PES
UNIVERSITY

UE18CS101:
INTRODUCTION TO COMPUTING USING
python™

Department of Computer Science and Engineering
PES UNIVERSITY

Lecture 2

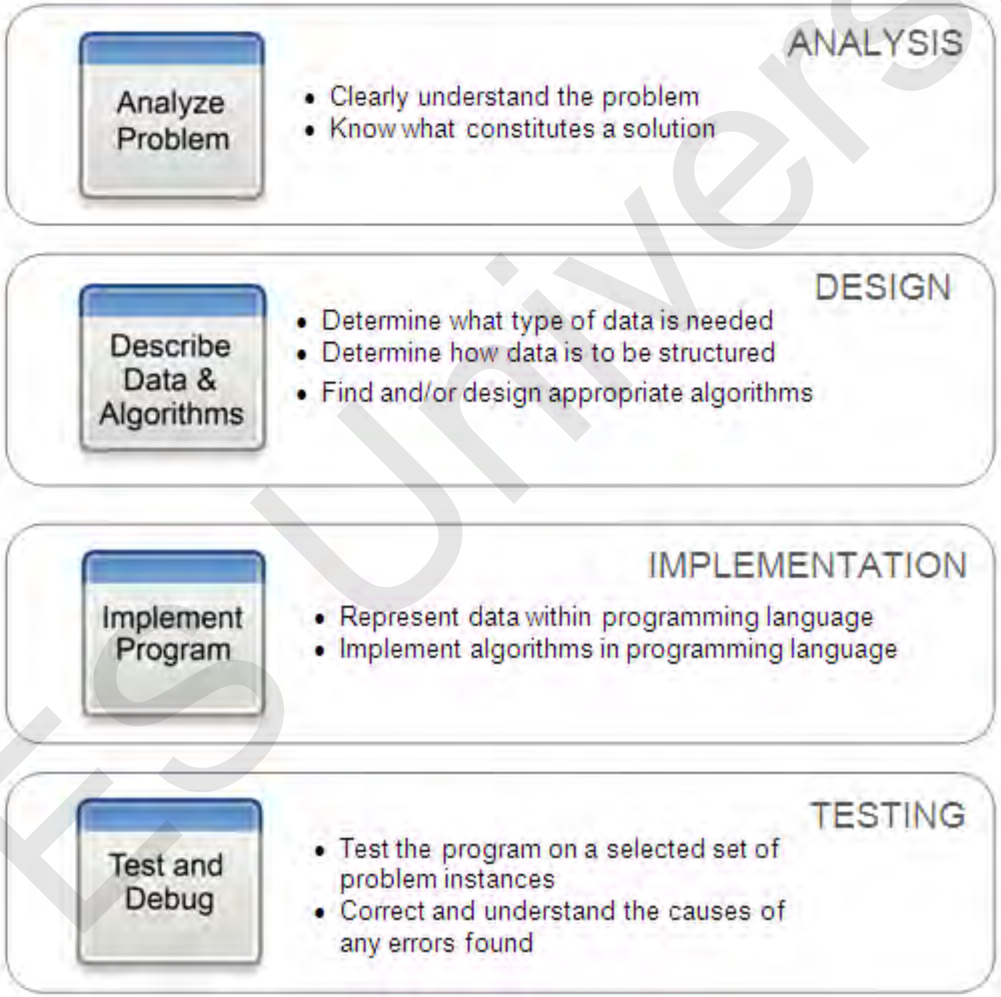
Process of Computation Problem Solving - Analysis, Design, Implementation, Testing

The Process of Computational Problem Solving

Computational problem solving does not simply involve the act of computer programming. It is a *process*, with programming being only one of the steps.

Before a program is written, a design for the program must be developed. And before a design can be developed, the problem to be solved must be well understood. Once written, the program must be thoroughly tested.

Computational Problem Solving Steps



ANALYSIS

Analyze Problem

- Clearly understand the problem
- Know what constitutes a solution

DESIGN

Describe Data & Algorithms

- Determine what type of data is needed
- Determine how data is to be structured
- Find and/or design appropriate algorithms

IMPLEMENTATION

Implement Program

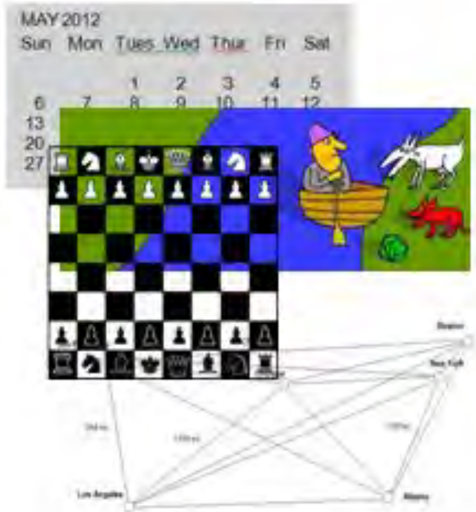
- Represent data within programming language
- Implement algorithms in programming language

TESTING

Test and Debug

- Test the program on a selected set of problem instances
- Correct and understand the causes of any errors found

Problem Analysis



Must understand the fundamental computational issues involved

- For **calendar month problem**, can use direct calculation for determining the day of the week for a given date
- For **MCGW problem**, can use brute-force approach of trying all of the possible rowing actions that may be taken
- For the **Traveling Salesman** and **Chess playing problems**, a brute-force approach is intractable. Therefore, other more clever approaches need to be tried

Knowing what constitutes a solution.

For some problems, there is only one solution. For others, there may be a number (or infinite number) of solutions. Thus, a problem may be stated as finding,

- **A solution** (calendar month, chess playing)
- **An approximate solution**
- **A best solution** (MCGW, Traveling Salesman Problem)
- **All solutions**

Describe Data and Algorithms

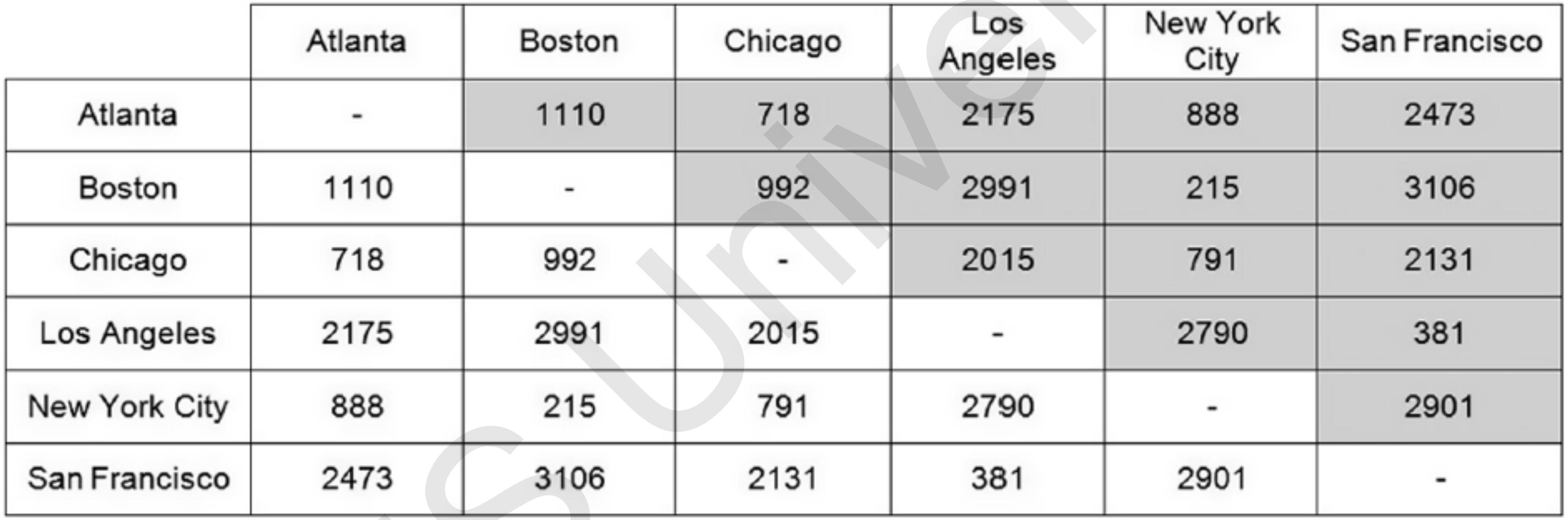
- For **calendar month problem**, need to store the month and year, the number of days in each month, and the names of the days of the week

- For the **MCGW problem**, need to store the current state of the problem (as earlier shown)

- For **Traveling Salesman** need to store the distance between every pair of cities

- For the **chess playing problem**, need to store the configuration of pieces on a chess board

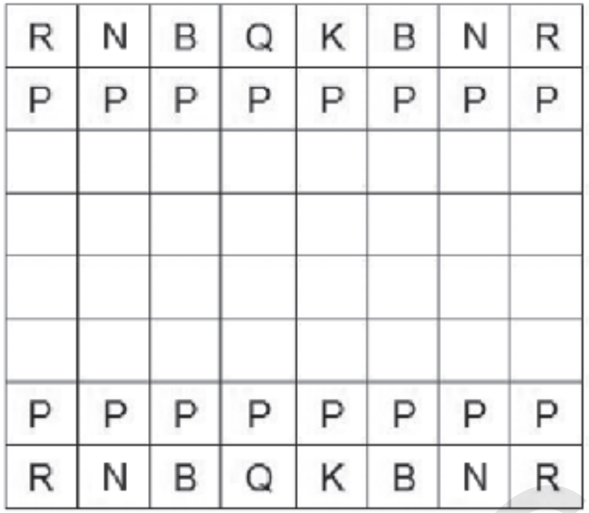
Table Representation of Data for the Traveling Salesman Problem



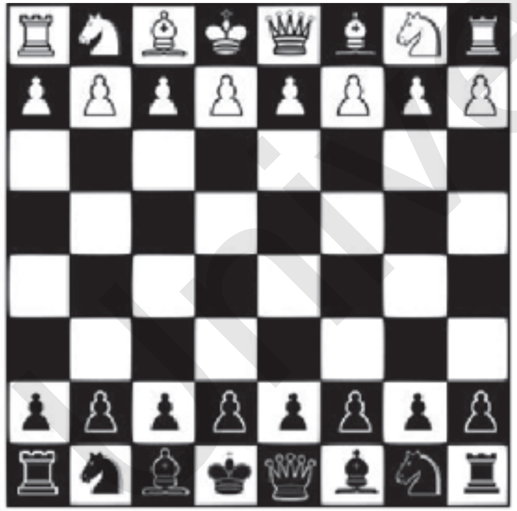
	Atlanta	Boston	Chicago	Los Angeles	New York City	San Francisco
Atlanta	-	1110	718	2175	888	2473
Boston	1110	-	992	2991	215	3106
Chicago	718	992	-	2015	791	2131
Los Angeles	2175	2991	2015	-	2790	381
New York City	888	215	791	2790	-	2901
San Francisco	2473	3106	2131	381	2901	-

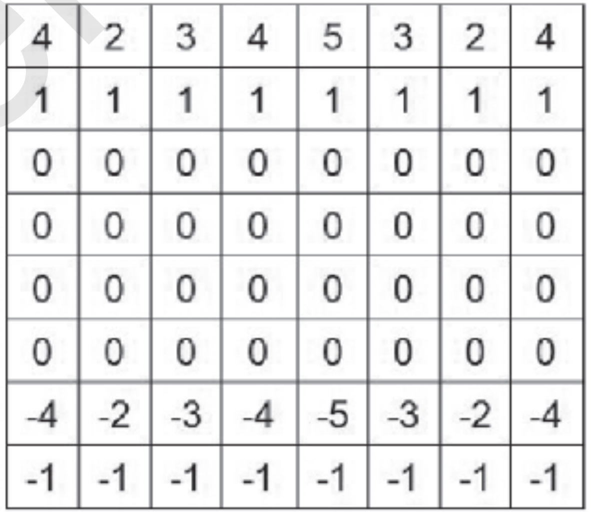
Note that only half of the table need be stored

Representation for Chess Playing Program



R	N	B	Q	K	B	N	R
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
R	N	B	Q	K	B	N	R





4	2	3	4	5	3	2	4
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-4	-2	-3	-4	-5	-3	-2	-4
-1	-1	-1	-1	-1	-1	-1	-1

Although the representation on the left is intuitive, the one on the right is more appropriate for computational problem solving.

Describing the Algorithms Needed

When solving a computational problem, either suitable existing algorithms may be found, or new algorithms must be developed.

For the MCGW problem, there are **standard search algorithms** that can be used.

For the calendar month problem, **a day of the week algorithm already exists**.

For the Traveling Salesman problem, **there are various (nontrivial) algorithms that can be utilized** for solving problems with tens of thousands of cities.

Finally, for the chess playing, since it is infeasible to look ahead at the final outcomes of every possible move, **there are algorithms that make a best guess at which moves to make**. Algorithms that work well in general but are not guaranteed to give the correct result for each specific problem are called ***heuristic algorithms***.

Program Implementation

Design decisions provide general details of the data representation and the algorithmic approaches for solving a problem. The details, however, do not specify which programming language to use, or how to implement the program. That is a decision for the implementation phase.

Since we are programming in Python, the implementation needs to be expressed in a syntactically correct and appropriate way, using the instructions and features available in Python.

Program Testing

Writing computer programs is difficult and challenging. As a result, **programming errors are pervasive, persistent and inevitable**.

Given this fact, **software testing is a crucial part of software development**. Testing is done incrementally as a program is being developed, when the program is complete, and when the program needs to be updated.

Truisms of Software Development

1. Programming errors are pervasive, persistent, and inevitable.

2. Software testing is an essential part of software development.

3. Any changes made in correcting a programming error should be fully understood as to why the changes correct the detected error.