6.25


input
food_diary
init res table
for each (day, food, quantity) in food_diary
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output res table

Computing nutrient intake per day

```
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            # append day to days
            # append nutr to intake
        else:
            # update last entry of intake by adding nutr
    return intake, days
```


for each (day, food, quantity) in food_diary
input
food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output
res table

Finding nutrients for single entry of food diary

entry	day	food	quantity
1	1	beef	300
2	1	potato	300
3	1	broccoli	200
4	1	apple	100
		...	

food	energy	water	protein	carbs	sugars	fat	fibres
apple	229	84.3	0.4	12	11.8	0	2.3
orange	186	84.3	1	9.5	8.3	0.2	2.1
				...			
potato	346	77.4	2	17	0	0.1	2.5
				...			

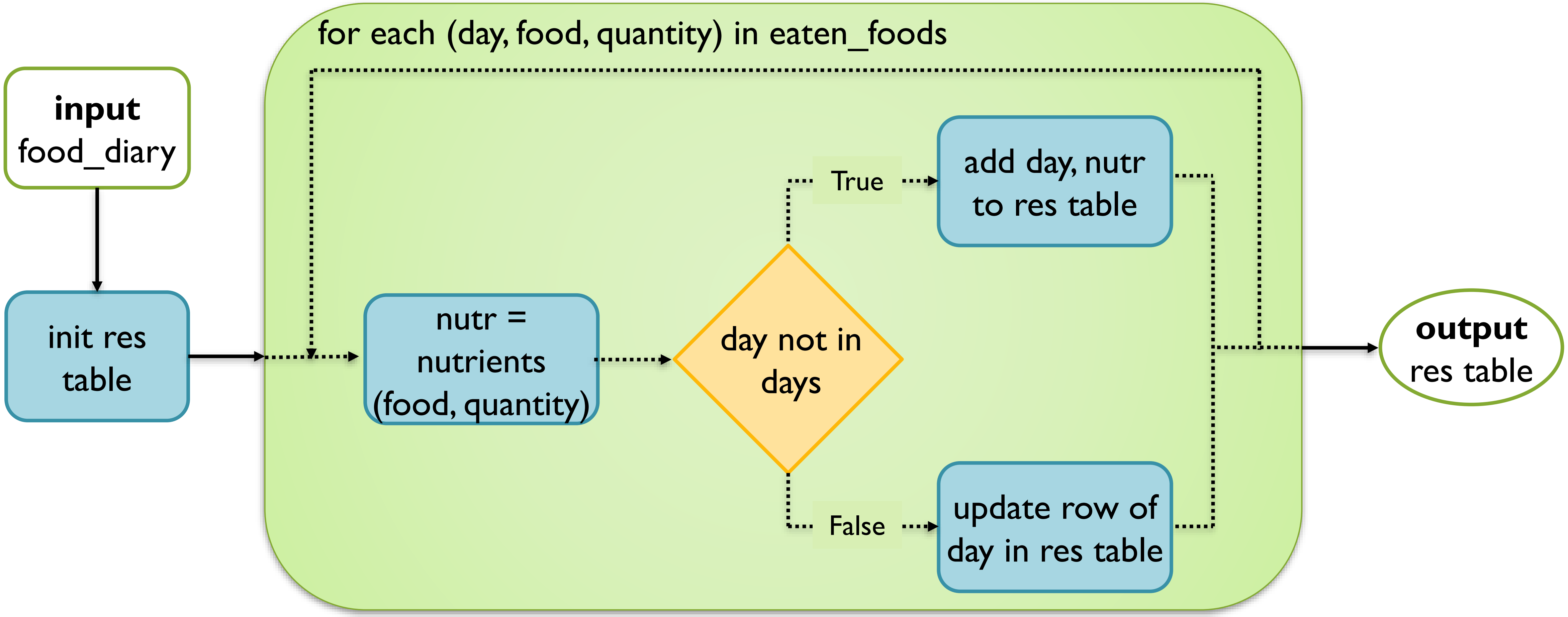
energy	water	protein	carbs	sugars	fat	fibres
1038	232.2	6	51	0	0.3	6.9

```python
def nutrients(food, quantity):
    for i in range(len(foods)):
        if foods[i]==food:
            nutr_100g = nutr_vals[i]
            return scaled(nutr_100g, quantity/100)
```

```python
def scaled(row, alpha):
    """
    Input : list with numeric entries (row), scaling factor (alpha)
    Output: new list (res) of same length with res[i]==row[i]*alpha

    For example:
    >>> scaled([1, 4, -1], 2.5)
    [2.5, 10.0, -2.5]
    """
```

Computing nutrient intake per day

```
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            # append day to days
            # append nutr to intake
        else:
            # update last entry of intake by adding nutr
    return intake, days
```



input
food_diary

init res table

for each (day, food, quantity) in eaten_foods

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

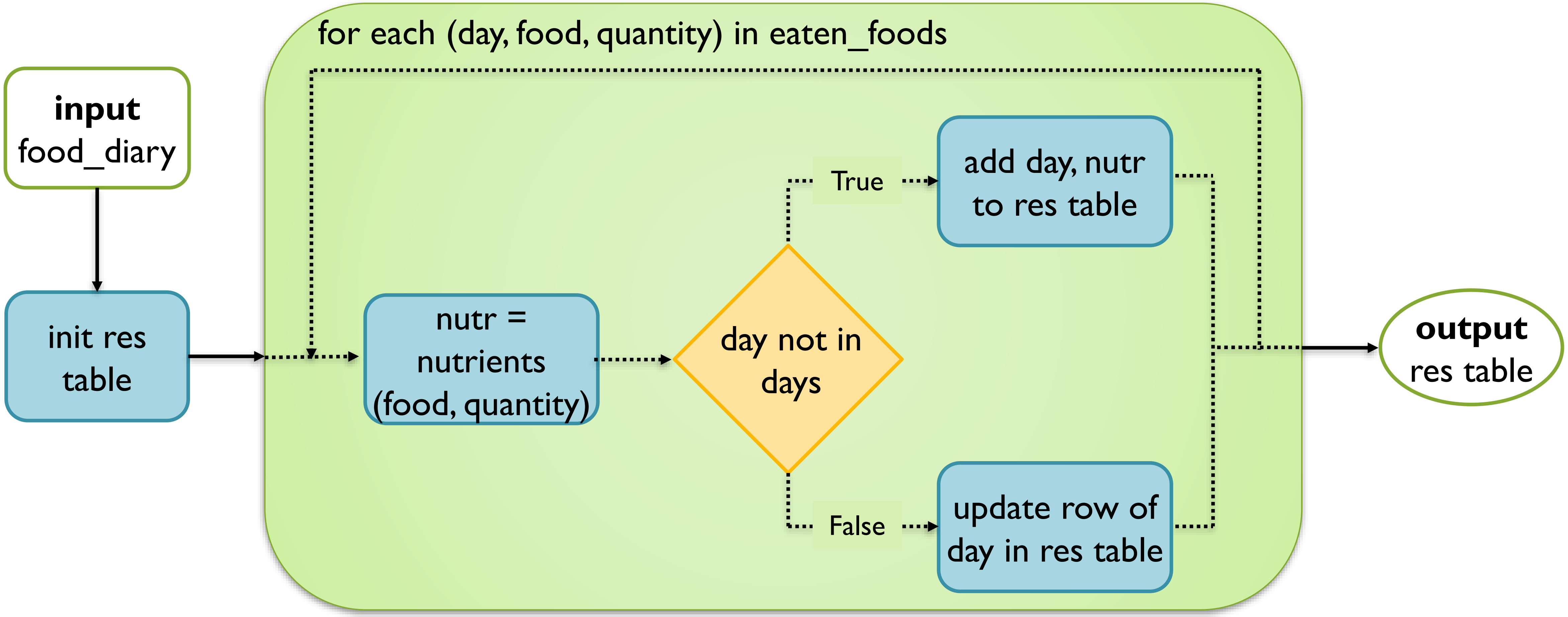
False

update row of day in res table

output
res table

Computing nutrient intake per day

```python
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            days = days + [day]
            intake = intake + [nutr]
        else:
            # update last entry of intake by adding nutr
    return intake, days
```



for each (day, food, quantity) in eaten_foods

input food_diary

init res table

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

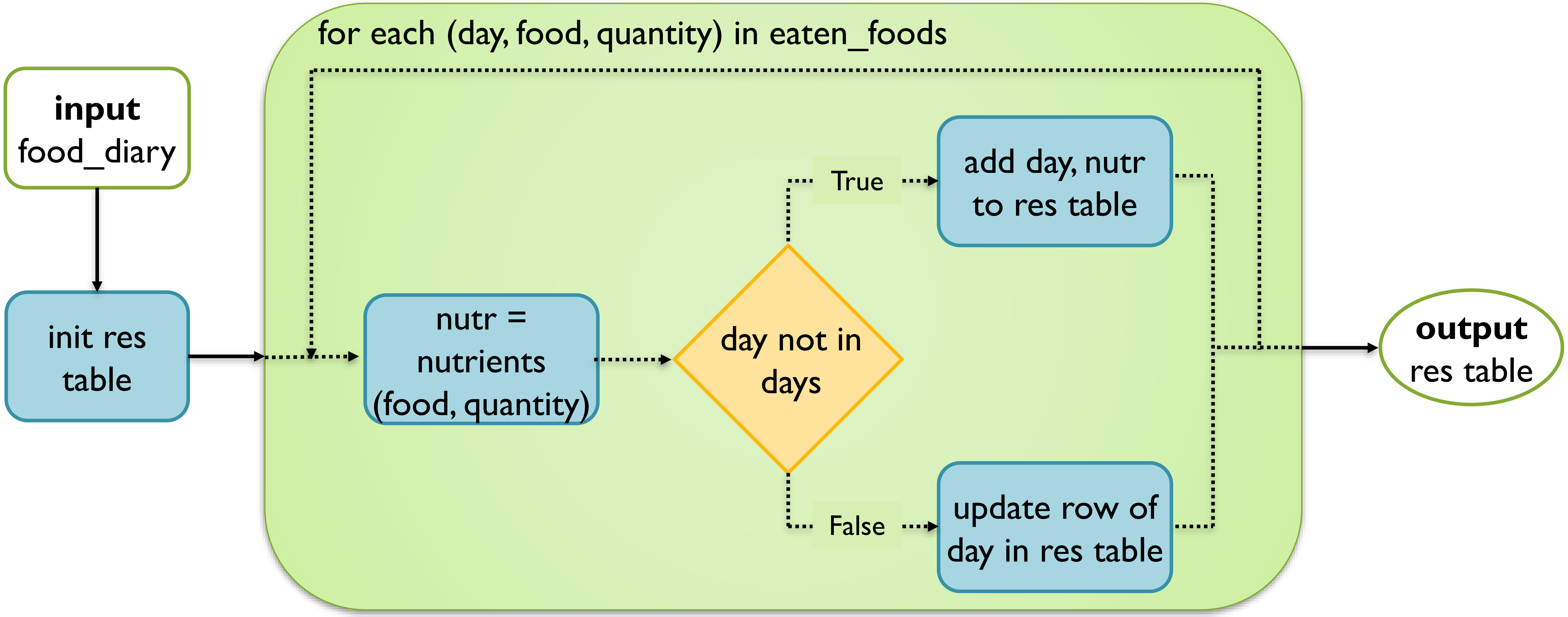
False

update row of day in res table

output res table

Computing nutrient intake per day

```
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            days = days + [day]
            intake = intake + [nutr]
        else:
            # update last entry of intake by adding nutr
    return intake, days
```



for each (day, food, quantity) in eaten_foods

input
food_diary

init res table

nutr = nutrients (food, quantity)

day not in days

True

add day, nutr to res table

False

update row of day in res table

output
res table

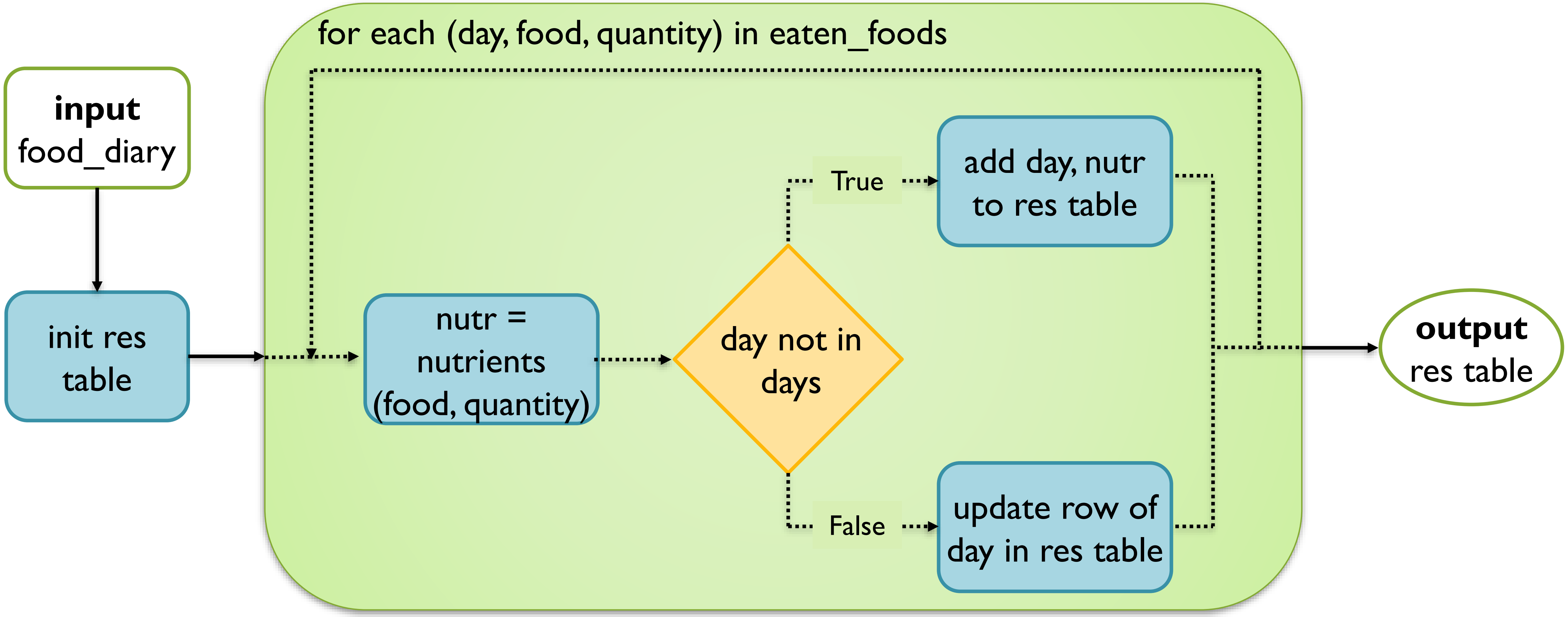
List objects are *mutable*

```
>>> items
['milk', 'eggs', 'bread', 'jam', 'soup']
>>> items[0] = 'JOGHURT'
>>> items
['JOGHURT', 'eggs', 'bread', 'jam', 'soup']
>>> 'abcde'[0] = 'z'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item
assignment
```

String are *immutable*, i.e., they cannot be modified

Computing nutrient intake per day

```
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            days = days + [day]
            intake = intake + [nutr]
        else:
            intake[-1] = sum_of_rows(intake[-1], nutr)
    return intake, days
```



for each (day, food, quantity) in eaten_foods
input
food_diary
init res table
nutr = nutrients (food, quantity)
day not in days
True
add day, nutr to res table
False
update row of day in res table
output
res table

Computing nutrient intake per day

```
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            days = days + [day]
            intake = intake + [nutr]
        else:
            intake[-1] = sum_of_rows(intake[-1], nutr)
    return intake, days
```

```
def sum_of_rows(r1, r2):
    """
    Input : two lists (r1, r2) with same number of numeric entries
    Output: new list (res) of same length with res[i]==r1[i]+r2[i]
            for all i in range(len(r1))

    For example:
    >>> sum_of_rows([100, -4, 10], [0, 3.5, -10])
    [100, -0.5, 0]
    """
    return [a[j]+b[j] for j in range(n)]
```

Computing nutrient intake per day

```
def intake_per_day(food_diary):
    days, intake = [], []
    for day, food, quantity in food_diary:
        nutr = nutrients(food, quantity)
        if day not in days:
            days = days + [day]
            intake = intake + [nutr]
        else:
            intake[-1] = sum_of_rows(intake[-1], nutr)
    return intake, days
```

```
nvals, nutr_cols, foods = table_from_file('nutr_tab.csv', range(7))
food_diary, _, _ = table_from_file('food_diary.csv', [3])
intake, days = intake_per_day(food_diary)
```

```
>>> intake
[[3354.0, 705.7, 81.20000000000002, 67.6, 15.8, 18.5, 18.0],
[1868.0, 553.2, 21.799999999999997, 62.099999999999994,
14.200000000000001, 8.450000000000001, 16.55], [2833.4, 621.31,
11.1, 72.89, 20.51, 35.58, 11.52]]
>>>
```

Finally: write output into file for user

```
def table_to_file(vals, cols, ids, filename):
    """
    Writes a table with column names and ids to csv file.

    Input : table (vals) with column names (cols), and
            row ids (ids), name of output file (filename)
    Output: None; writes table to file
    """
```

```
>>> intake, days = intake_per_day(food_diary)
>>> intake
[[3354.0, 705.7, 81.20000000000002, 67.6, 15.8, 18.5, 18.0], [1868.0, 
553.2, 21.799999999999997, 62.099999999999994, 14.200000000000001, 
8.450000000000001, 16.55], [2833.4, 621.31, 11.1, 72.89, 20.51, 35.58, 
11.52]]
>>> table_to_file(intake, nutrient_names, days, 'intake_per_day.csv')
```

intake_per_day.csv

```
id,energy (kJ),water,protein,carbs,sugars,fat,fibres
1,3354.0,705.7,81.2,67.6,15.8,18.5,18.0
2,1868.0,553.2,21.8,62.1,14.2,8.45,16.55
3,2833.4,621.31,11.1,72.89,20.51,35.58,11.52
```

Finally: write output into file for user

```
def table_to_file(vals, cols, ids, filename):
    """
    Writes a table with column names and ids to csv file.

    Input : table (vals) with column names (cols), and
            row ids (ids), name of output file (filename)
    Output: None; writes table to file
    """
    file = open(filename, 'w')
    header = 'id,' + ','.join(cols) + '\n'
    file.write(header)
    for i in range(len(vals)):
        line = str(ids[i]) + ',' + ','.join(as_str(vals[i])) + '\n'
        file.write(line)
    file.close()
```

```
def as_str(lst):
    """Converts lst of objects to list of strings."""
    res = []
    for x in lst:
        res.append(str(x))
    return res
```

Finally: write output into file for user

```
def table_to_file(vals, cols, ids, filename):
    file = open(filename, 'w')
    header = 'id,' + ','.join(cols) + '\n'
    file.write(header)
    for i in range(len(vals)):
        line = str(ids[i]) + ',' + ','.join(as_str(vals[i])) + '\n'
        file.write(line)
    file.close()
```

```
nvals, nutr_cols, foods = table_from_file('nutr_tab.csv', range(7))
food_diary, _, _ = table_from_file('food_diary.csv', [3])
intake, days = intake_per_day(food_diary)
table_to_file(intake, nutrient_names, days, 'intake_per_day.csv')
```

intake_per_day.csv

```
id,energy (kJ),water,protein,carbs,sugars,fat,fibres
1,3354.0,705.7,81.2,67.6,15.8,18.5,18.0
2,1868.0,553.2,21.8,62.1,14.2,8.45,16.55
3,2833.4,621.31,11.1,72.89,20.51,35.58,11.52
```

Summary

- Tables can be represented as list of lists, each of which describing a "data point" described by a common set of columns
- Multiple assignment statements can be combined with sequence unpacking for intuitive for loops
- The built-in function open creates iterable file-object (files should be explicitly closed)
- Method write can be used to write to file (opened in write-mode)

Recommended reading

"*Introduction to Computing using Python: An Application Development Focus*", by L. Perkovic

- **Sections 4.3 (files)**

Next week

- Diving deeper into the Python program execution
- Sorting