Lesson 4

Python Essentials

Overview

1. Lists
2. Arrays
3. Slicing
4. Tuples
5. Sets

Lists

- Lists are collections of items.
- Lists can be expanded or contracted as needed.
- Can contain any data type.
- Commonly used to store a single column collection of information, however it is possible to nest lists.

```
empty_list = []
empty_list = list()

names = ['James', 'David']
scores = []
scores.append(98) # Add new item to the end
scores.append(99)

print(names) # ['Christopher', 'Susan']
print(scores) # [98,99]
print(scores[1]) # 99
```

Arrays

- Arrays are collections of items.

- Designed to store a uniform basic data type, such as integers or floating point numbers.
- Use array module

```
from array import array
scores = array('d')
scores.append(97)
scores.append(98)
print(scores)
print(scores[1])
```

Notes: - from array library - Later module will talk about libraries and modules and packages. - Basically we are importing an array

Lists Vs. Arrays

- Arrays:
  - Simple types such as numbers
  - Must all be the same type
- Lists:
  - Store anything
  - Store any type

Notes: - numpy will give you additional support

Common Operations

```
names = ['James', 'David']
print(len(names)) # Get the number of items
names.insert(0, 'Bill') # Insert before index
print(names)
names.sort()
print(names)
```

```
2
['Bill', 'James', 'David']
['Bill', 'David', 'James']
```

sorts side effect is that it will modify the list

Slicing

```
names = ['James', 'David', 'Bill','Justin']
names
names[3]
names[1:3]
names[:3]
```

```
['James', 'David', 'Bill','Justin']
['Justin']
['David', 'Bill']
['James', 'David', 'Bill']
```

Dictionaries

- Dictionaries are key/value pairs of a collection of items.
- Dictionaries use keys to identify each item.

```
empty_dictionary = {}
empty_dictionary = dict()

person = {'first': 'John'}
person['last'] = 'Doe'

print(person)
print(person['first'])
```

```
{'first': 'John', 'last': 'Doe'}
John
```

Dictionaries Vs Lists

- Dictionaries:
  - Key/Value pairs
  - Storage order not guaranteed
- Lists:
  - Zero-based index
  - Storage order guaranteed

Tuples

- Create a tuple

```
empty_tuple = ()
empty_tuple = tuple()

tup = ('32', 4, 'yes', 3.14)
```

- Similar to lists:

```
tup[1:4] # (4, 'yes', 3.14)
```

- Tuples are immutable

Notes: https://docs.python.org/2/library/functions.html#tuple

Sets

- Sets are unordered.
- Set elements are unique. Duplicate elements are not allowed.
- A set itself may be modified, but the elements contained in the set must be of an immutable type.
- Common operations
  - union
  - intersect
  - difference

```
empty_set = set()

set1 = {1, 5, 10, 15, 20}
set2 = {2, 5, 11, 15, 21}
```

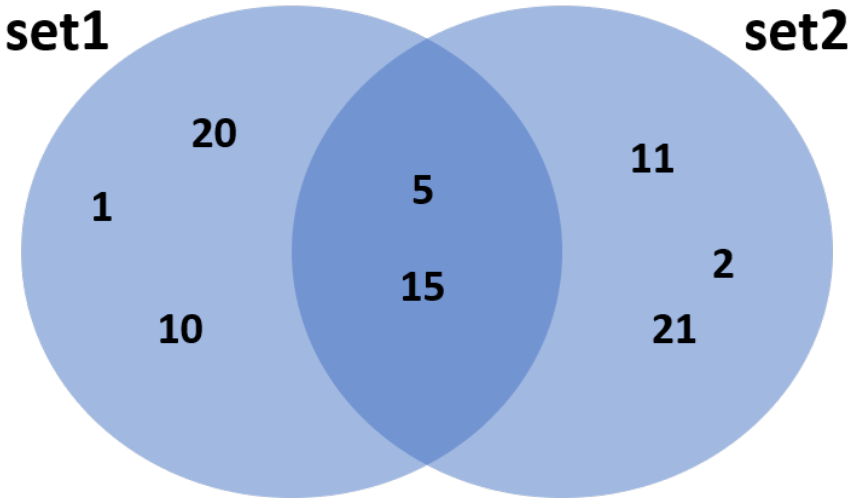

Figure 1: image