course_content

top

W2: In-built Python
W3: Arrays and Hash Tables
W4: Stacks and Queues
W5: Linked Lists
W6: Sorting Algorithms
W8: Trees
W9: Graphs
W10: Heaps
W13: Recursion
W14: Greedy Algorithms

Data Structures and Algorithms

Python Collective Data Structures

Week 2

Class Quiz!

Time:
5 min

bit.ly/DSA1920Quiz2

Bonus qn at the end to help you make up for other questions or missed quizzes..

Time:
20 min

Presentation Time

- Each Group has 5 minutes to present
- Group 1 - Lists, Group 2- Dictionaries, Group 3 - Sets, Group 4 - Tuples.
  - What is this data structure and what can we do with it?
  - When should we use it? Which kind of situations?
  - What are its' limitations? When should we not use it?
- Please have questions - challenge each other!

Time:
10 min

Lists

- **Ordered, mutable** Python Data Structure
- Written with square brackets []
- Each element can by any object (even other collective data structures)
- Accessed and modified by position e.g. myList[4] returns the 5th element
- len(myList) returns the length of the list
- Very useful for applying with a for loop e.g. a list of strings

Time:
10 min

Dictionaries

- **Unordered, mutable** Python Data Structure
- Changed to be ordered in Python 3.6 onwards. Read more here
- Written with curly brackets {}
- Each item is a key-value pair
- Dictionary Keys must be **immutable types.** Values can be any type.
- Accessed and modified by key e.g. myDict[1] or myDict[“Kigali”]

Time:
10 min

Tuples

- **Ordered, Immutable** Python Data Structure
- Written with normal brackets () separated by commas
- Items can be any object (even collective data structures)
- Just two methods - count and index
- **Cannot** change items by index - must convert to list first
- (“hello”,) for a single item tuple - need comma!
- Variables within tuples can be mutable (and can be mutated!)

Time:
10 min

Sets

- **Unordered, mutable** Python Data Structure with **no duplication**
- Written with curly braces or set() function. Empty set needs to be set().
- **Elements** within sets must be **immutable**
- Can do mathematical operations like union, intersection, add, diff and even comparison operators e.g. s1 > s2 tells you if s2 is a subset of s1.
- **Frozen Sets** work just like sets but are themselves immutable.

Time:
10 min

Comparison of 4 Data Types

Data Types	Mutable	Ordered	Duplicates	Allowed Data Types
Lists	Yes	Yes	Yes	All
Dictionaries	Yes	No (Yes, in Py3.6 +)	Keys - No Values - Yes	Keys - Immutable Values - All
Tuples	No	Yes	Yes	All
Sets	Yes	No	No	Immutable

Time:
10 min

Ranges

- Ways of specifying arithmetic sequences over integers
  - range(n) - (1,2,3,.....n-1)
  - range(a,b) - (a,a+1,a+2,.....b-1)
  - range(a,b,k) - (a,a+k, a+2k, a+3k, a+4k, ...b-1) (Note: k can be negative)
- Very useful for looping as it is **significantly faster** than for loops and while loops. Doesn't have to create, delete and iterate variables as often.
- There is just one range object and you iterate through it until you've reached the max level.

Time:
10 min

Iterator

- Lists, Sets, Tuples, Tuples, Dictionaries, Strings and Range Objects are all iterables - you can convert them to iterators.
- Iter() will convert an iterable to an iterator
- You loop through an iterator using the next() function
- For-loops do this implicitly. They convert an iterable into an iterator type and then loop through them using the next() function.
- While loops are different - there is no clearly defined start and end point.

Time:
10 min

One-line Iteration

- [x+2 for x in [1,2,3]] - create list using one line for loop
- [x + 2 if x%2==1 else 0 for x in range(10)] - if else statement within for loop
- {x+3 for x in (1,2,3)} - create set instead from tuple
- tuple(x +3 for x in {1,2,3}) - create tuple from set
- print(key,value) for key,value in myDict.items() - using key,value in dictionary
- Module **itertools** is very good for more advanced iteration

Time:
1 min

Questions

Time:
3 min

Next Steps

1. Commit Week 1 Implementations (Deadline tomorrow midnight!)
2. Week 3 Readings (to be posted on Piazza)
3. Get Started on Week 2 Implementations