Abstract Data Types

Intermediate Application Development

Otago Polytechnic
Dunedin, New Zealand
Kaiako: Tom Clark

Abstract Data Types

An *Abstract Data Type* (ADT) is comprised of two things:

- The set of values which the type can represent;
- The set of operations that can be performed with elements of that type.

It is not concerned with the implementation of the type. Two ADTs that represent the same set of the values with the same operations are the same ADT.

Examples of ADTs

- Stack
- Queue
- Set
- Tree
- Graph

Stack

A *stack* is a **Last in, first out** structure. Think of a stack of plates.

- Primary operations
  - push
  - pop
- Additional operations
  - peek
  - depth (size)
  - empty
  - full

List as a stack

- List methods
  - `append()` - basically push by another name
  - `pop()`
- See `https://docs.python.org/3/tutorial/datastructures.html#using-lists-as-stacks`

```
stack = []
print(stack) # []
stack.append('apple')
print(stack) # ['apple']
stack.append('banana')
print(stack) # ['apple', 'banana']
stack.append('cherry')
print(stack) # ['apple', 'banana', 'cherry']
stack.pop()
print(stack) # ['apple', 'banana']
```

A Stack Class

```
class Stack:
        def __init__(self):
            self.stack = []
        def push(self, item):
            pass
        def pop(self):
            pass
        def peek(self):
            pass
        def empty(self):
            pass

    stk = Stack()
    stk.push('cat')
```

Queue

A *queue* is a **First in, first out** structure. Think of a queue to check out at a supermarket.

- Primary operations
  - enqueue
  - dequeue
- Additional operations
  - peek
  - length
  - empty
  - full

Deque as a queue

- A List is not efficient when used for a queue.
- The `collections` module in the standard library provides a `Deque`.
- Deque methods
  - `append()`
  - `popleft()`
- See `https://docs.python.org/3/library/collections.html#collections.deque`

```
from collections import deque
queue = deque([])
print(queue) # deque([])
queue.append('apple')
print(queue) # deque(['apple'])
queue.append('banana')
print(queue) # deque(['apple', 'banana'])
queue.append('cherry')
print(queue) # deque(['apple', 'banana', 'cherry'])
queue.popleft()
print(queue) # deque(['banana', 'cherry'])
```

A Queue Class

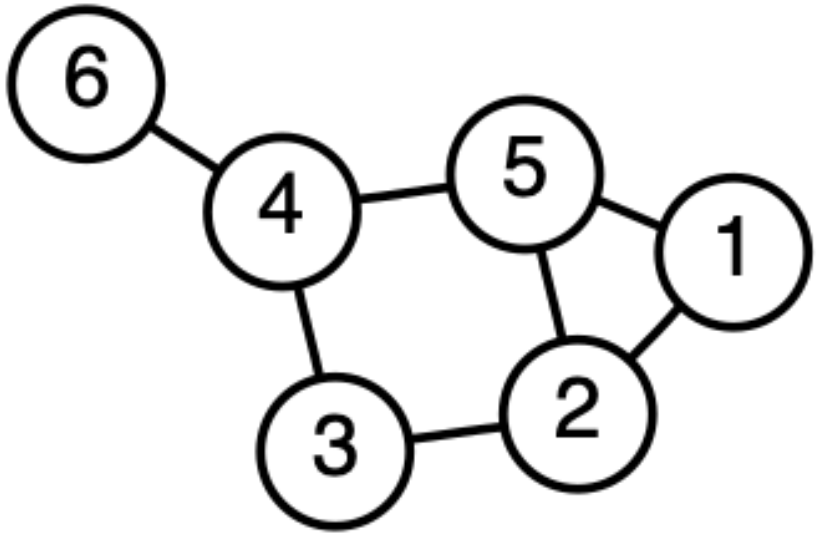
```
from collections import deque
class Queue:
    def __init__(self):
        self.queue = deque()
    def enqueue(self, item):
        pass
    def dequeue(self):
        pass
    def peek(self):
        pass
    def empty(self):
        pass

q = Queue()
q.enqueue('cat')
```

Programming Activity

1. Pull the course materials repo.
2. Create a new branch, `03-practical` in your practicals repo.
3. Add a subdirectory, `03-practical` and copy `03-practical.ipynb` from the class materials into it.
4. Open a shell, cd to this directory, and run `jupyter notebook` to open the notebook. Complete the first questions.
5. We will discuss results in 30ish minutes.

Graph

A *graph* is made up of a collection of values, called *nodes*, and a collection of *edges* that connect two nodes and indicate some sort of relationship between the nodes.



A special case of a graph is a *tree*.