# Parker, Prokofiev
# &
# Python

Programming is musicianship

# Charlie Parker

Jazz Musician

Alto Saxophone

# Sergei Prokofiev

Classical Pianist
Composer  Conductor

# Autodidact
# &
# OTJ

(on the Job Training)

3 Years of
Woodshedding

Daily Practice

Nightly Gigs

# Conservatory Trained

Formal and established pedagogy

Mastery of:
Instrument
Technique
Theory

Innovative
Compositions

# Orthogonal Approach

# Mastery of Language

Sergei Prokofiev

# Ten Pieces

## 1. March

**Allegro**

*pp*

1. 2.

8

*pp*

8

*p*

Explicit Instructions

Sergei Prokofiev

# Ten Pieces

## 1. March

**Allegro**

*pp*

*pp*

*p*

Perform as written

# Formal Specification

318

# PASSPORT

-CHARLIE PARKER

(BOP)

B♭maj7 C-7 F7 D-7 G7#5 C-7 F7

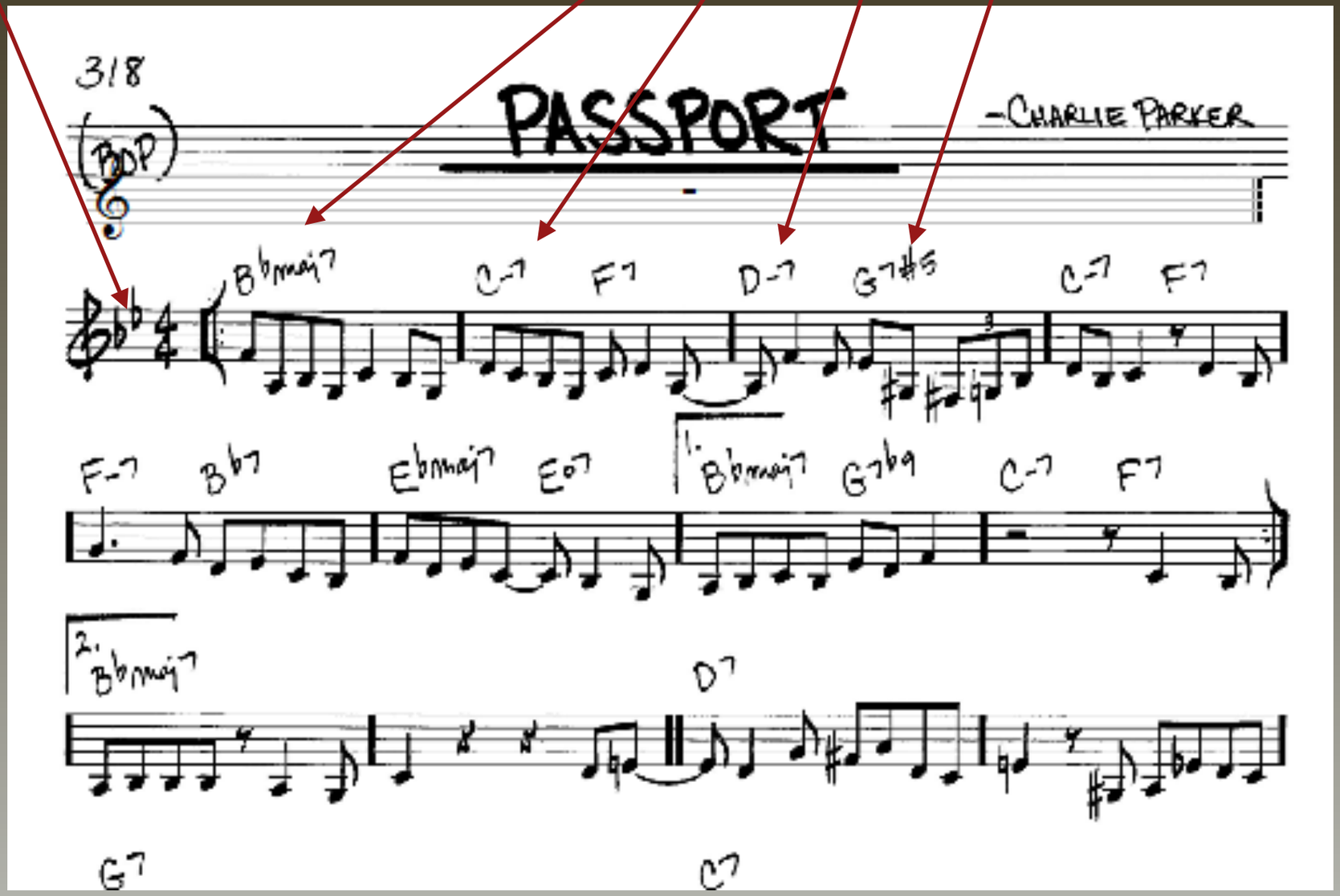
F-7 B♭7 E♭maj7 E°7 1. B♭maj7 G7♭9 C-7 F7

2. B♭maj7 D7

G7 C7

Instructions

Guidelines



Perform as suggested

# General Request

# Open to interpretation*

* If you know what you are doing

# Classical: Final Class

# Jazz: Extensible class

# Jazz: S.O.L.I.D

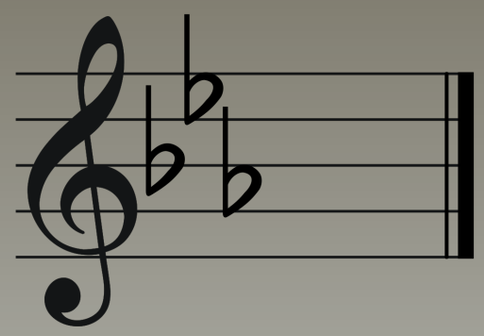
# Not Ashford and Simpson*

* Solid (as a rock)

S : Single Responsibility
O : Open/Closed
L : Liskov Substitution
I : Interface Segregation
D : Dependency Inversion

1, b3, 5, 7

```
@logging
def some function(self):
    pass

__slots__
```



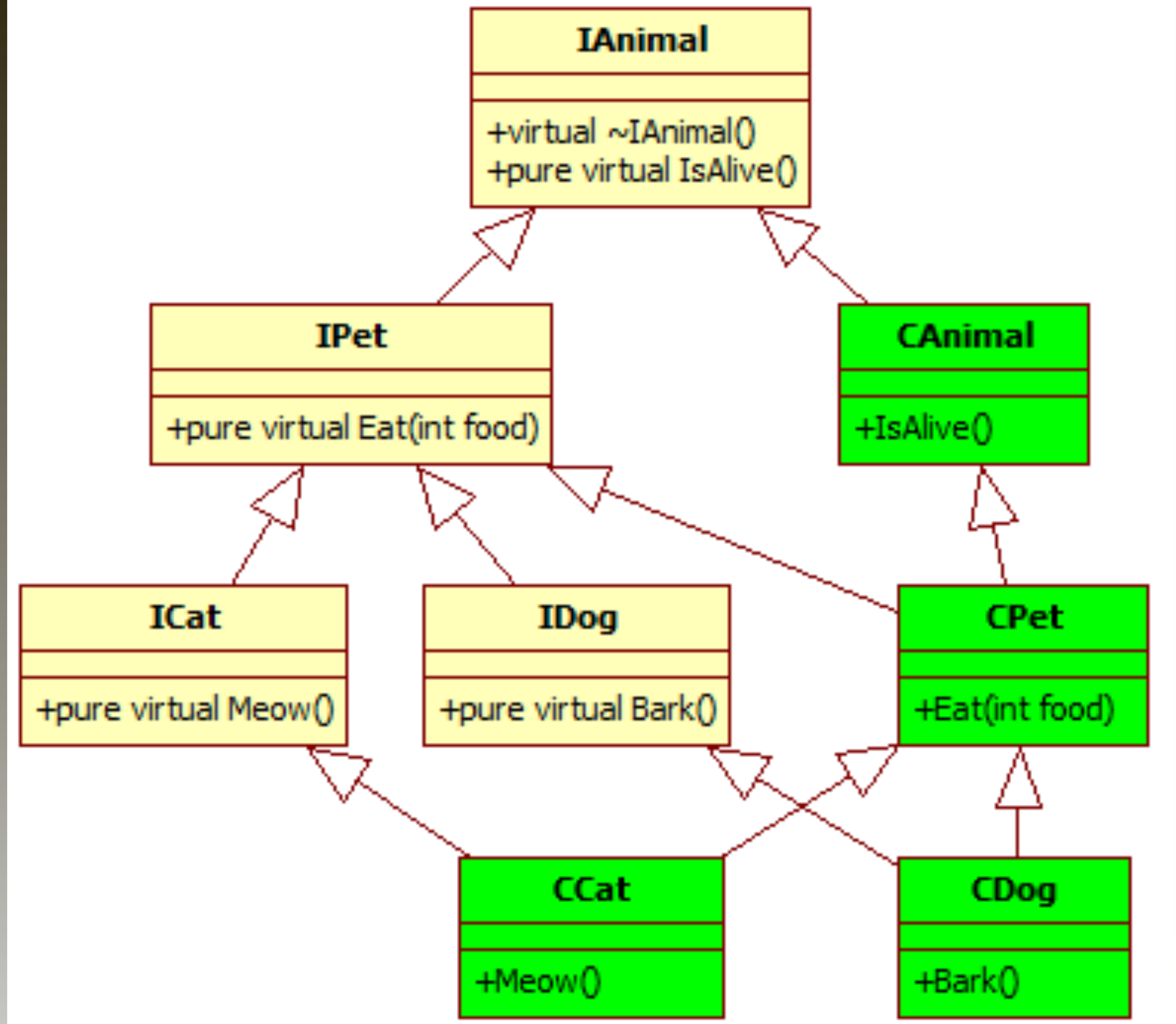
# Contextual Information

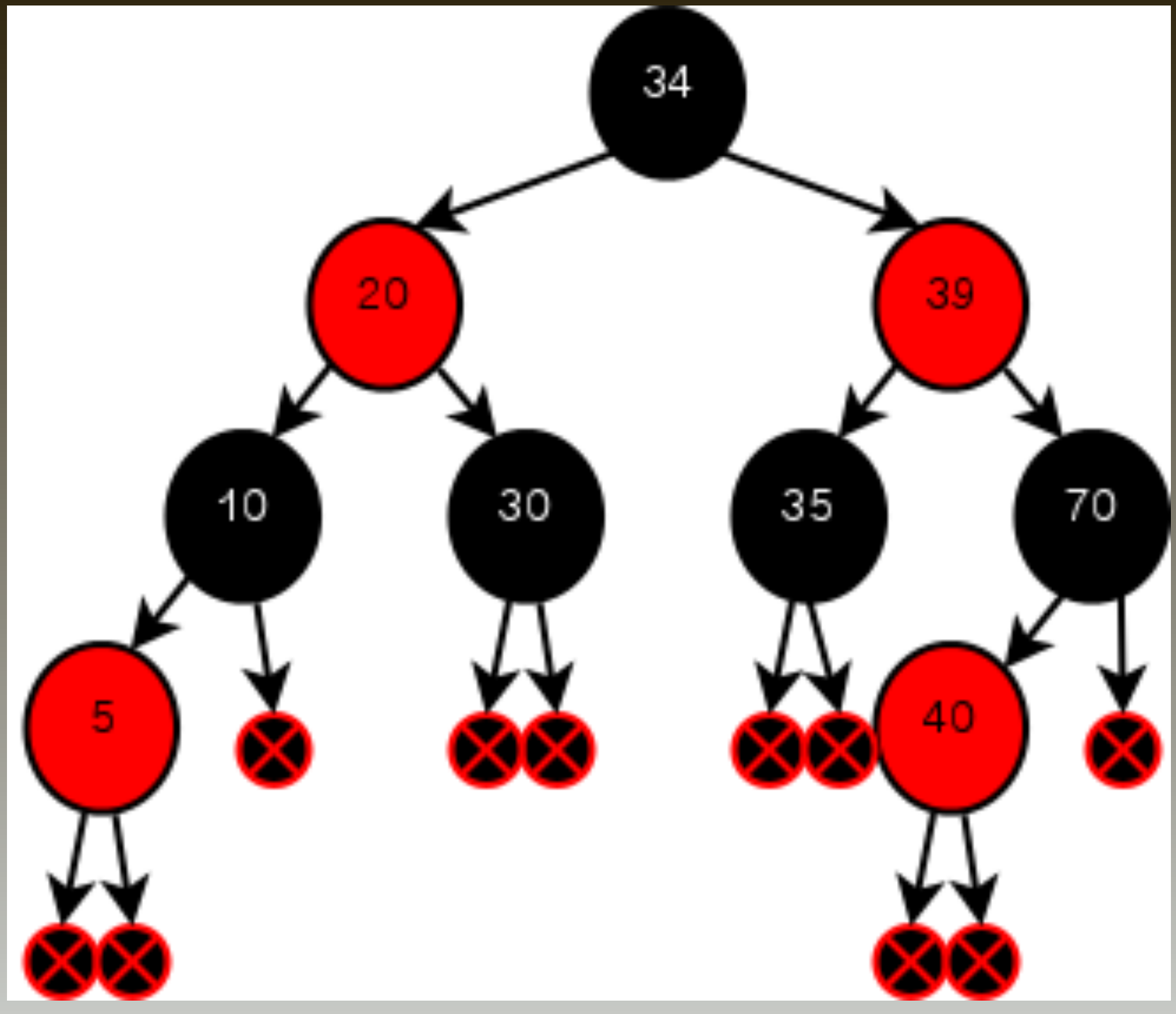
# Mastery of Language

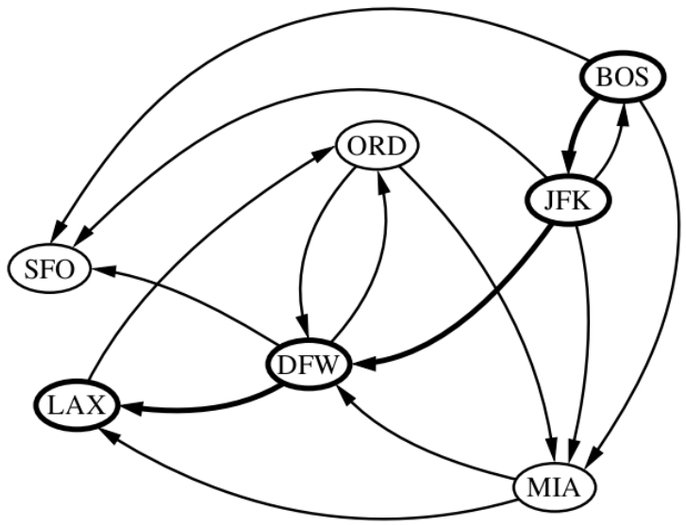
# Classical: Etudes

# Jazz: Standards

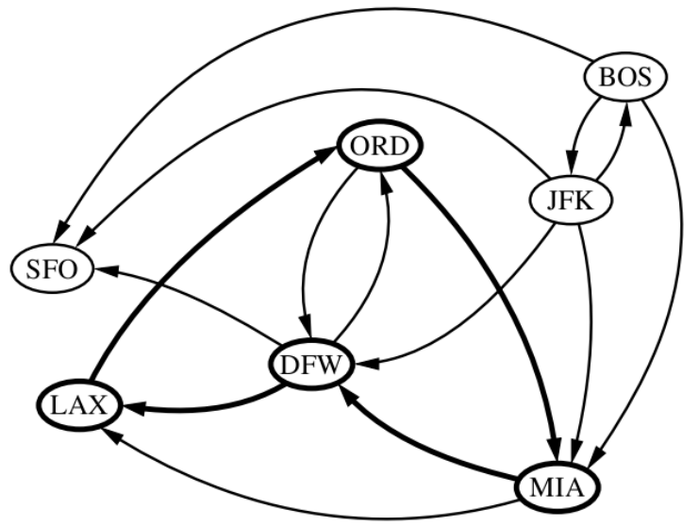
```
class EmitVisitor(asdl.VisitorBase):
    """Visit that emits lines"""

    def __init__(self, file):
        self.file = file
        self.identifiers = set()
        super(EmitVisitor, self).__init__()

    def emit_identifier(self, name):
        name = str(name)
        if name in self.identifiers:
            return
        self.emit("_Py_IDENTIFIER(%s);" % name, 0)
        self.identifiers.add(name)

    def emit(self, s, depth, reflow=True):
        # XXX reflow long lines?
        if reflow:
            lines = reflow_lines(s, depth)
        else:
            lines = [s]
        for line in lines:
            line = (" " * TABSIZE * depth) + line + "\n"
            self.file.write(line)
```


IAnimal
+virtual ~IAnimal()
+pure virtual IsAlive()
IPet
+pure virtual Eat(int food)
CAnimal
+IsAlive()
ICat
+pure virtual Meow()
IDog
+pure virtual Bark()
CPet
+Eat(int food)
CCat
+Meow()
CDog
+Bark()

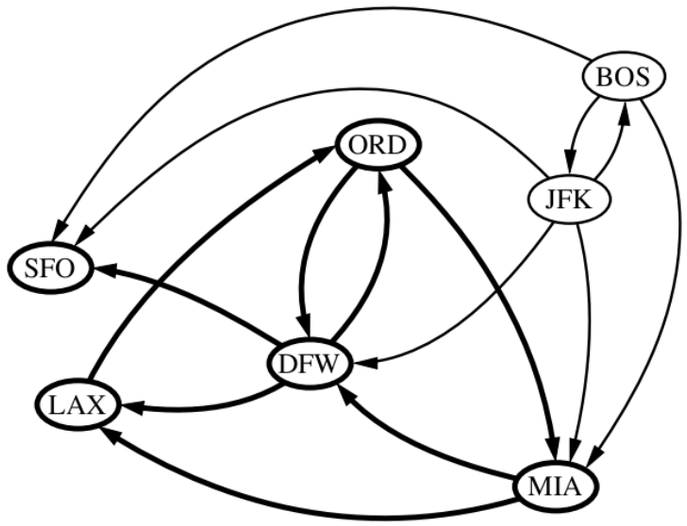

34
20
39
10
30
35
70
5
40


BOS
ORD
JFK
SFO
DFW
LAX
MIA

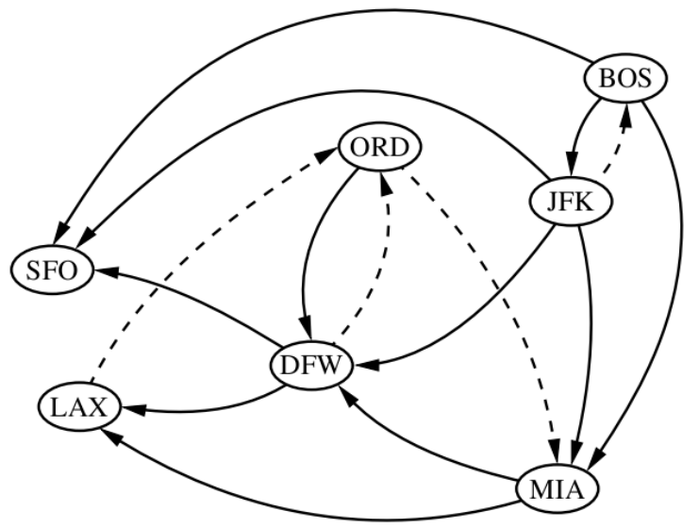
(a)


BOS
ORD
JFK
SFO
DFW
LAX
MIA

(b)


BOS
ORD
JFK
SFO
DFW
LAX
MIA

(c)


BOS
ORD
JFK
SFO
DFW
LAX
MIA

(d)

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \sum_{s=0}^{n-1} \mathbf{A}^s \sum_{k_1,k_2,\ldots,k_{n-1}} \prod_{l=1}^{n-1} \frac{(-1)^{k_l+1}}{l^{k_l}\, k_l!} \operatorname{tr}(\mathbf{A}^l)^{k_l}$$

# Big O Notation Summary

| Notation | Type | Examples | Description |
|---|---|---|---|
| O(1) | Constant | Hash table access | Remains constant regardless of the size of the data set |
| O(log n) | Logarithmic | Binary search of a sorted table | Increases by a constant. If n doubles, the time to perform increases by a constant, smaller than n amount |
| O(< n) | Sublinear | Search using parallel processing | Performs at less than linear and more than logarithmic levels |
| O(n) | Linear | Finding an item in an unsorted list | Increases in proportion to n. If n doubles, the time to perform doubles |
| O(n log(n)) | n log(n) | Quicksort, Merge Sort | Increases at a multiple of a constant |
| $O(n^2)$ | Quadratic | Bubble sort | Increases in proportion to the product of n*n |
| $O(c^n)$ | Exponential | Travelling salesman problem solved using dynamic programming | Increases based on the exponent n of a constant c |
| O(n!) | Factorial | Travelling salesman problem solved using brute force | Increases in proportion to the product of all numbers included (e.g., 1*2*3*4…) |

# Implementation

# Application of Theory

# Regular Practice

# Know Your Audience

# What to Learn?

# Know your business

# Data Structures

# Algorithms

# Computer Architecture

# Python Internals

# When to ask questions

# You Can Be Incredible Too!