Python Fundamentals, Data Structures, and Algorithms:

Week 4 Workshop Presentation

Workshop Agenda

Activity	Estimated Duration
Set up and check in	10 mins
Week 4 Review	60 mins
Assignment Tasks	40 mins
Break	15 mins
Remaining Assignment Tasks	100 mins
Check-Out (Feedback & Wrap-Up)	15 mins

Week 4 Review

Week 4 Review

Overview

OOP	Linked Lists
Classes	Doubly Linked Lists
Instances and methods	Stacks
__init__, self	Conditional expressions
Class inheritance	Queues
super()	

Review: OOP

- Object-Oriented Programming
- Classes are templates for objects
- Objects are instantiated from classes (thus also called instances)
- Classes/objects have attributes and methods
- Attributes and methods are like variables and functions, attached to specific classes/objects

Review: Classes

Syntax:

```
class ClassName:
    def __init__(self, attr1, attr2, …):
        self.attr1 = …
        self.attr2 = ...

    def method1(self, …):
        …
```

Review: Classes

Class example:

```
class Vehicle:
    def __init__(self, num_wheels, color):
        self.num_wheels = num_wheels
        self.color = color
        self.fuel_percent = 0
    def add_fuel(self, fuel_amt):
        self.fuel_percent += fuel_amt
```

Questions:

1. **How would you create an instance of this class with the name of "motorcycle" and a color of "black"?**
2. **How would you use the method to add fuel to the instance?**
3. **How would you write code to retrieve the fuel_percent?**

Review: Classes

```
class Vehicle:
    def __init__(self, num_wheels, color):
        self.num_wheels = num_wheels
        self.color = color
        self.fuel_percent = 0
    def add_fuel(self, fuel_amt):
        self.fuel_percent += fuel_amt
```

Possible answers:

1. **How would you create an instance of this class with the name of "motorcycle" and a color of "black"?** motorcycle = Vehicle(2, "black")
2. **How would you use the method to add fuel to the instance?** motorcycle.add_fuel(50)
3. **How would you write code to retrieve the fuel_percent?** motorcycle.fuel_percent

Review: Class Inheritance

Class inheritance example:

```
class Truck(Vehicle):
    def __init__(self, num_wheels, color):
        super().__init__(num_wheels, color)
```

The Truck class inherits from the Vehicle class.
Truck is child class/subclass, Vehicle is parent class/superclass.
The super function references the superclass, Vehicle

Question: Why do we use the super function as in the above code example?

Review: Class Inheritance

Class inheritance example:

```
class Truck(Vehicle):
    def __init__(self, num_wheels, color):
        super().__init__(num_wheels, color)
```

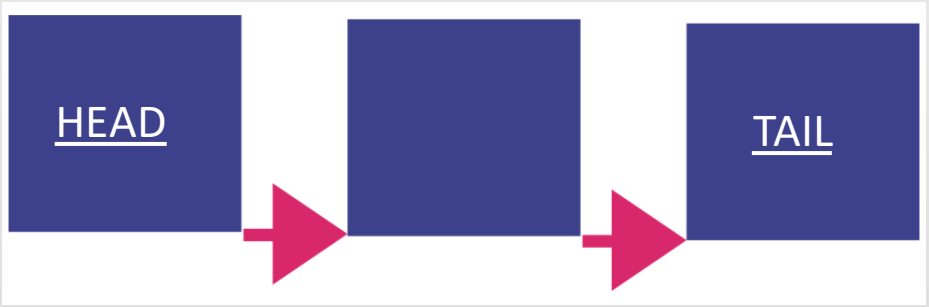
Answer: The super function is used to access the constructor method __init__ in the superclass, Vehicle, so that we can reuse the code there to initialize the num_wheels and color attributes.

Review: Linked Lists

- No native implementation in Python
- Like a List, but not indexed
- Thus less efficient at retrieving data, more efficient at inserting/deleting

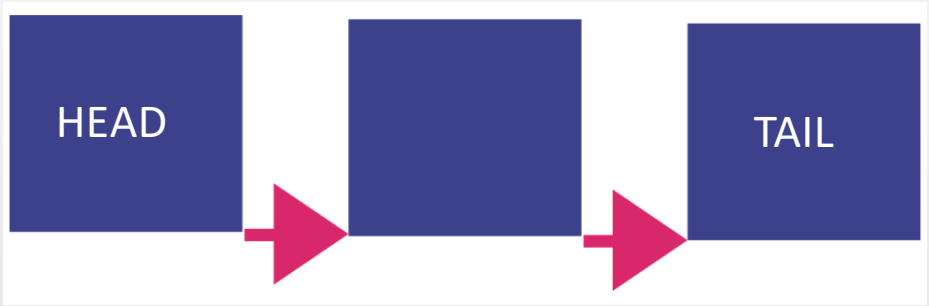
Review: Linked Lists

- Head node is first, tail node is last
- Each item ("node") in a linked list has reference to the next node
- Traverse entire linked list by going from one node to the next
- For tail node, the next node has a null value (such as Python's None value)
-

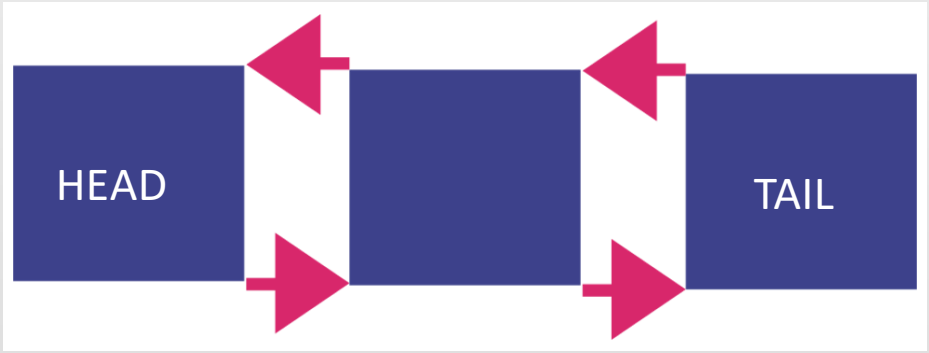


Review: Linked Lists

- **Singly Linked List:**



Doubly Linked List:



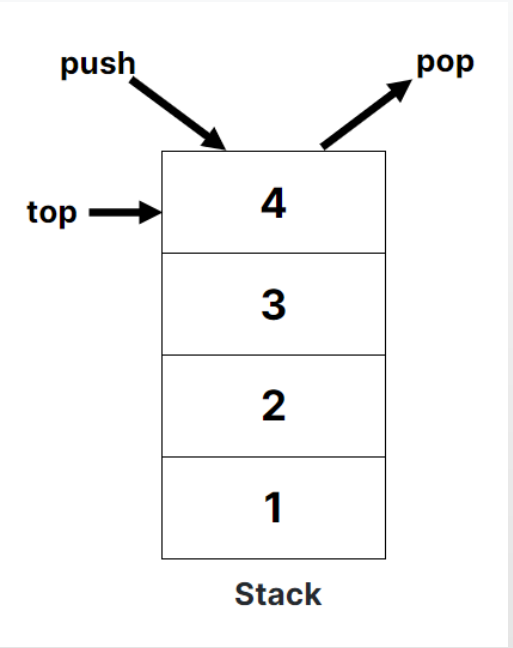
Review: Stacks

- Think of stacks like a stack of books or plates
- Two main operations: **push** and **pop**
- Last item added to stack is called the **"top"**

Questions:

1) LIFO stands for …?
2) What does push do?
3) What does pop do?

Review: Stacks



Questions:

1) LIFO stands for …?
   Last In First Out
2) What does push do?
   Adds new item to stack (becomes new "top")
3) What does pop do?
   Removes top item of stack

Review: Conditional Expressions

Example:

```
print("Pass" if num == 5 else "Fail")
```

Equivalent to:

```
if num == 5:
    print("Pass")
else:
    print("Fail")
```

Review: Conditional Expressions

Example:

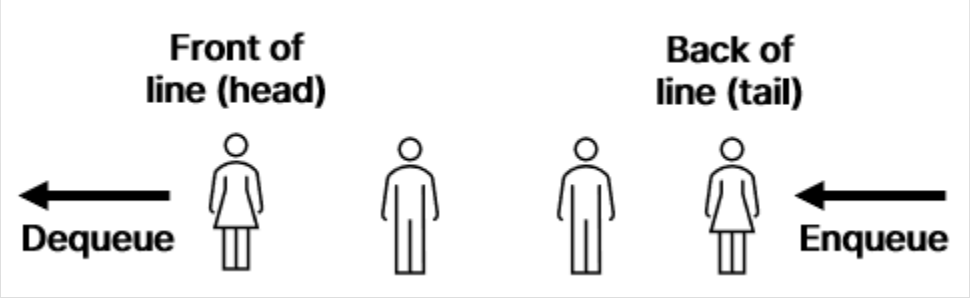
```
can_vote = True if age >= 18 else False
```

Equivalent to:

```
if age >= 18:
    can_vote = True
else:
    can_vote = False
```

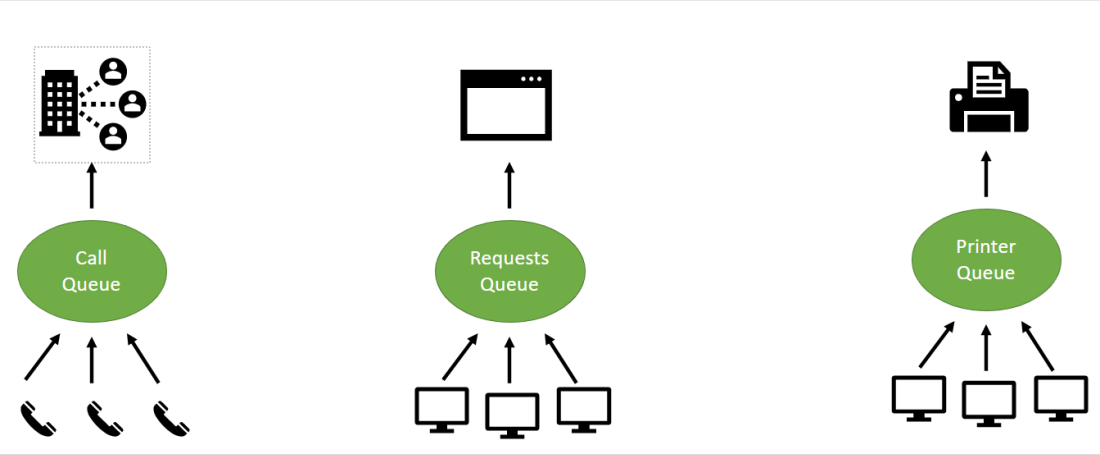
Review: Queues

- Think of queues like a line of people who are waiting for something
- Two main operations: **enqueue** and **dequeue**
- **Enqueue** adds an item, **dequeue** removes
- First item is the "head", last item is the "tail"
- **FIFO – First In First Out**
- First item in is the first item out



Review: Queues

- Queues are common all around us
- Examples:
  - Phone center call queues
  - Web server requests queue
  - Network printer jobs queue



Workshop 4 Assignment

Goal: Code part of a banking app using OOP

Task 1

Create User class

Task 2

Create User class instance methods

Task 3

Create BankUser subclass

Task 4

Create BankUser class instance methods

Task 5

Transfer and request money

You will be split up into groups to work on the assignment together.
Talk through each step out loud with each other, code collaboratively.
If your team spends more than 10 minutes trying to solve one problem,
ask your instructor for help!