# SCC461 – Programming for Data Scientists

Leandro Marcolino

Lancaster University

Week 8

# Outline

# What will be printed?

```
def callMe(input):
        b = 5;
        return input*b;

x = callMe(5);
print(x);
```

# What will be printed?

```
def callMe(input):
        b = 5;
        x = 10;
        return input*b;

b = 3;
x = callMe(5);
print(x);
print(b);
```

# What will be printed?

```
def callMe(input):
        print(input);
        if (input >= 0):
                callMe(input - 1);
        return 0;

x = callMe(5);
print(x);
```

# What will be printed?

```
a = 2;

b = a;

b = 999;

print(a);
```

# What will be printed?

```
a = [1, 2, 3, 4, 5];

b = a;

b[1] = 999;

print(a);
```

# What will be printed?

```
def callMe(input):
    input = 1;
    return;

myInput = 0;
callMe(myInput);
print(myInput);
```

# What will be printed?

```
def callMe(input):
    input[0] = 1;
    return;

myList = [0];
callMe(myList);
print(myList[0]);
```

# What will be printed?

```
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

p = Point()
q = Point(3,4)

print(p.x)
print(q.x)
```

# What will be printed?

```
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

p = Point()
q = Point(3,4)

p = q

q.x = 99

print(p.x)
print(q.x)
```

# What will be printed?

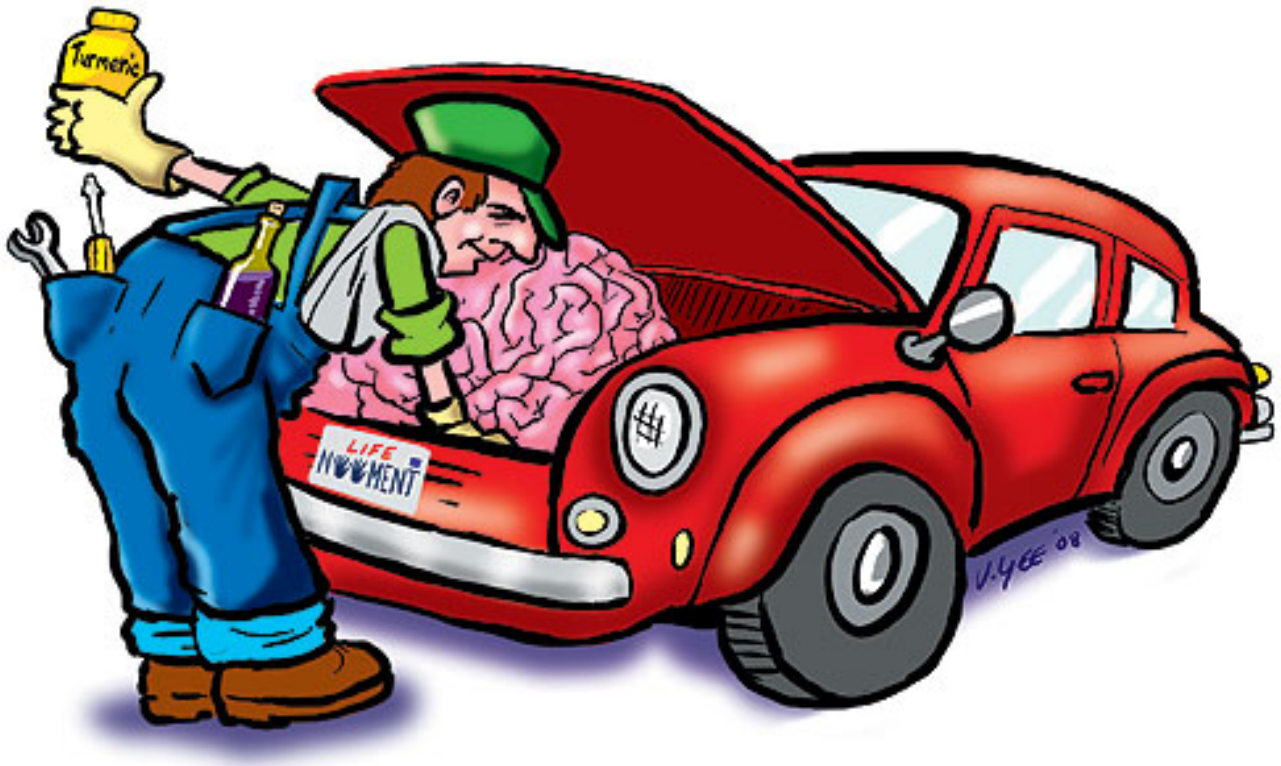
```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def distance_from_origin(self):
        return ((self.x ** 2) + (self.y ** 2))
            ** 0.5

def distance_from_origin():
    return -99

p = Point(1,1)
print(p.distance_from_origin())
print(distance_from_origin())
```

# What will be printed?

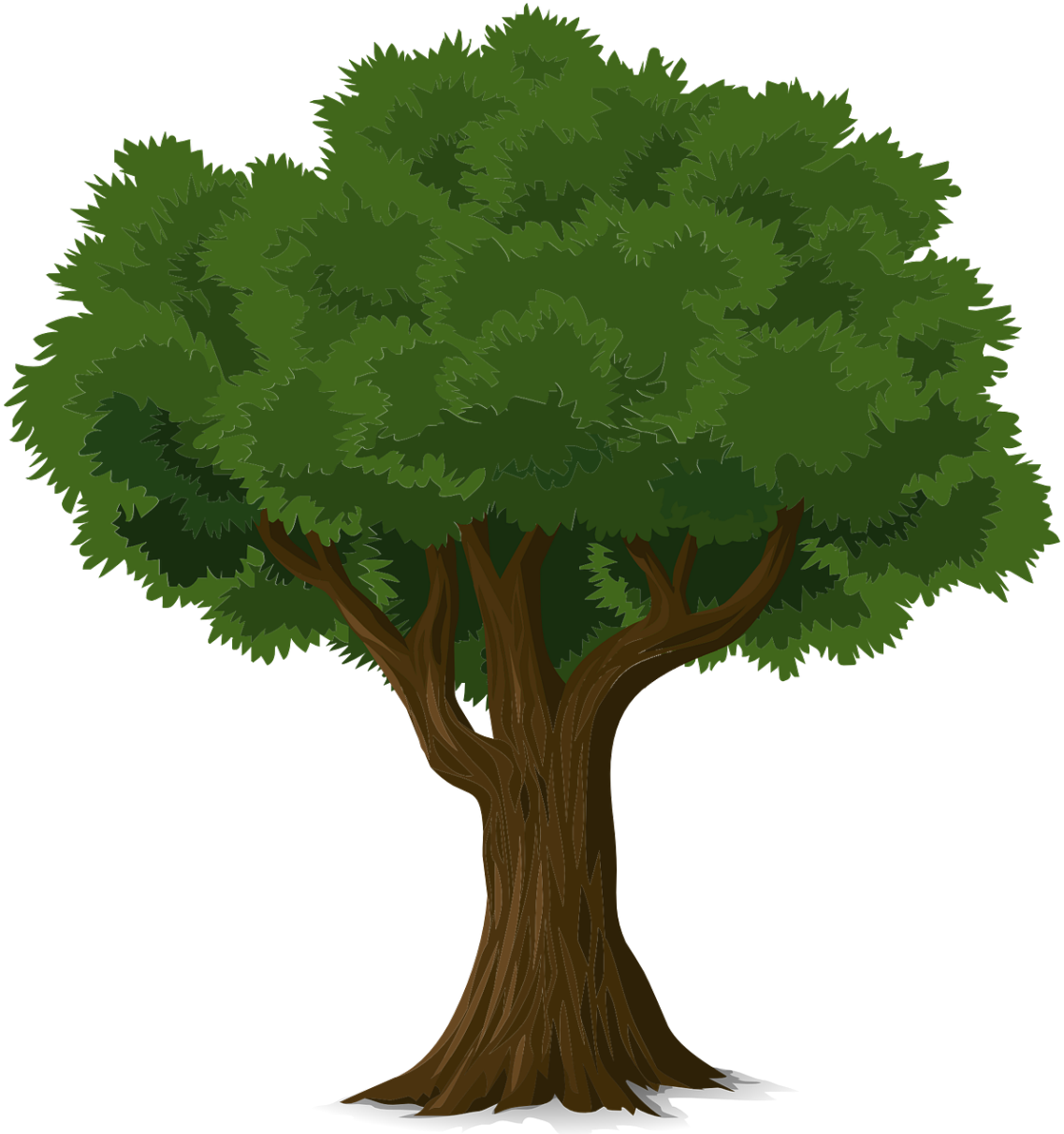
```
def x(pt):
    pt.x = -1;
    print(pt.x);

pt1 = Point();
pt2 = Point();

pt1.x = 5;
pt1 = pt2;

x(pt2);
print(pt1.x);
```

# Data Structures

- Stack
- Linked List
- Queue



# Why are we learning Stack, Queue and Linked List?

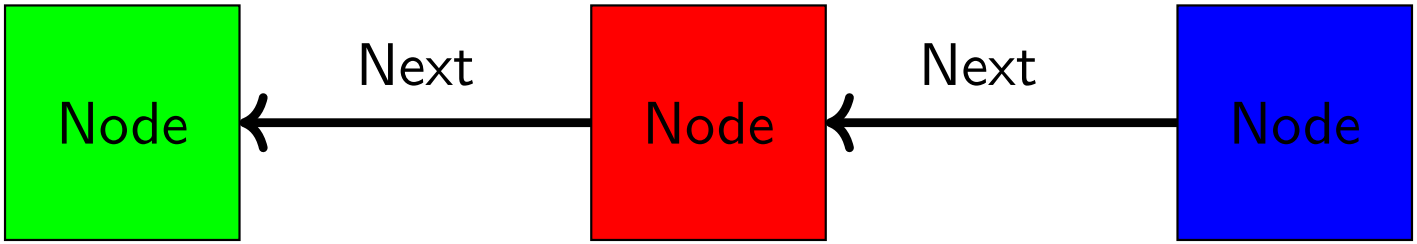


# OOP in R

```
setClass("student", slots=list(name="character",
    age="numeric", GPA="numeric"))

setMethod("show",
          "student",
          function(object) {
            cat(object@name, "\n")
            cat(object@age, "years old\n")
            cat("GPA:", object@GPA, "\n")
          }
)

s <- new("student",name="John", age=21, GPA=3.5)
```

# Linked List

Linked List: A node points to the next node



# Linked List

```
class Node:
    def __init__(self, content=None, next=None):
        self.content = content
        self.next    = next
```

# Linked List

## Example

```
node = Node("test")
print(node.content)
```

# Linked List

## Example

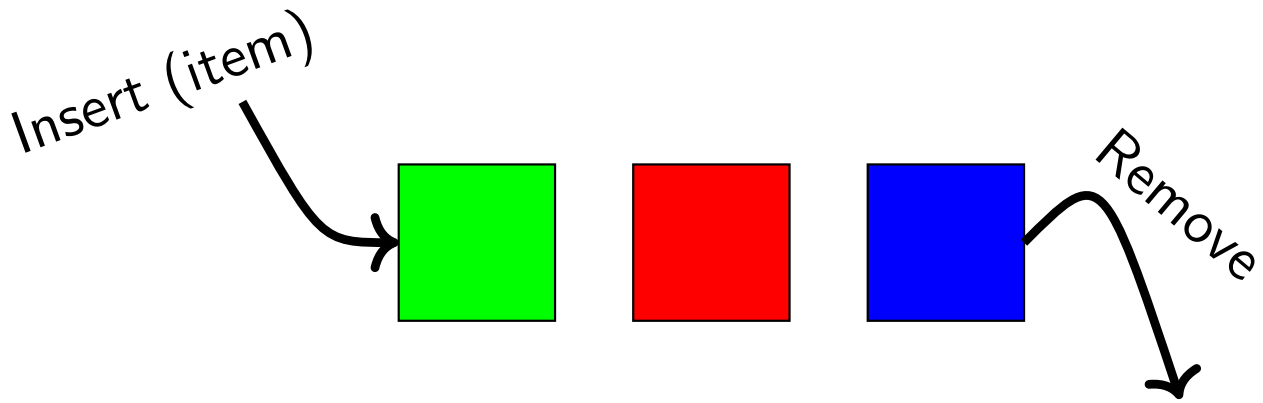
```
node1 = Node(1)
node2 = Node(2)
node3 = Node(3)

node1.next = node2
node2.next = node3
```

# What will be printed?

```
def print_list(node):
    while node is not None:
        print(node.content)
        node = node.next

node1 = Node(5)
node2 = Node(99)
node3 = Node(-3)
node4 = Node(67)

node1.next = node4
node3.next = node2
node4.next = node3

print_list(node1)
```

# Queue

Queue: First in, first out!



# Queue

Queue: A collection of items, with four methods:

- **Insert:** Adds an item to the end of the Queue.
- **Remove:** Return the item that is in the Queue for the longest time, and remove the item.
- **Clear:** Removes all items of the Queue.
- **Is Empty:** Returns whether the Queue is empty or not.

# Queue

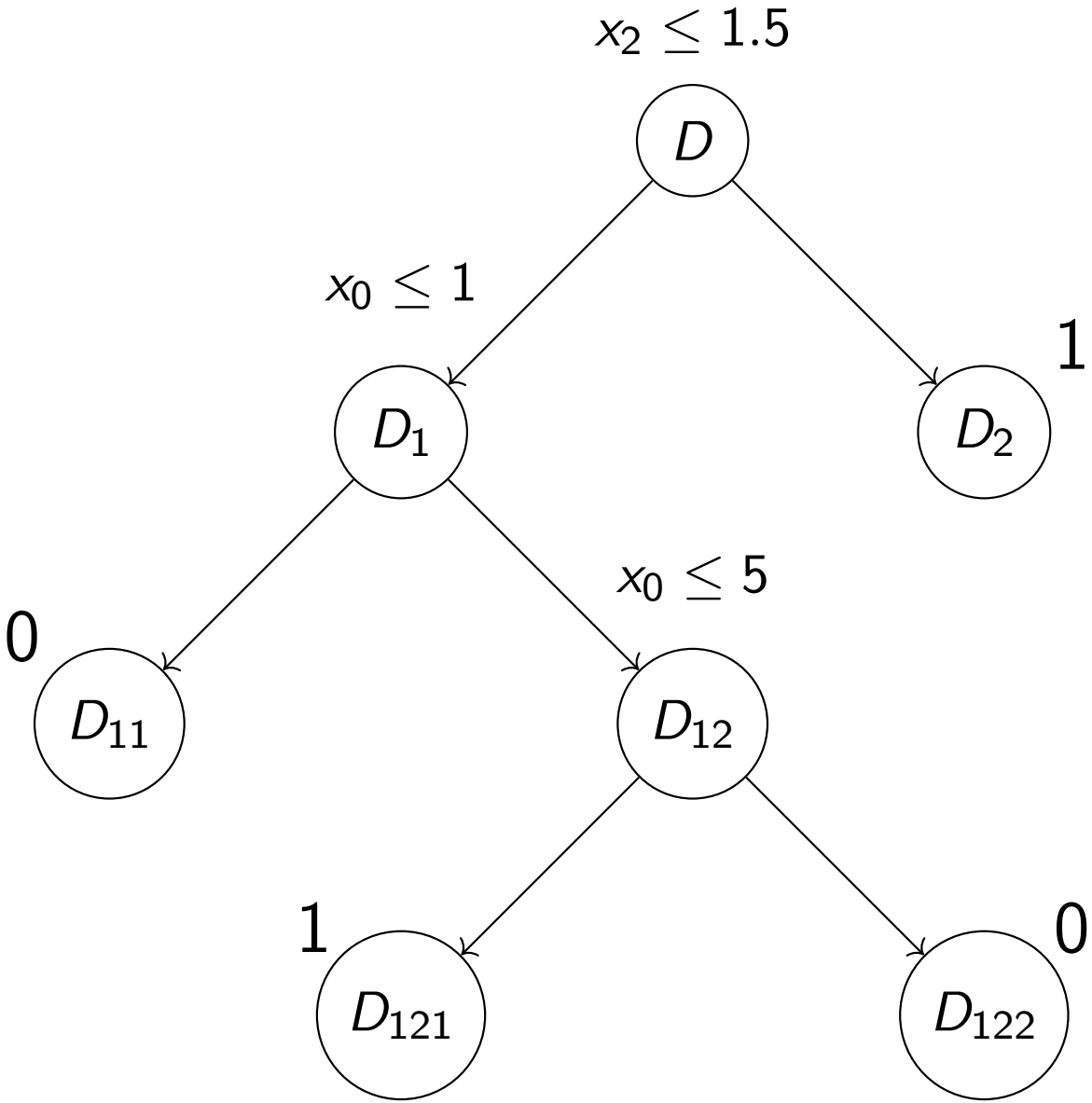
Queue.py file

# Peer Feedback Exercise

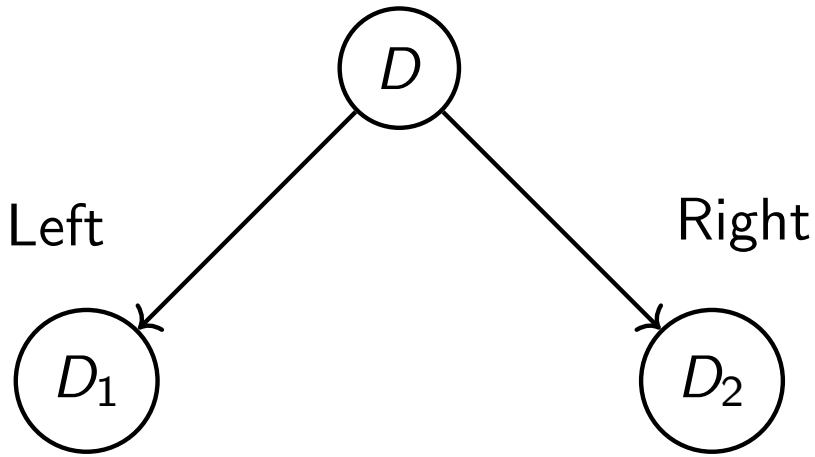
- WAIT FOR ALL INSTRUCTIONS BEFORE YOU START MOVING!
- You will work in pairs
- Discuss your CW 7 with your partner
- Ask your partner what he/she is struggling with, and teach him/her
- Similarly, tell your partner what you are struggling with, and he/she will teach you
- If you are new to programming, find an experienced programmer as a pair
- If you are an experienced programmer, find someone that is new to programming as a pair

# Outline

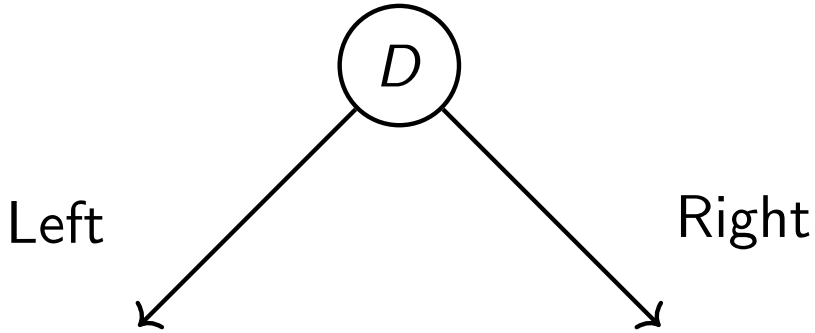
# Trees



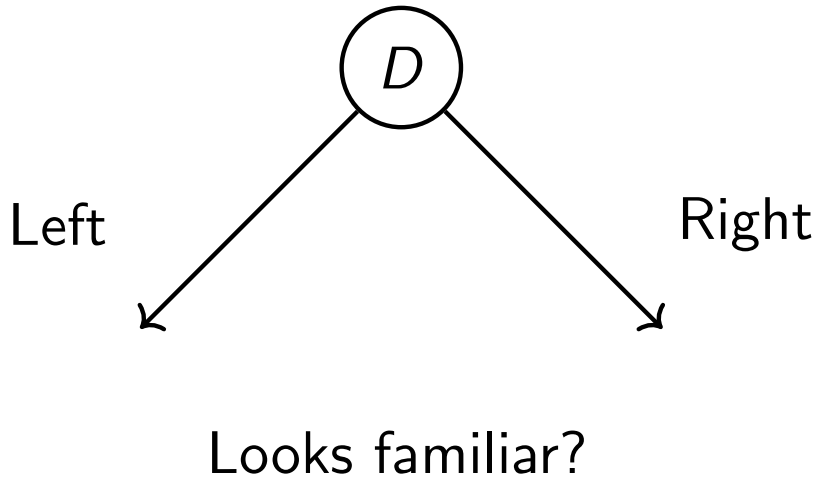
# Trees



# Trees



# Trees

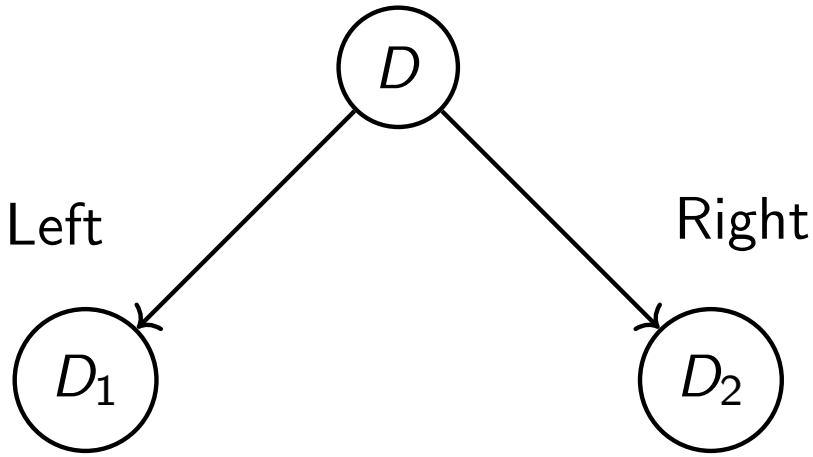


# Trees

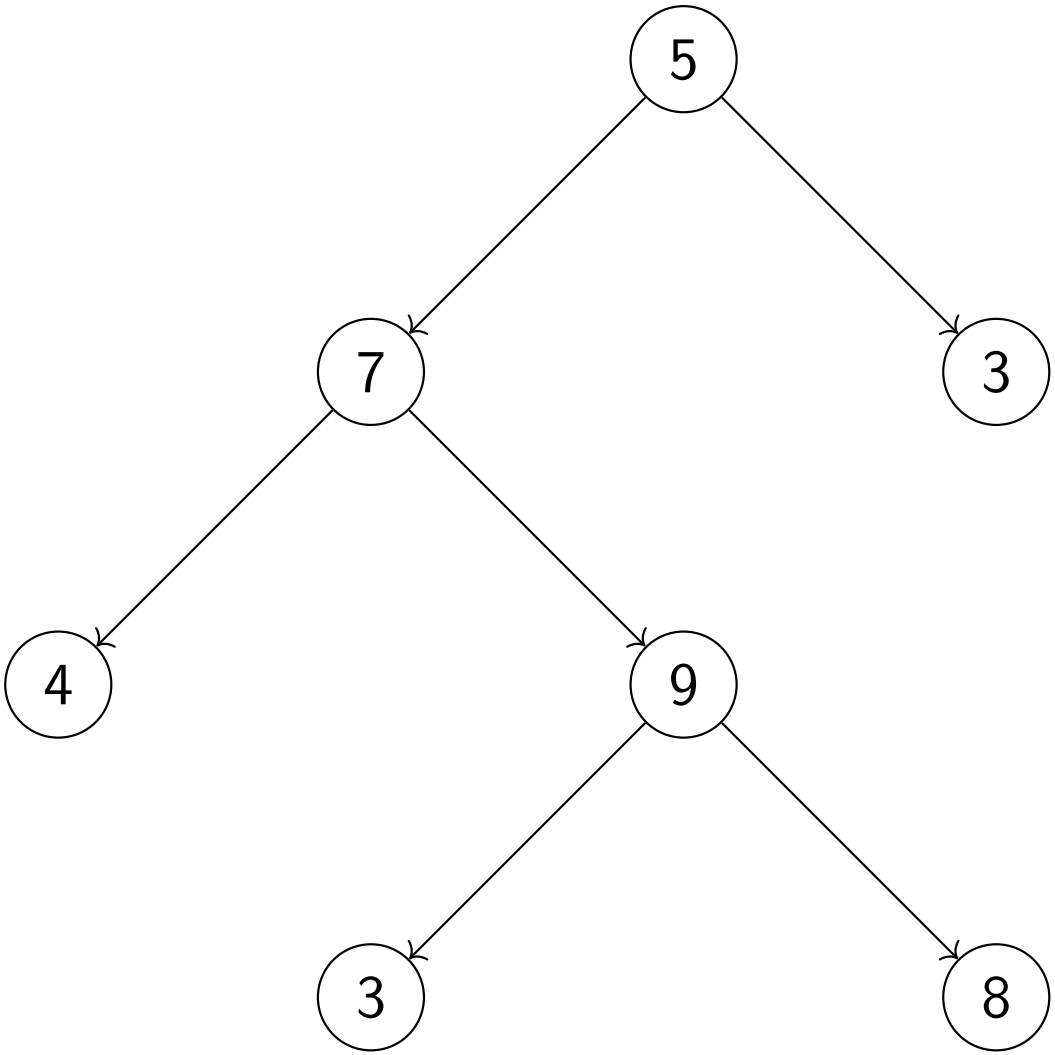
```
class Tree:
    def __init__(self, cargo, left=None,
        right=None):
         self.content = content
         self.left = left
         self.right = right
```

# Trees

```
nodeD = Tree("D");
nodeD1 = Tree("D1");
nodeD2 = Tree("D2");

nodeD.left = nodeD1;
nodeD.right = nodeD2;
```



# Trees



# Trees

## Exercise

- Write a function that sums up all the nodes of a tree

# Knowledge Trees

Section 27.7 of
http://openbookproject.net/thinkcs/python/english3e/trees.html

# Outline

# Stack again...

```
class Stack:
      def __init__(self):
          self.items = 10*[0];
          self.position = 0;

      def push(self, item):
           if (self.position < 10):
               self.items[self.position] = item;
               self.position = self.position + 1;
               return True;
           else:
               return False;

...
```

# Stack again...

```
    def pop(self):
        if (self.position <= 0):
            return False;
        else:
            self.position = self.position - 1;
            return self.items[self.position];

stack = Stack();

stack.push(5);
stack.push(10);
stack.pop();
stack.pop();
print(stack.items[1]);
```

# Private Members

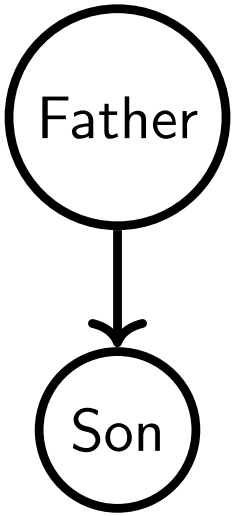
```
class Stack:
      def __init__(self):
          self._items = 10*[0];
          self._position = 0;

      def push(self, item):
           if (self._position < 10):
               self._items[self._position] = item;
               self._position = self._position +
                   1;
               return True;
           else:
               return False;
...
```

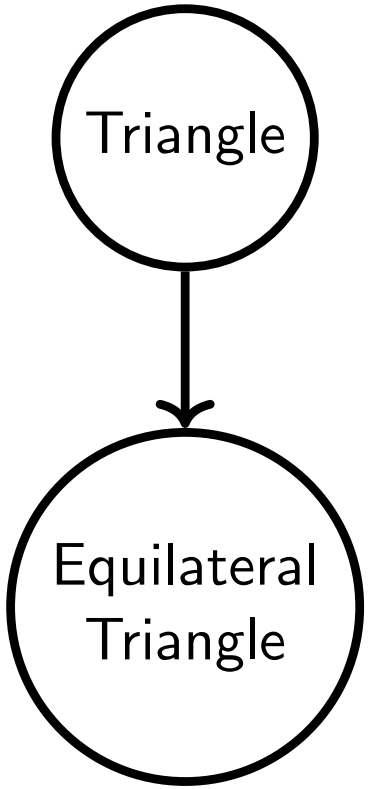
# Stack again...

```
def pop(self):
    if (self._position <= 0):
        return False;
    else:
        self._position = self._position - 1;
        return self._items[self.position];

def _checkStack(self):
    ....
```

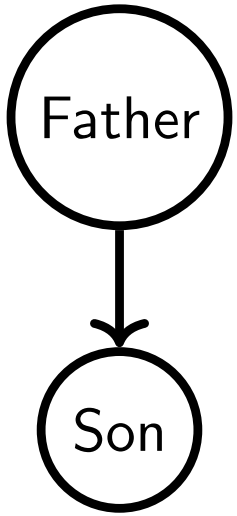
# Outline

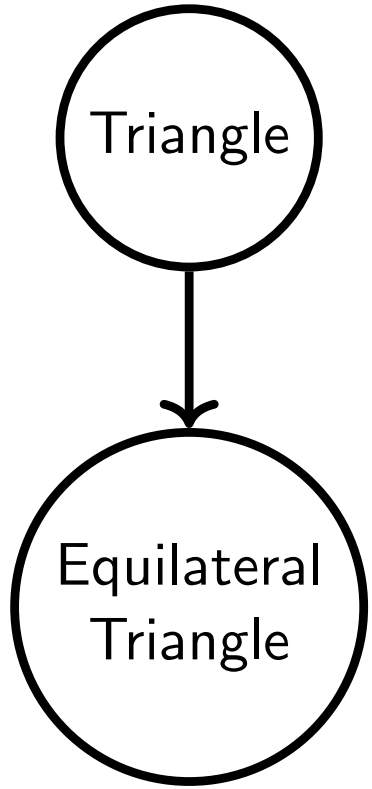
# Inheritance



# Inheritance



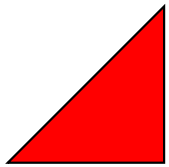

Inheritance: "Son" class has all that "Father" class has, AND MORE!

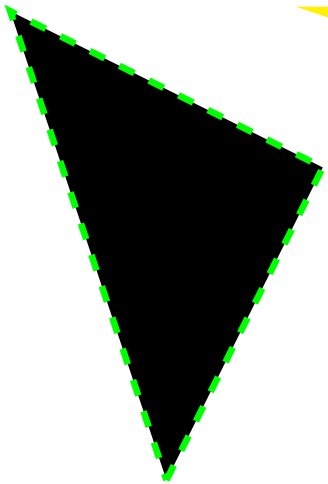
# Inheritance

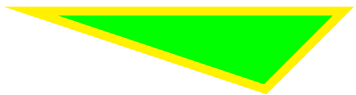
**Triangle:** The plane figure formed by connecting three points not in a straight line by straight line segments; a three-sided polygon. *American Heritage Dictionary of the English Language, Fifth Edition*

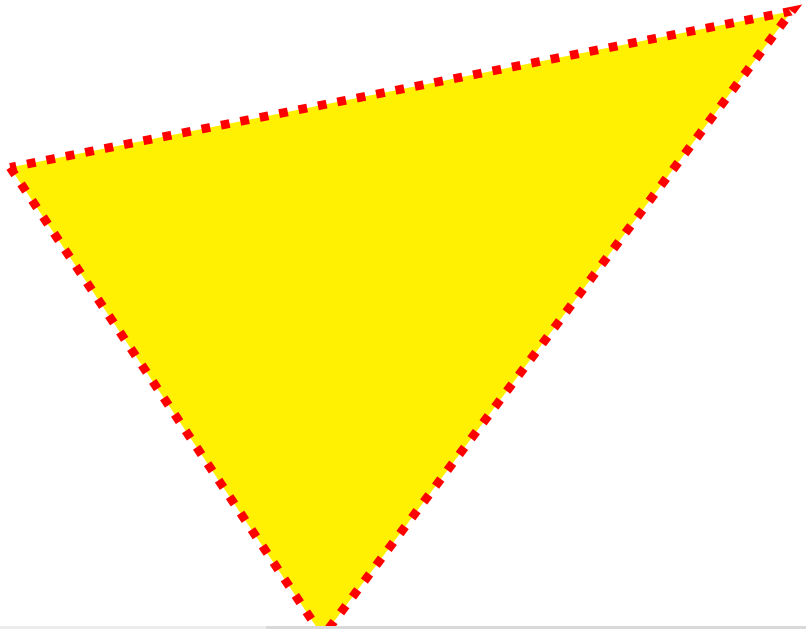
# Inheritance

**Triangle:** The plane figure formed by connecting three points not in a straight line by straight line segments; a three-sided polygon. *American Heritage Dictionary of the English Language, Fifth Edition*







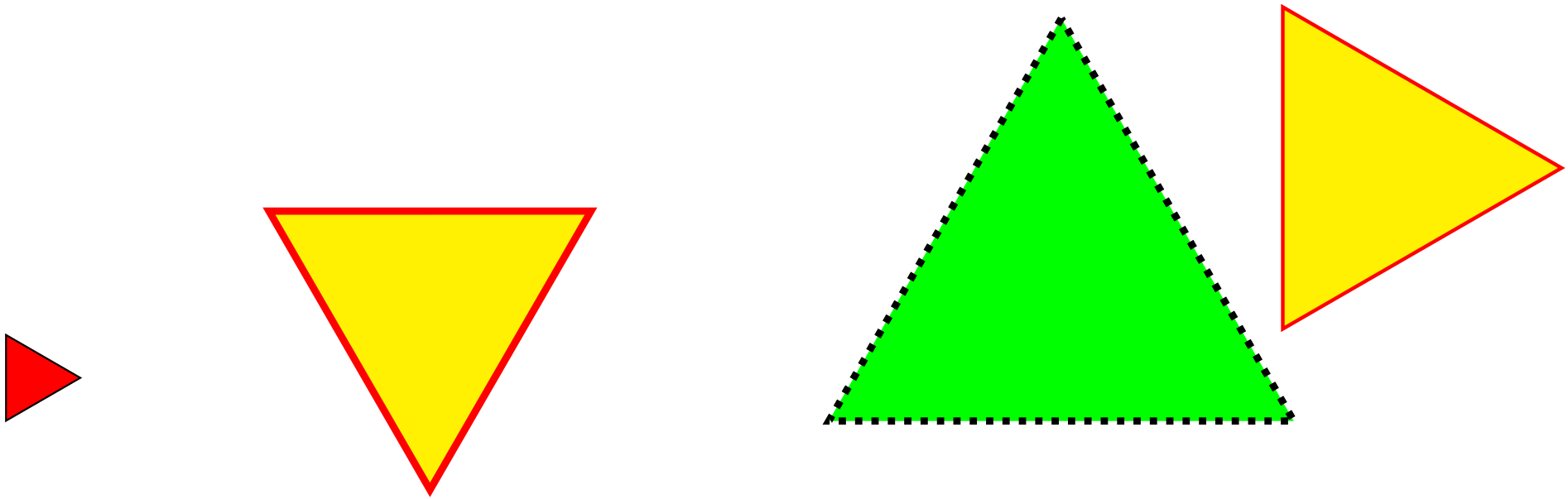


# Inheritance

**Equilateral Triangle:** a **triangle** in which all three sides are equal.
*Wikipedia*

# Inheritance

**Equilateral Triangle:** a **triangle** in which all three sides are equal.
*Wikipedia*



# Inheritance

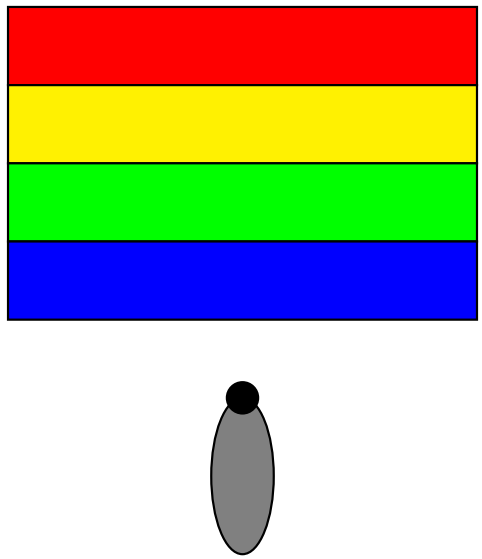
Why use inheritance?

- Organize code
- Re-use code

# Example

## OpenStack

A stack with an open bottom
We can see the bottom, but it does not get removed

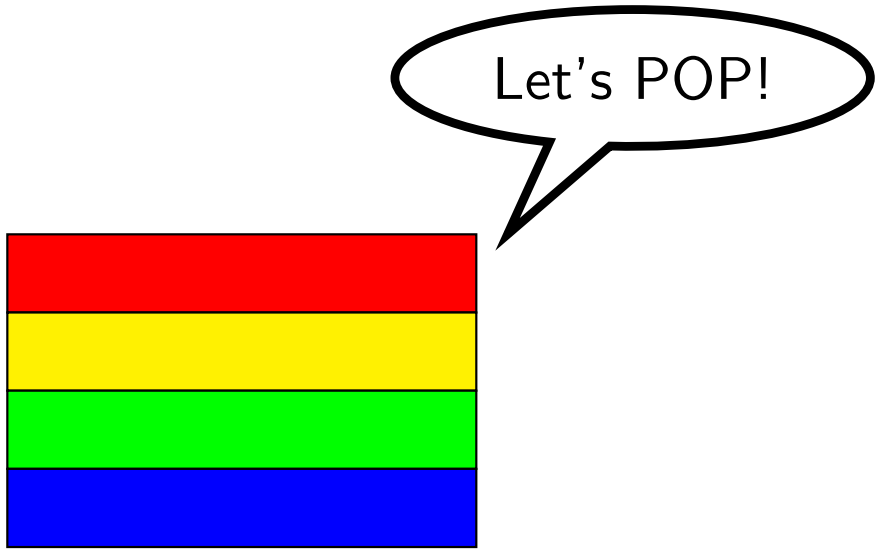


# OpenStack

```
class OpenStack(Stack):
    def seeBottom(self):
        return self.items[0];
```

# Example

## Talking Stack!



# Talking Stack

```
class TalkingStack(Stack):
    def __init__(self):
        Stack.__init__(self);
        print("Let's POP!");

    def push(self, item):
        print("I will push " + str(item) + " to
            the stack!");
        return Stack.push(self,item);

    def pop(self):
        returnMe = Stack.pop(self);
        print("Oh! Just got " + str(returnMe) +
            " from the stack!");
        return returnMe;
```

# Polymorphism

How do you print an equilateral triangle?

# Polymorphism

How do you print an equilateral triangle?
In the same way that you print a triangle!

# Polymorphism

```
def printTriangle(x):
    ...


triangle = Triangle();
equilateral = EquilateralTriangle();

printTriangle(triangle);
printTriangle(equilateral);
```

# Polymorphism

```
def addTenToStack(x):
    for i in range(10):
        x.push(i);

s = Stack();
addTenToStack(s);

ts = TalkingStack();
addTenToStack(ts);
```

# Outline

# Module

- Module: A collection of classes and/or functions
- Can be loaded with "import"

# Module

## Example

```
import random

generator = random.Random();

print (generator.random());
```

# Library

- Library: a collection of modules
- Python Standard Library: https://docs.python.org/3/library/index.html

# Standard Library

## Exercise

1. Using the library, calculate the mean and standard deviation of a set of integers
2. Using the library, find a way to divide a string with "," into a list of integers. Example: "1, 2, 3, 4" should become [1, 2, 3, 4].
3. Using the library, find a way to randomly sample, without replacement, 3 items out of a set of integers

# Outline

# Assignment

1. Write a DecisionTree class, for a problem with two possible labels (0 or 1), and two features. The first is a continuous number between 0 and 10, the second is a categorial variable with the following possible values: 0, 1, 2. (4%)
2. Re-write your CW 7, using inheritance. (1%)

Thank you!

l.marcolino@lancaster.ac.uk
http://www.lancaster.ac.uk/staff/sorianom/