DATA SCIENCE 101:
PYTHON ITERATIONS (PART 2)

AGENDA

- Go through homework
- Recap
- While Loops
- Nested Loops
- Applied Iterations
- Counting with Dictionaries

RECAP OF PREVIOUS LESSON

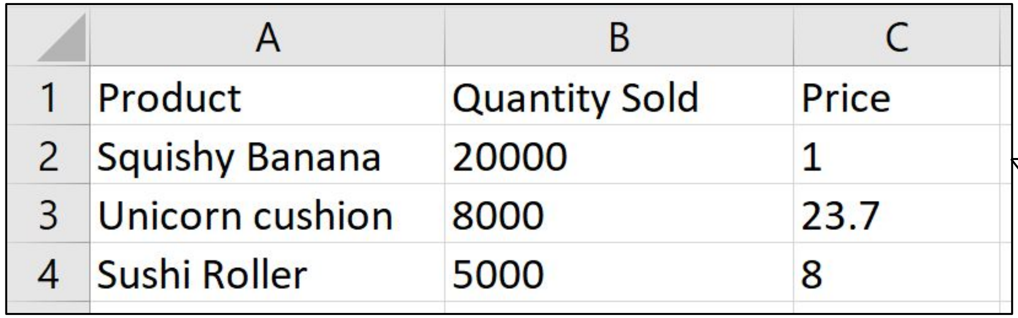
- Iterations Part 1
    - Iterating through singular list (counting, finding certain elements, and aggregating & statistics)
    - Iterating through dictionary

INTRODUCTION

- Concept of accessing data in a nested list
- Consolidation with jupyter notebook practice

REMEMBER WE COVERED THIS IN COLLECTIONS?

- How are collections applied or used for data science?



	A	B	C
1	Product	Quantity Sold	Price
2	Squishy Banana	20000	1
3	Unicorn cushion	8000	23.7
4	Sushi Roller	5000	8

Imagine your company have the following excel data and you want to port it over to python to do further analysis

So, the next question is then, how do we access them?

[[‘Squishy Banana’, 20000, 1],
 [‘Unicorn cushion’, 8000, 23.7],
 [‘Sushi Roller’, 5000, 8]]

[{Product: ‘Squishy Banana’, Quantity Sold: 20000, Price: 1},
 {Product: ‘Unicorn cushion’, Quantity Sold: 8000, Price: 23.7},
 [Product: ‘Sushi Roller’, Quantity Sold: 5000, Price: 8}]

LET’S EXAMINE WITH AN EXAMPLE

SINGLE LAYER LIST

```
Data = [10, 20, 30]
```

Each iteration, one element separated by a comma will by stored in the arbitrary variable i, which in this case are the numbers

```
for i in data:

    print(i)
```

NESTED LIST

```
Data =
[[‘Squishy Banana’, 20000, 1],
 [‘Unicorn cushion’, 8000, 23.7],
 [‘Sushi Roller’, 5000, 8]]
```

Likewise, each iteration, one element separated by a comma will by stored in the arbitrary variable i, which in this case are the individual list

```
for i in data:

    print(i[1])
```

This is why when I put [1] beside the i, which is a list, it does a further drill down to extract the second element of each list!!

NESTED LOOPS

- Notion of Nested for Loops
- Purpose of Nested Loops
- Execution Trace

NOTION OF NESTED FOR LOOPS

- All programming languages allows for loops within loops
- The process continues until all the nested lists have been iterated through

PURPOSE OF NESTED LOOPS

- With only a single loop you can only access a row of data, or one particular data from the row at a time. Let’s take a look at some examples:

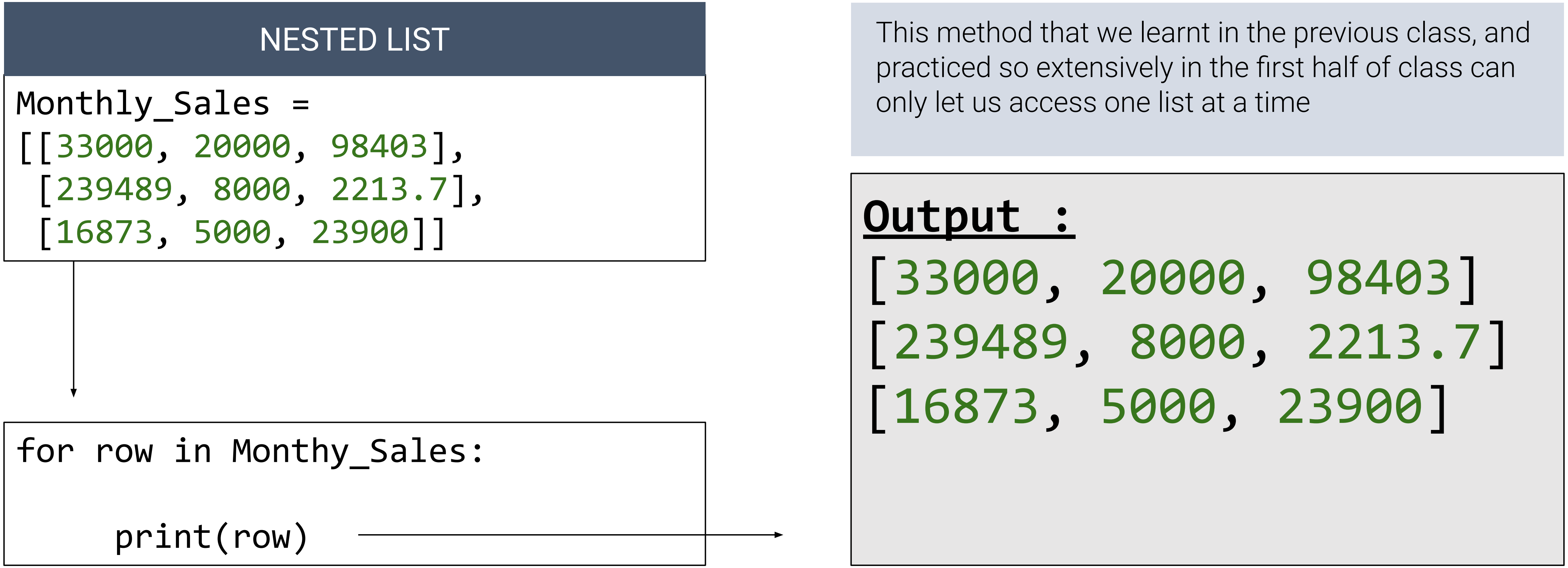
NESTED LIST

```
Monthly_Sales =
[[33000, 20000, 98403],
 [239489, 8000, 2213.7],
 [16873, 5000, 23900]]
```

Imagine if I have the monthly sales of heavy machineries of a company, with each list representing a month’s sales, and each entry in the list being the sales price of a machine. How do I write a code to assess the number of instances when the sales price is >20000?

PURPOSE OF NESTED LOOPS

- With only a single loop you can only access a row of data, or one particular data from the row at a time. Let’s take a look at some examples:



NESTED LIST

```
Monthly_Sales =
[[33000, 20000, 98403],
 [239489, 8000, 2213.7],
 [16873, 5000, 23900]]
```

```
for row in Monthy_Sales:

     print(row)
```

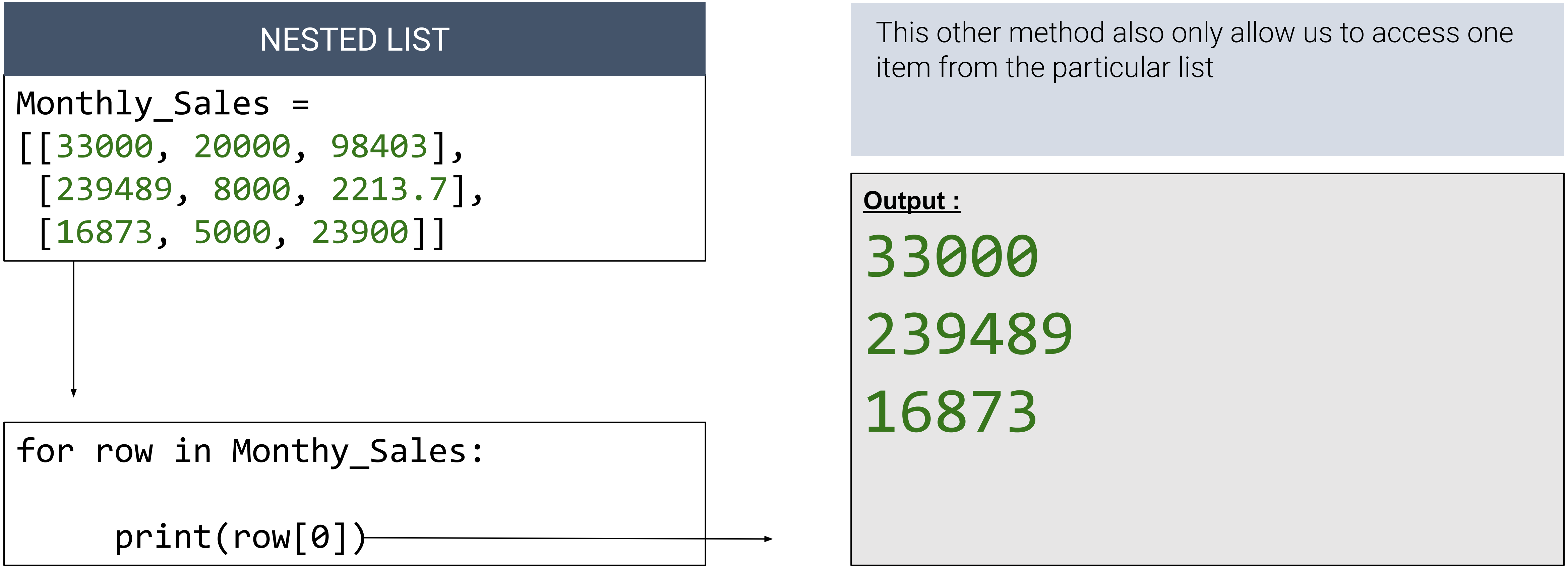
This method that we learnt in the previous class, and practiced so extensively in the first half of class can only let us access one list at a time

Output :

```
[33000, 20000, 98403]
[239489, 8000, 2213.7]
[16873, 5000, 23900]
```

PURPOSE OF NESTED LOOPS

- With only a single loop you can only access a row of data, or one particular data from the row at a time. Let’s take a look at some examples:



NESTED LIST

```
Monthly_Sales =
[[33000, 20000, 98403],
 [239489, 8000, 2213.7],
 [16873, 5000, 23900]]
```

```
for row in Monthy_Sales:

    print(row[0])
```

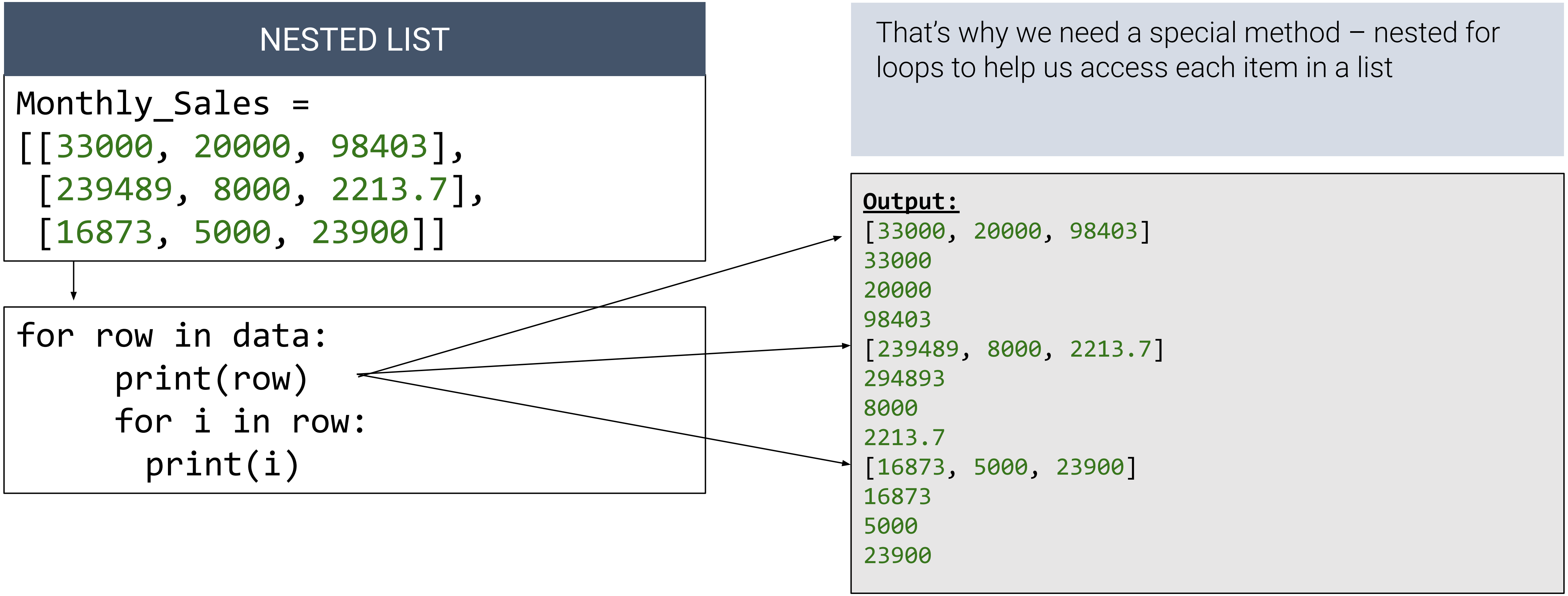
This other method also only allow us to access one item from the particular list

Output :

```
33000
239489
16873
```

PURPOSE OF NESTED LOOPS

- With only a single loop you can only access a row of data, or one particular data from the row at a time. Let’s take a look at some examples:



NESTED LIST

```
Monthly_Sales =
[[33000, 20000, 98403],
 [239489, 8000, 2213.7],
 [16873, 5000, 23900]]
```

```
for row in data:
     print(row)
     for i in row:
      print(i)
```

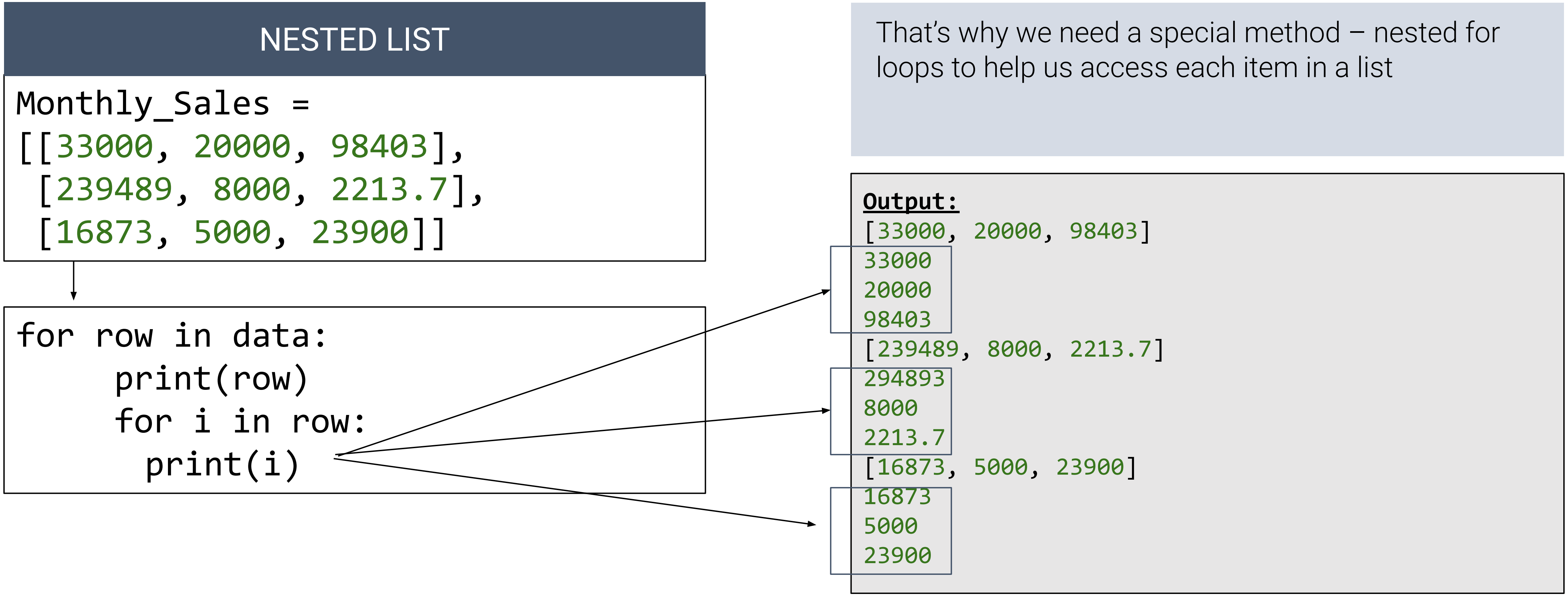
That’s why we need a special method – nested for loops to help us access each item in a list

Output:

```
[33000, 20000, 98403]
33000
20000
98403
[239489, 8000, 2213.7]
294893
8000
2213.7
[16873, 5000, 23900]
16873
5000
23900
```

PURPOSE OF NESTED LOOPS

- With only a single loop you can only access a row of data, or one particular data from the row at a time. Let's take a look at some examples:



NESTED LIST

```
Monthly_Sales =
[[33000, 20000, 98403],
 [239489, 8000, 2213.7],
 [16873, 5000, 23900]]
```

```
for row in data:
     print(row)
     for i in row:
       print(i)
```

That's why we need a special method – nested for loops to help us access each item in a list

Output:

```
[33000, 20000, 98403]
33000
20000
98403
[239489, 8000, 2213.7]
294893
8000
2213.7
[16873, 5000, 23900]
16873
5000
23900
```

PURPOSE OF NESTED LOOPS

- Back to the original question of assessing number of instances where sales price is > 20000

```
count = 0

for row in data:
    for i in row:
        if i > 20000:
            count += 1

print(count)
```

LET’S LOOK AT ANOTHER EXAMPLE

- Let’s take a look at another example in detail to see how the nested loop works

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

1

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

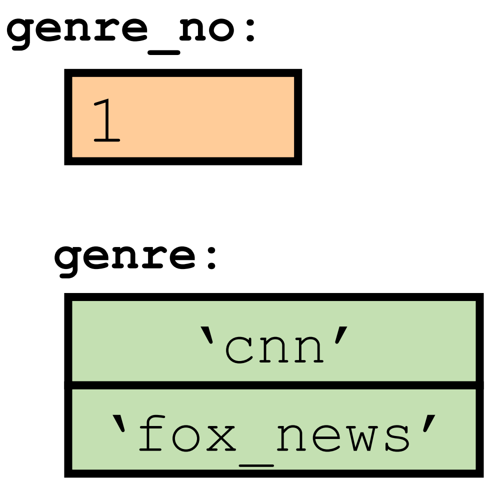
genre_no:

1

2

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

1

genre:

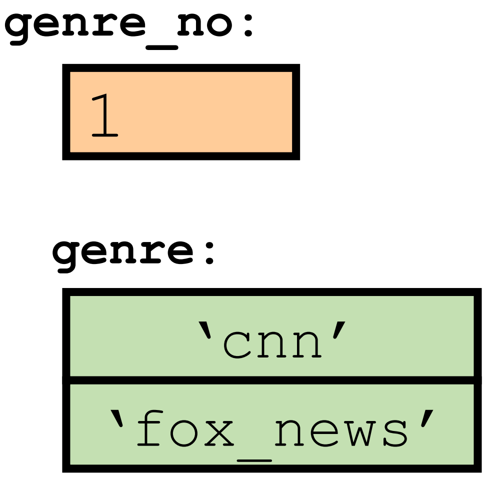
‘cnn’

‘fox_news’

3

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



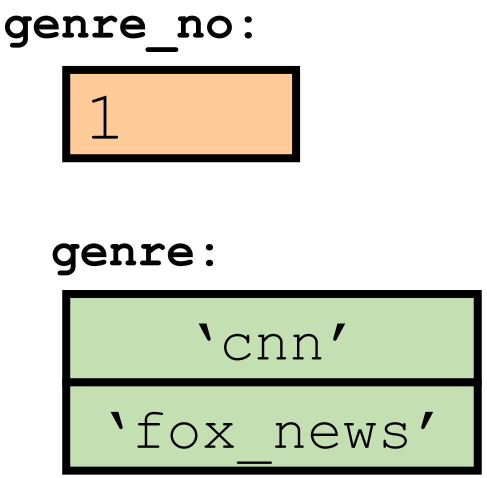
Output:

```
channels in genre 1:
```

4

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

1

genre:

'cnn'

'fox_news'

Output:

```
channels in genre 1:
```

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

genre_no:

1

genre:

‘cnn’

‘fox_news’

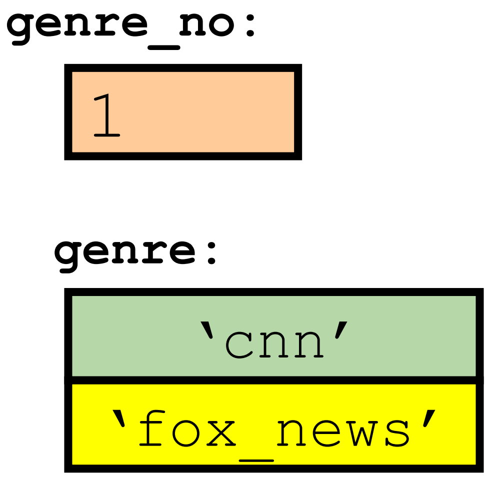
Output:

```
channels in genre 1:
cnn
```

6

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

1

genre:

‘cnn’

‘fox_news’

Output:

```
channels in genre 1:
cnn
```

7

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

genre_no:

1

genre:

‘cnn’
‘fox_news’

Output:

```
channels in genre 1:
cnn
fox_news
```

8

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

genre_no:

2

genre:

‘cnn’

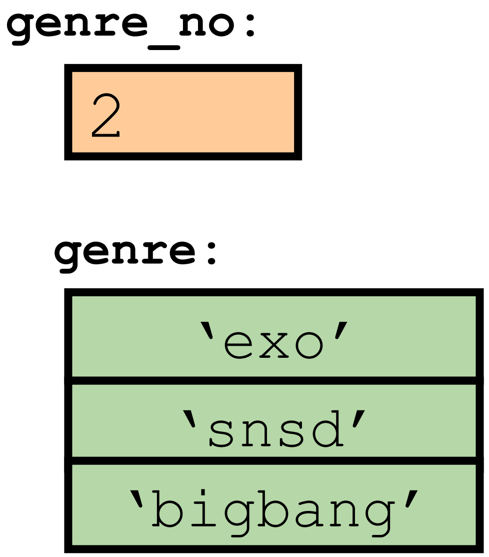
‘fox_news’

```
Output:
channels in genre 1:
cnn
fox_news
```

9

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

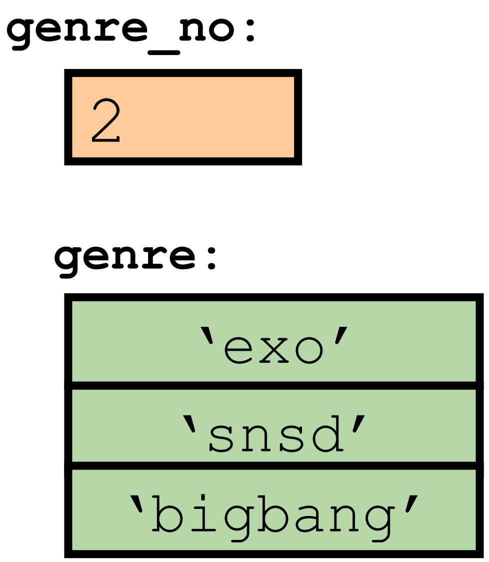


Output:

```
channels in genre 1:
cnn
fox_news
```

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

2

genre:

‘exo’

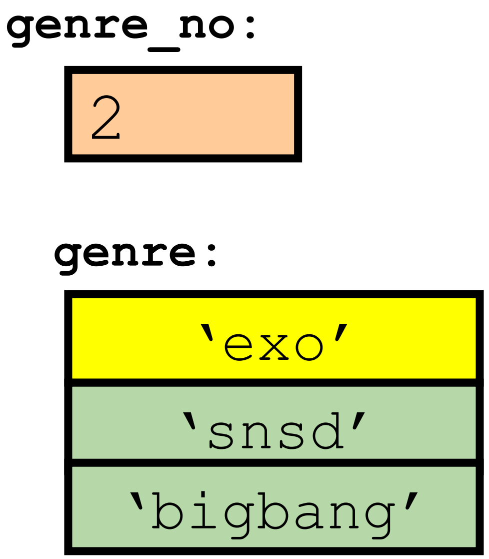
‘snsd’

‘bigbang’

```
Output:
channels in genre 1:
cnn
fox_news
channels in genre 2:
```

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
```

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

genre_no:

2

genre:

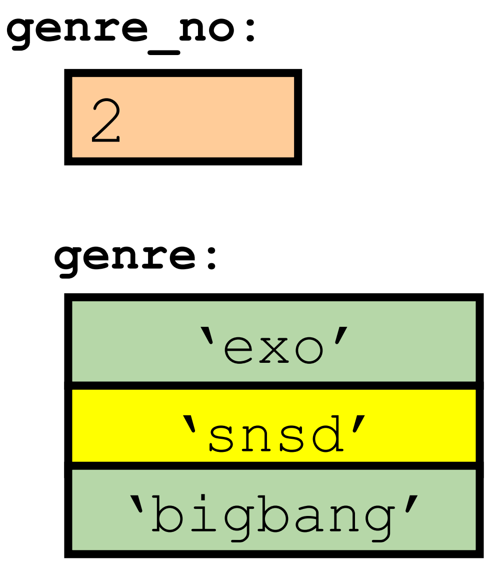
'exo'
'snsd'
'bigbang'

Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
exo
```

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

2

genre:

'exo'

'snsd'

'bigbang'

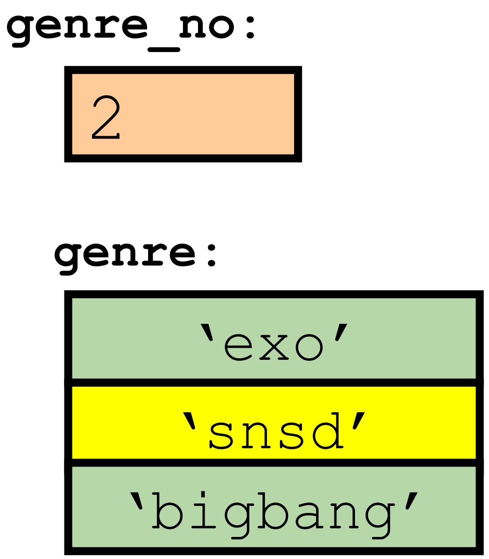
Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
exo
```

14

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

2

genre:

‘exo’

‘snsd’

‘bigbang’

Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
exo
snsd
```

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```

genre_no:

2

genre:

‘exo’

‘snsd’

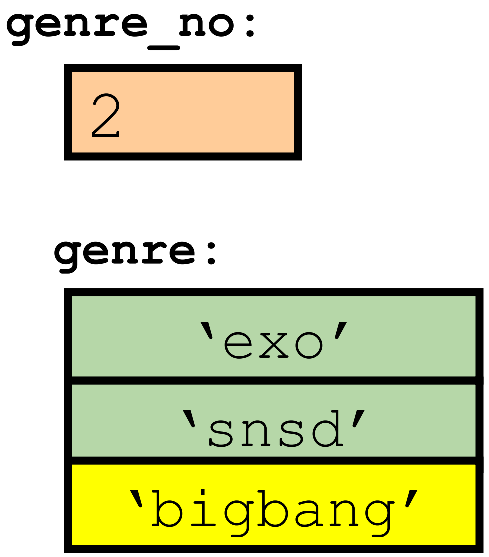
‘bigbang’

Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
exo
snsd
```

EXECUTION TRACE: NESTED FOR LOOP

```python
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

2

genre:

‘exo’

‘snsd’

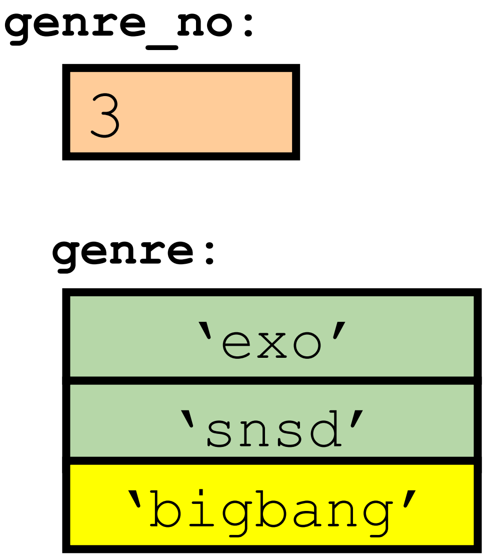
‘bigbang’

Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
exo
snsd
bigbang
```

EXECUTION TRACE: NESTED FOR LOOP

```
genre_channels = [['cnn','fox_news'],['exo','snsd','bigbang']]
genre_no = 1
for genre in genre_channels:
    print('channels in genre' + str(genre_no) + ':')
    for channel in genre:
        print(channel)
    genre_no += 1
```



genre_no:

3

genre:

'exo'

'snsd'

'bigbang'

Output:

```
channels in genre 1:
cnn
fox_news
channels in genre 2:
exo
snsd
bigbang
```

IN-CLASS PRACTICE: NESTED LOOPS WITH CONDITIONS*

- Try out the in class practice questions!

APPLIED ITERATIONS

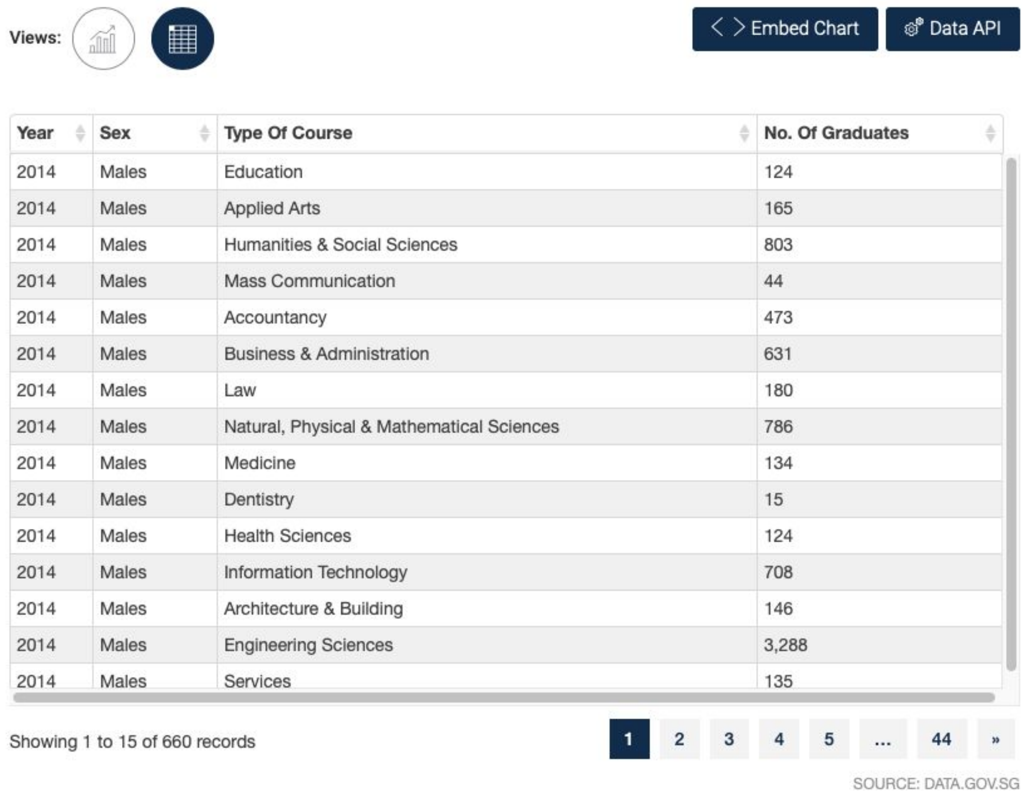
- Applying iterations on real data set
- Asking the right questions
- CITU Framework

APPLIED ITERATIONS

- When dealing with large amounts of data, iterations provide us with the necessary tools to go through each row of data one by one.
- Doing so allows us to answer some basic analytics questions about the data.
- To approach solving these questions, we have to follow a structure which will discuss later.
- Take the following dataset from Data.gov.sg as an example.

APPLIED ITERATIONS

- This dataset is from the Singapore Government's Open Data Portal.
- It is a breakdown of the Graduates from University First Degree Courses By Type of Course.



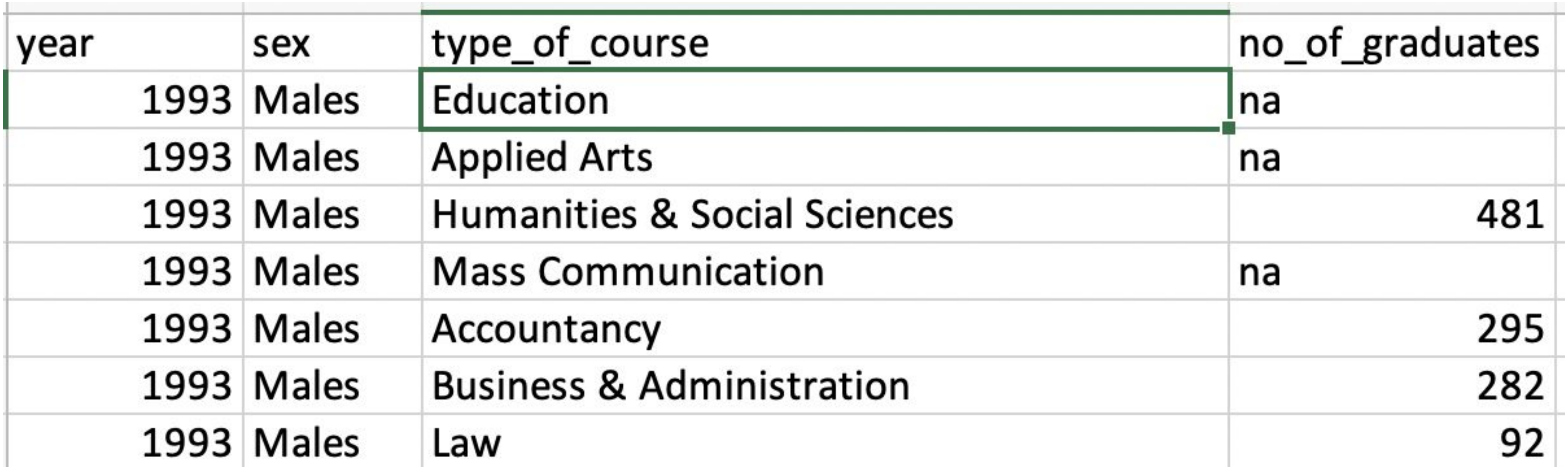
Year	Sex	Type Of Course	No. Of Graduates
2014	Males	Education	124
2014	Males	Applied Arts	165
2014	Males	Humanities & Social Sciences	803
2014	Males	Mass Communication	44
2014	Males	Accountancy	473
2014	Males	Business & Administration	631
2014	Males	Law	180
2014	Males	Natural, Physical & Mathematical Sciences	786
2014	Males	Medicine	134
2014	Males	Dentistry	15
2014	Males	Health Sciences	124
2014	Males	Information Technology	708
2014	Males	Architecture & Building	146
2014	Males	Engineering Sciences	3,288
2014	Males	Services	135

Showing 1 to 15 of 660 records

SOURCE: DATA.GOV.SG

APPLIED ITERATIONS

- To analyse this data in a Python sense, let’s break this down to something which we are familiar with: **Nested Lists**.



year	sex	type_of_course	no_of_graduates
1993	Males	Education	na
1993	Males	Applied Arts	na
1993	Males	Humanities & Social Sciences	481
1993	Males	Mass Communication	na
1993	Males	Accountancy	295
1993	Males	Business & Administration	282
1993	Males	Law	92

APPLIED ITERATIONS

```
[
       [ # Row 0
           1993, # Year - Index 0
           'Males',  # Sex - Index 1
           'Education',  # Type of Course - Index 2
           0 # No of Graduates - Index 3
       ],
       [ # Row 1
           1993, # Year - Index 0
           'Males', # Sex - Index 1
           'Applied Arts',  # Type of Course - Index 2
            0 # No of Graduates - Index 3
       ],
       [ # Row 2
           1993, # Year - Index 0
           'Males', # Sex - Index 1
           'Humanities & Social Sciences',  # Type of Course - Index 2
           481 # No of Graduates - Index 3
       ]
]
```

APPLIED ITERATIONS

- One simple descriptive analytics question you can ask is:
  - How many students have taken `'Education'`?

- To answer this question, we must do the following:
  - Ask what variables do you need to solve this problem?
  - Apply the CITU Framework

APPLIED ITERATIONS

- What variables do we need for this question?
  - Type of Course
  - No. of Graduates
- After knowing what variables you need, apply the CITU Framework
  - **Create** the result container
  - **Loop** each row of data
  - From each row, **take out** *Type of Course* and *No. of Graduates*
  - **Test** the variables to see if they match the conditions
  - **Update** the result container

APPLIED ITERATIONS

```
# 1. Create results container
education_students = 0

# 2. Loop data
for current_course in all_courses:
    # 3. Take out the variables you need (IMPORTANT)
    course_type = current_course[2]
    no_of_students = current_course[3]

    # 4. Test the variables with condition
    if course_type == 'Education':
        # 5. Update the results container
        education_students += no_of_students
```

IN-CLASS PRACTICE: APPLIED ITERATIONS*

- Using the example earlier, answer the following 2 analytics questions in your in class practice:
    - How many `'Females'` students have taken `'Law'`?
    - How many students have taken `'Information Technology'` between `2000` and `2014`?

AGGREGATION WITH DICTIONARIES

- Counting Algorithm
- Dictionary as a database

COUNTING WITH DICTIONARIES

- Dictionary is a versatile data type that allows us to do aggregation of data, i.e. counting items.
- Take for example the given list

```
['Pikachu','Charmander','Pikachu','Charmander']
```

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokedex_dictionary:

```
{


}
```

1

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Pikachu'
```

pokedex_dictionary:

```
{

}
```

2

COUNTING WITH DICTIONARIES: CODE TRACE

```python
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

True

pokemon:

```
'Pikachu'
```

pokedex_dictionary:

```
{

}
```

3

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Pikachu'
```

pokedex_dictionary:

```
{
    'Pikachu': 1
}
```

4

COUNTING WITH DICTIONARIES: CODE TRACE

```python
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Charmander'
```

pokedex_dictionary:

```
{
    'Pikachu': 1


}
```

5

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

True

pokemon:

```
'Charmander'
```

pokedex_dictionary:

```
{
    'Pikachu': 1
}
```

6

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Charmander'
```

pokedex_dictionary:

```
{
    'Pikachu': 1 ,


    'Charmander': 1


}
```

7

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Pikachu'
```

pokedex_dictionary:

```
{
    'Pikachu': 1 ,


    'Charmander': 1


}
```

8

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

False

pokemon:

```
'Pikachu'
```

pokedex_dictionary:

```
{
    'Pikachu': 1 ,


    'Charmander': 1


}
```

9

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Pikachu'
```

pokedex_dictionary:

```
{
    'Pikachu': 2 ,


    'Charmander': 1


}
```

10

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Charmander'
```

pokedex_dictionary:

```
{
    'Pikachu': 2 ,


    'Charmander': 1


}
```

11

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

False

pokemon:

```
'Charmander'
```

pokedex_dictionary:

```
{
    'Pikachu': 2 ,


    'Charmander': 1


}
```

12

COUNTING WITH DICTIONARIES: CODE TRACE

```
caught = ['Pikachu','Charmander','Pikachu','Charmander']

# 1. Create results container
pokedex_dictionary = {}

# 2. Loop data
for pokemon in caught:
    # 3. Test the variables with condition
    if pokemon not in pokedex_dictionary:
        # 4. Create new key-value pair with value of 1
        pokedex_dictionary[pokemon] = 1
    else:
        # 5. Increase count by 1
        pokedex_dictionary[pokemon] += 1
```

pokemon:

```
'Charmander'
```

pokedex_dictionary:

```
{
    'Pikachu': 2 ,


    'Charmander': 2


}
```

13

IN-CLASS PRACTICE: COUNTING WITH DICTIONARIES

- Try out the in class practice questions!

SUMMARY

- Accessing data, and interacting with them in a nested collections format
- Iterations applied to a real dataset
- Aggregation with Dictionaries