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 Data Types

Week 2

Class Quiz!

Time:
5 min

bit.ly/DSA1920Quiz1

Time:
3 min

In-Built Python - Exercise

- Get into groups of 3 - no laptops allowed
- List all of the in-built python data types that you know.
- Categorise whether they are ‘atomic’ or ‘collective’ data types

Time:
10 min

In-Built Python

- **Atomic** - integers, floats, complex numbers, booleans
- **Collective** - strings, dictionaries, lists, tuples, set, range, iterators
- There are many others that you can find here but we focus on the main ones
- type() - gives you the type of any object (even user-defined ones)
- You can often convert between types e.g. int('32') or str(1.823) or list('abc')
- Some types are **immutable** - any changes point to a different object
    - Int, Float, Complex Numbers, Booleans, Strings, Tuples, Range
- Others are **mutable** - change the object that the variable points to
    - List, Dictionary, Set

Time:
10 min

In-Built Python - Exercise 2

- Group 1 - Numbers (Ints, Floats, Complex)
- Group 2 - Booleans
- Group 3 - Strings

You have 10 minutes to find out as much as you can about your chosen data structure and its’ implementation in Python.

2 from each group will present to the class on your findings.

Assume the class has zero knowledge on the data structure. Start from nothing.

Time:
5 min

Numbers

- Integers e.g. -1, 0 , 7 , 2**100
  - No size limit in Python 3 (Use longs in Python 2 for this property)
  - Default - base 10 (decimal), can be base 8 or base 16 (useful for brevity)
- Floating Point Numbers e.g. -2.0, 34.3453, 2.5e6, 2E-10
  - Decimals - sys.float_info.dig tells you max digits Python can differentiate between
  - Operations between Int and Float give you a float
  - **float('Inf')** and **float('Nan')** are special floats useful for calculations
- Complex Numbers e.g. -2+3j, 2.4+1.2j, 1+1j *(not supported in all Python installations)*

Time:
5 min

Numbers

- Operators (in order of completion)
  - ** - exponentiation
  - *, / - multiplication and division
  - %, // - remainder and quotient on integer division
  - +,- - addition and subtraction
- Bitwise Operations (Convert to Bits and operate e.g. -, &, |, ^, >>, <<)
- Mathematical Functions (round, max, min, abs)
- Use the **math** module for more mathematical functions (sqrt, log, floor, exp, ceil, sin etc)
- Use the **random** module for more randomisation functions

Time:
5 min

Booleans

- Two possible values - True, False
- There are only two objects stored. You can have multiple reference to them.
- Each additional reference takes up 4 or 8 bytes of memory to store the reference.
- bool() - can be applied on any object and returns **true** unless it is:
  - None, False, 0, “”, [], (), {}
  - Objects from classes defined to have a __nonzero__ method
  - Objects from classes defined to return 0 or False in the __len__ method
- x **and** y **-** if bool(x) is False, return x, otherwise return y
- x **or** y - if bool(x) is False, return y, otherwise return x
- **not** x - if bool(x) is False, return True, otherwise return False

Time:
5 min

Booleans

- Comparison Operators return booleans: <, >, <=, >=, ==, !=
- Comparisons must be objects of same type (one-element at a time for collective DS)
- You can chain multiple comparisons together e.g. 1 < 2 < 3 returns True
- **is** and **is not** tell you if two variables point to the same object
- **in** and **not in** tell you if a variable is contained in a sequence (collective data type)
- **all** and **any** tell you if there are all or any true statements in a collective DS
- Order of Operations:
  - Calculations → Comparisons → not → and → or
  - Better to use brackets for clarity

Time:
5 min

Booleans - a few exercises

- What is the result of?
  - not 17- 8 and 3
  - [1,2] > [] and [1,3,2] > [2,1] or []
- Implement the following using boolean operators
  - nor - positive if and only if both inputs are negative
  - xor - positive if and only if one of two inputs are negative
  - nand - negative if and only if both inputs are positive
  - xnor - positive if and only if both inputs are the same

Time:
5 min

String

- Sequence of letters, symbols and numbers (known as **characters**)
- Can be declared in many ways - ‘hello’, “what’s the time”, ‘’’ hi ’’’, “””what’s up”””
- Triple quotes are used for multi-line quotes, double for including apostrophes
- You can treat them like lists: “hi” + “ bye”, “hello”[2], “hi” * 3, ‘hello’[1:-1], len(“hi”)
- Can be **formatted** in a print statement or using string.format() with the % sign
- A huge number of built-in string methods
  - .lower(), .upper(), .find(), .count(), .split(), .isdigit(), .capitalize(), .title(), .replace()
  - Can built one string from a list of strings using .join() (.split() breaks string to list)
  - Very complex but useful module named **regex** for analysing strings (texts)
- mystr = “hello”, mystr[1] = “a” will not work as strings are **immutable**!

Time:
5 min

Strings - a few exercises

- Implement the following functions in Python
  - Count the number of capital letters in a string
  - Check if a string is a palindrome (same forward as backwards)
  - Input a string and replace all instances of multiple spaces (2 or more) with tab ‘\t’
  - Convert an integer to a string with commas e.g. 1000 → ‘1,000’

Time:
5 min

Other Useful Data Types

- **datetime** package is very useful for working with dates and times (e.g. timeseries)
- **numpy.datetime64** and **pandas.timestamp** are also good options
- **decimal** package is good for more advanced real number manipulations
- We will look at lists but **numpy arrays** or **array** package will generally be more efficient
- **pandas dataframes** are convenient for manipulating tables

Time:
7 min

Questions

Time:
3 min

Next Steps

1. Commit Week 1 Implementations
2. Group 1 - Lists, Group 2- Dictionaries, Group 3 - Sets, Group 4 - Tuples.
    a. What is this data structure and what can we do with it?
    b. When should we use it? Which kind of situations?
    c. What are its' limitations? When should we not use it?
3. More challenges posted on Piazza (by End of Day)