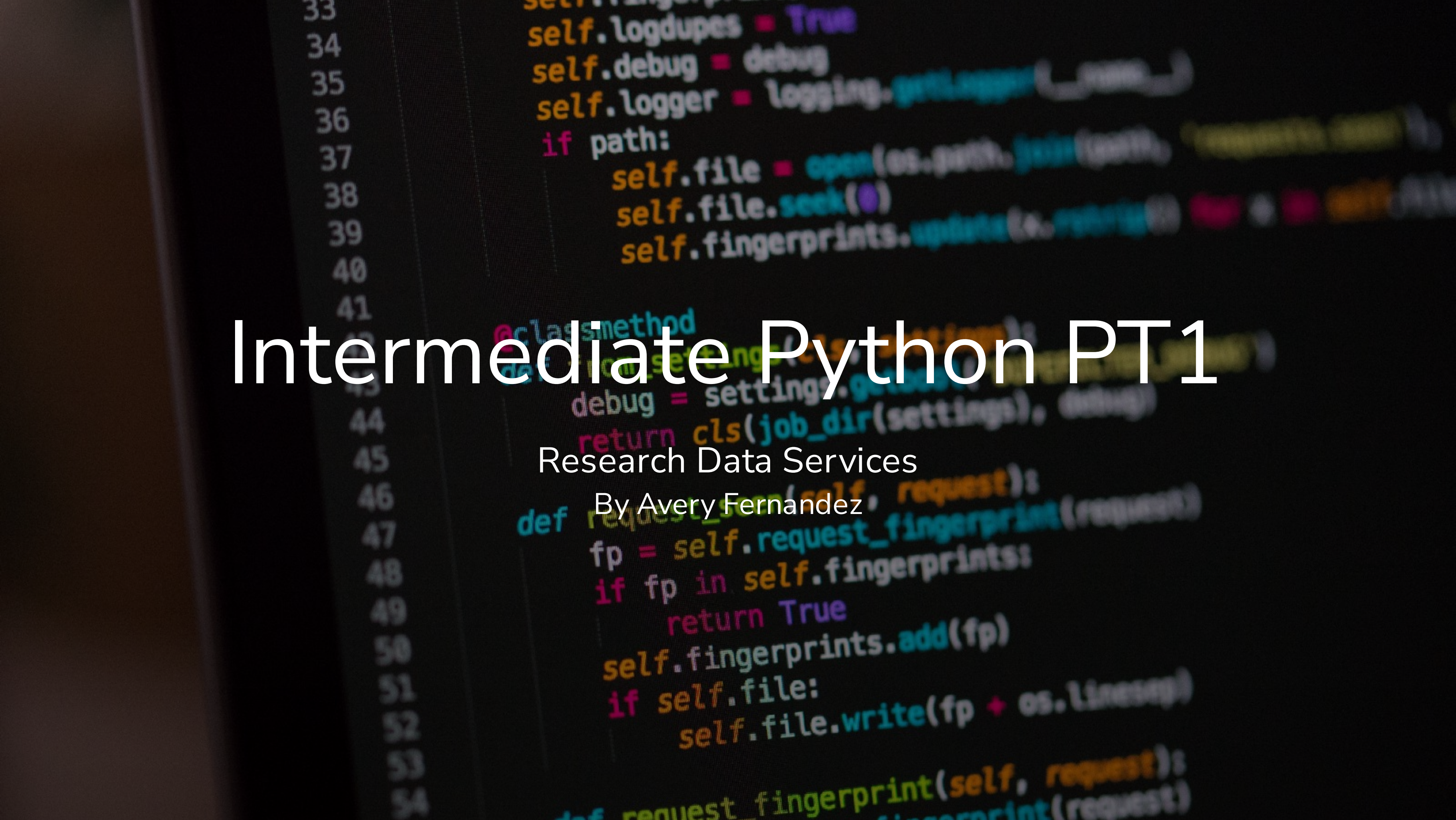


Intermediate Python PT1

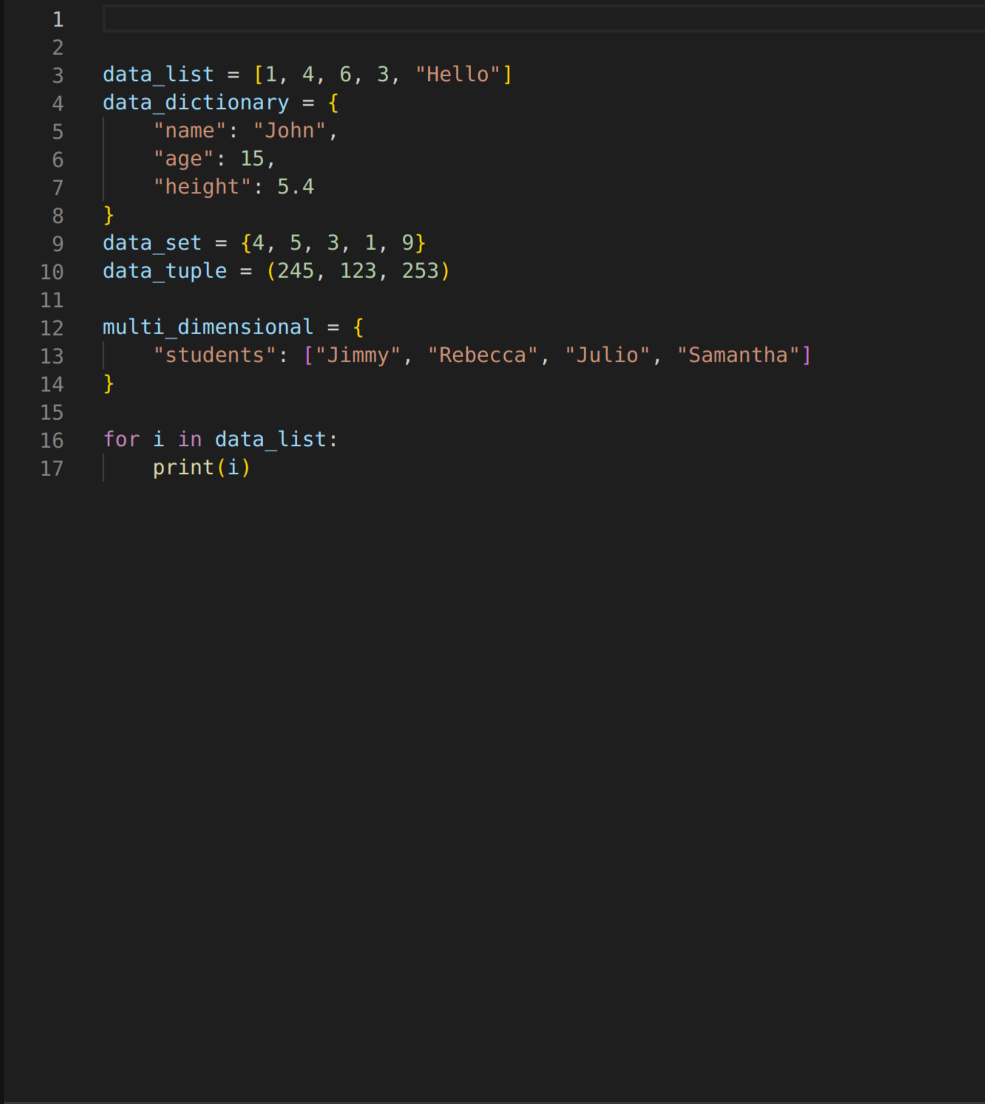
Research Data Services

By Avery Fernandez

Plans For Today

Learning the basics to python

- **Data Structures** storing multiple values in one
- **Multi-dimensional Structures** inception!



```python


data_list = [1, 4, 6, 3, "Hello"]
data_dictionary = {
    "name": "John",
    "age": 15,
    "height": 5.4
}
data_set = {4, 5, 3, 1, 9}
data_tuple = (245, 123, 253)

multi_dimensional = {
    "students": ["Jimmy", "Rebecca", "Julio", "Samantha"]
}

for i in data_list:
    print(i)
```

Why do we need data structures?

No need to duplicate names for similar things

Imagine creating variables for students

```
student1 = "John"
student2 = "Jaiden"
student3 = "Sarah"
student4 = "Jennifer"
student5 = "Ryan"
```

Doing this for 20+ students would be impossible

So what if we could store all these things in one variable?

Types of Data Structures

Let’s talk about the main ones we will use

There are 4 main data structures:

1. Lists
2. Dictionaries
3. Sets
4. Tuples

What makes them special is that they can store any basic data types

Lists

- It is ordered
- Can have duplicates
- It is changeable

Tuples

- It is ordered
- Can have duplicates
- It is not changeable

Dictionaries

- Indexing is done with the keys
- It is ordered
- Can't have duplicates
- It is changeable

Sets

- Cannot be indexed
- It is unordered
- Can't have duplicates
- It is unchangeable

Lists

Lists are a variable that can store multiple items

Denoted by being surrounded by **[ ]**

Syntax:

```
myList = [ ]
```

Our First List

```
myList = ["apple", "orange", "apple", "banana"]
```

Items in the lists are separated by ,

Indexing

Extracting specific data

Index starts with 0

We can do 3 main things with indexing:

- Regular Indexing
- Negative Indexing
- Ranges

Indexing Our First List

Some basic syntax

Printing the First Item

```
print(myList[0])
```

Last Item

```
print(myList[-1])
```

Ranges

```
print(myList[1:3])
print(myList[1:-1])
```

Output

```
banana
```

```
apple
```

```
['orange', 'apple']
['orange', 'apple']
```

Basic List Functions

Some common functions for lists

len(), outputs size of list

```
len(myList)
```

append(), adds to end of list

```
myList.append("banana")
```

sort(), sorts list

```
myList.sort()
```

Changing values

```
myList[0] = "grape"
```

Exercise One

Using Lists

Create a list of with:

- at least 5 vegetables
- sort it in alphabetical order
- print out the third item

Exercise One Code

```python
vegetables = ["broccoli", "caper", "cauliflower", "squash", "avocado"]
vegetables.sort()
print(vegetables)
```

Dictionaries

Dictionaries use key and value pairs

Syntax:

```
myDictionary = {key: value}
```

Denoted by **{ }**

First Dictionary

```
myDictionary = {
  "name": "John",
  "age": 25,
  "height":5.123,
  "adult":False
}
```

```
print(myDictionary[0])
```

Output

```
KeyError: 0
```

```
print(myDictionary["age"])
```

Output

```
25
```

Indexing

Can’t use the same indexing as Lists

As we saw above, dictionaries are not indexed with index values, but with their keys

There are 3 functions that are useful for looking at the data stored in dictionaries:

- keys()
- values()
- items()

Indexing Functions

Using those functons

Keys

```
keys = myDictionary.keys()
print(keys)
```

Output

```
dict_keys(['name', 'age', 'height', 'adult'])
```

Values & Items

```
values = myDictionary.values()
print(values)
items = myDictionary.items()
print(items)
```

Output

```
dict_values(['John', 25, 5.123, False])
dict_items([('name', 'John'), ('age', 25), ('height', 5.123), ('adult', False)])
```

Getting and Changing Dictionary Values

The Key is the Key

We can use the dictonary keys to grab out their individual values

```
print(myDictionary["name"])
```

Output

```
John
```

```
myDictionary["age"] = 34
print(myDictionary)
```

Output

```
{'name': 'John', 'age': 34, 'height': 5.123, 'adult': False}
```

Exercise 2

Using Dictionaries

- Create a dictionary 5 food types and their foods like soda -> sprite.
- Print one of your values using key indexing.

Exercise 2 Code

```
foods = {
  "soda": "Dr.Pepper",
  "chips": "Talkis",
  "fruit": "Strawberry",
  "soup": "Tomato",
  "salad": "Fruit"
}
print(foods["salad"])
```

Tuples

A storage type that stores preset values

Syntax:

```
myTuple = ()
```

Tuple Examples

```
my_tuple = (1, 3, 5)
print(my_tuple)
my_tuple[0] = 10
```

Output

```
(1, 3, 5)
TypeError: 'tuple' object does not support item assignment
```

There is an error due to trying to change a value, which tuples don't support

Sets

One use-case is in removing duplicates of data

Syntax:

```
sets = {}
```

Sets Example

```python
sets = {"apple", "banana", "banana", "orange"}
print(sets)
```

Output

```
{'apple', 'banana', 'orange'}
```

The extra banana gets removed

Multi-dimensional Storage Types

Inception

While it sounds complicated, it is just put storage types in other storage types:

- Dictionaries in Lists
- Lists in Dictionaries
- Etc.

There is no special syntax, just use the syntax of dictionaries, lists, sets, and tuples

Multi-Dimensional Examples

```
school = {
    "grades": {
        "freshmen": ["Jim", "Abby", "Paul"],
        "sophomores": ["Liv", "Trin", "Feliza"],
        "juniors": ["Avery", "Michael", "Cyrus"],
        "seniors": ["Gabby", "Maddie", "Jordan", "Drew"]
    }
}
```

Exercise 3

Using everything

- Make a multi-dimensional data structure for your weekly schedule
- Take a user input of a day of the week i.e. Monday, Tuesday...
- Output your schedule for the day given by the user

Exercise 3 Code

```
weekly_schedule = {
    "monday": [{
            "event": "MATH 301",
            "time": "10-11 AM"},
        {
            "event": "ECE 383",
            "time": "12-1 PM"}],
    "tuesday": [{
            "event": "CS 200",
            "time": "11-12:30 PM"}],
    "wednesday": [{
            "event": "MATH 301",
            "time": "10-11 AM"},
        {
            "event": "ECE 383",
            "time": "12-1 PM"}],
    "thursday": [{
            "event": "CS 200",
            "time": "11-12:30 PM"}],
    "friday": [{
            "event": "MATH 301",
            "time": "10-11 AM"},
        {
            "event": "ECE 383",
            "time": "12-1 PM"}]
```

Extended

```
day = input("Enter a day: i.e. monday")
print(weekly_schedule[day])
```

Output

```
Enter a day i.e. monday:
tuesday
[{'event': 'CS 200', 'time': '11-12:30 PM'}]
```