JFSS Data Science Club

PYTHON 101

Meeting #3

TABLE OF CONTENTS

Data Structures:

- Lists
- Dictionaries
- Tuples
- Sets

WHY DO WE NEED DATA STRUCTURES FOR DATA SCIENCE?

WHY DATA STRUCTURES?

- Organize + manage data
- Fast + easy data retrieval
- Flexibility with the type of data
- Represent data in desired ways (visualization, etc.)
- Implement Algorithms

REVIEW: DATA TYPES

String

Integer

```
1  name = "Jeremy"
2
3  age = 12
4
5  likes_pineapple_on_pizza = False
6
7  weight_in_lbs = 92.7
```

Boolean

Float

LISTS

```python
nums = list() # creates an empty list

# creates a list with specific items
names = ["Prerana", "Arya", "Grace"]

# Printing the lists
print("nums: ", nums)
print("names: ", names)
```

2 ways of creating a list

```
nums:  []
names:  ['Prerana', 'Arya', 'Grace']
```

LISTS

How indexing works in a list

```
                0   1   2   3   4   5
4   ages = [23, 12, 45, 67, 23, 18]
               -6  -5  -4  -3  -2  -1
```

LISTS

Accessing items in a list using their index

```
planets = ["Earth", "Mercury", "Jupiter"]
mystery_planet = planets[1]
print(planets)
```

What is the mystery planet?

LISTS

Accessing items in a list using their index

```
planets = ["Earth", "Mercury", "Jupiter"]
mystery_planet = planets[1]
print(planets)
```

What is the mystery planet?

```
mystery planet:  Mercury
```

LISTS

Accessing multiple items in a list using a range

```
15   ages = [23, 12, 45, 67, 23, 18]
16   print(ages[2:5])    # prints 45 67 23
```

```
[45, 67, 23]
```

LISTS

```python
ages = [23, 12, 45, 67, 23, 18]
# prints how many items are in the list
print(len(ages))
```

Printing the number of items in the list

Output: 6

LISTS

Having multiple data types in one list

```
# list can have various data types
miscellaneous = [True, "Jack", 23.0, 12,
45.345, 67]
```

LISTS

Adding items to a list

```
18  print(ages)
19  ages.append(78)
20  print(ages)
```

```
[23, 12, 45, 67, 23, 18]
[23, 12, 45, 67, 23, 18, 78]
```

Element 78 has been added to the list

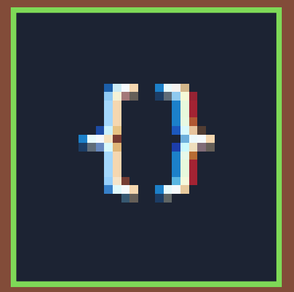
```
#inserts in the given index
ages.insert(0, "hi")
print(ages)
```

```
[23, 12, 45, 67, 23, 18]
[23, 12, 45, 67, 23, 18, 78]
['hi', 23, 12, 45, 67, 23, 18, 78]
```

DICTIONARIES

Creating a dictionary

```
foods = dict()
print(foods)
```



DICTIONARIES

Creating a dictionary

```
foods = dict({
  "breakfast": "Cereal",
   "lunch": "Sandwich",
   "dinner": "Pasta"
})
print(foods)
```

```
{'breakfast': 'Cereal', 'lunch': 'Sandwich', 'dinner': 'Pasta'}
```

DICTIONARIES

Store data value in key:value pairs

```
26 days_of_the_week = {
27     1: "Monday",
28     2: "Tuesday",
29     3: "Wednesday",
30     4: "Thursday",
31     5: "Friday",
32     6: "Saturday",
33     7: "Sunday"
34 }
```

```
36 print(days_of_the_week)
```

```
{1: 'Monday', 2: 'Tuesday', 3: 'Wednesday', 4: 'T
hursday', 5: 'Friday', 6: 'Saturday', 7: 'Sunday'
}
```

```
38 print(days_of_the_week[4])
```

```
Thursday
```

DICTIONARIES

Retrieve the keys and values in a dictionary

```
print(days_of_the_week.keys())
```

```
dict_keys([1, 2, 3, 4, 5, 6, 7])
```

```
print(days_of_the_week.values())
```

```
dict_values(['Monday', 'Tuesday', 'Wednesday', 'T
hursday', 'Friday', 'Saturday', 'Sunday'])
```

DICTIONARIES

Changing values by key

```
days_of_the_week[3] = "Thursday"
```

```
{1: 'Monday', 2: 'Tuesday', 3: 'Thursday', 4: 'Thursday', 5: 'Friday', 6: 'Saturday', 7
: 'Sunday'}
```

Adding a new key-value pair to a dictionary

```
days_of_the_week[8] = "Funday"
print(days_of_the_week)
```

```
{1: 'Monday', 2: 'Tuesday', 3: 'Thursday', 4: 'Thursday', 5: 'Friday', 6: 'S
aturday', 7: 'Sunday', 8: 'Funday'}
```

TUPLES

Immutable: cannot change once created

```
# Creating a tuple
monthsTuple = ("Januray", "February",
"March", "April")
print(monthsTuple)
```

```
('Januray', 'February', 'March', 'April')
```

TUPLES

```python
# List of tuples
technological_timeline = [
  ("fire", "1M years ago"),
  ("Irrigation", "6000BCE"),
  ("Steam engine", 1765),
  ("Internet", 1974)
]
print(technological_timeline)
```

List of tuples

- Would not want the order to change

```
[('fire', '1M years ago'), ('Irrigation', '6000BCE'), ('
Steam engine', 1765), ('Internet', 1974)]
```

SETS

A set is a collection which is unordered, unchangeable, and unindexed data

CANNOT USE INDEX!

How is it different from Lists?

SETS

```
num_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
print(num_set)
```

- Doesn't change the order or the item

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

SETS

```
fruits = {"apple", "banana", "cherry",
"apple"}

print(fruits)
```

- **Sets cannot have two items with the same value**
  - Duplicate values will be ignored in a set

What do you think is the result?

SETS

```
fruits = {"apple", "banana", "cherry",
"apple"}

print(fruits)
```

- **Sets cannot have two items with the same value**
  - Duplicate values will be ignored in a set

```
{'apple', 'banana', 'cherry'}
```

SETS

```
positives = {"apple", "banana", "cherry",
True, 1, 2}

print(positives)

negatives = {"apple", "banana", "cherry",
False, True, 0}

print(negatives)
```

- True and 1 is considered the same value

```
{True, 2, 'banana', 'apple', 'cherry'}
```

- False and 0 is considered the same value

```
{False, True, 'cherry', 'banana', 'apple'}
```

LISTS

DICTIONARIES

TUPPLES

SETS

LISTS

- Mutable
- Uses index to find items
- Ordered
- Example: [1, 2, 3, 4, 5]

DICTIONARIES

- Key value pairs
- Mutable
- Uses keys to find items
- No duplicate keys
- Example: {1: “a”, 2: “b”, 3: “c”, 4: “d”, 5: “e”}

TUPPLES

- Immutable - predefined items and number of items
- Ordered
- Example: (1, 2, 3, 4, 5)

SETS

- Mutable
- No index
- Unordered
- no duplicates allowed
- Example: {1, 2, 3, 4, 5}

THANK YOU!

Any other questions?