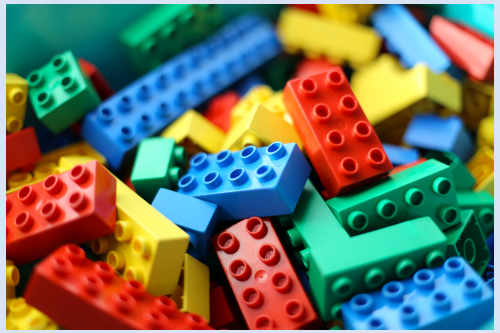
AMACSS Study Group

Object, ADTs, and implementation

Review of Objects

- Classes are the types of objects
- Allow us to define our own ”types”
  - Give them methods
  - Attach attributes to them
  - Allow us to customize our program beyond what’s built in
  - Think of them like building our own toys with Lego’s



```
class Lego():

    def __init__(self, size):
        self-_size = size
        print("You have created a lego object")

    def get_type(self):
        print("I am a coloured lego")

    def get_size(self):
        print("I am a " + _size + "sized lego")
```

Review of Objects

Private vs Public Variables

- Public variables are named regularly, anyone can access them in any way they want
- Private variables denoted by _var_name
  - Part of encapsulation is that users cant access these
  - create methods that allow them to change or access this var without ever calling it
    - i.e getters and setters
    - Accessing the variable requires knowledge of the implementation, and that's a no no

Examples of Objects

- Queue
  - First In First Out (FIFO)
  - Can be represented using a list
  - Usually has a get, put, peek, is_empty methods
  - See implementation
- Stack
  - Last in First Out (LIFO)
  - Can also be represented using a list
  - Similar methods to Queue
  - See implementation

More Examples of Objects

- Stueue
  - First insertion/removal operation acts like a Stack
    - Every operation after that switches between Queue and Stack
  - Similar methods as Stack and Queue
  - Try:
    - Implementation, OR
    - Turning ”CRABAPPLE” into “APPLECARB” using a Stueue
- Alternating Queue
  - Acts as a Queue, but alternates between left and right after every insertion/removal
  - Same methods as a Queue
  - See implementation

Review of ADTs

- Objects that can be used regardless of implementation
- User doesn't need to know the inner workings, only the methods
- Black box design
  - User has no idea what's going on under the hood

Let’s create an ADT!