# Introduction to Computers and Programming

CSL 101

# Logistics

- Instructors
  - Huzur Saran (saran@cse.iitd.ac.in)
  - Parag Singla (parags@cse.iitd.ac.in)
- Classes
  - M, Th: 8:00 am 9:20 am (VI LT 1) – Parag
  - M, Th: 2:00 pm 3:20 pm (IV LT 3) - Huzur
- Practical
  - Every Day – 11 am to 12:50 pm (CSC)
  - Every Day – 3 pm to 4:50 pm (CSC)
  - Based on Group

# Teaching Assistants

- A number of them
- Help with practical sessions
- 2 for each lab session
- Get familiarized!

# Logistics

- Website: http://www.cse.iitd.ac.in/~parags/courses/csl101/
- Class Mailing List
  - csl101@courses.iitd.ac.in
- Mail to TAs/Instructors
  - Should have CSL101 in the subject line
- References
  - Lectures Notes on Introduction to Computing
  - Online Material on Python Programming

# Computer Usage

- Access in the Computer Services Center
- Operating System – Linux (Unix like)
- Editor – vim

# Grading

- 2 Quizzes – 4% each
- Assignments (7) – 27%
- Minor 1, Minor 2 – 15% each
- Major – 35%

# Other Policies

- Attendance
  - Institute has a minimum attendance requirement
  - Should attend unless there is a reason not to do so
  - Difficult to do make up classes
- Late Assignments, Make-up Exams
  - Valid Certificate of illness

# Other Policies

- Attendance
  - Institute has a minimum attendance requirement
  - Should attend unless there is a reason not to do so
  - Difficult to do make up classes
- Late Assignments, Make-up Exams
  - Valid Certificate of illness
- Proxies!?

# Honor Code

- Expect everyone to follow an honor code of conduct
- All assignments must be done individually
  - Discussion is ok – mention in the submission
  - Actual solution to be written by self
- Copying/Cheating
  - May result in fail grade independent of performance

# Introduction to Computers and Programming

# Why Computer Science?

# Why Computer Science?

It's the best part of every one of us, the part that believes anything is possible. Things like a computer that can fit into a single room and hold millions of pieces of information..

- Movie Apollo 13

# Who is the course aimed at?

- First year students with no prior background in programming
  - Is it fair to say?

# What will the course address?

- Problem formulation in a precise and concise fashion independent of language
  - E.g. Create an efficient system for IIT course offering
  - Identify the courses, rooms, timings, students, faculty, various constraints (soft and hard) etc.
- Design of an algorithm – correct and efficient
  - How to divide courses in slots, satisfying various constraints (typically done on a paper)
- Write a Program
  - Create a program to actually do the mapping and assignment

# Other Examples

- Minimize the traffic congestion during holiday season (transportation)
- Minimize the heat dissipation from a silicon chip (VLSI - electrical)
- Satisfy the maximum number of mobile users given the fixed bandwidth (telecommunication)

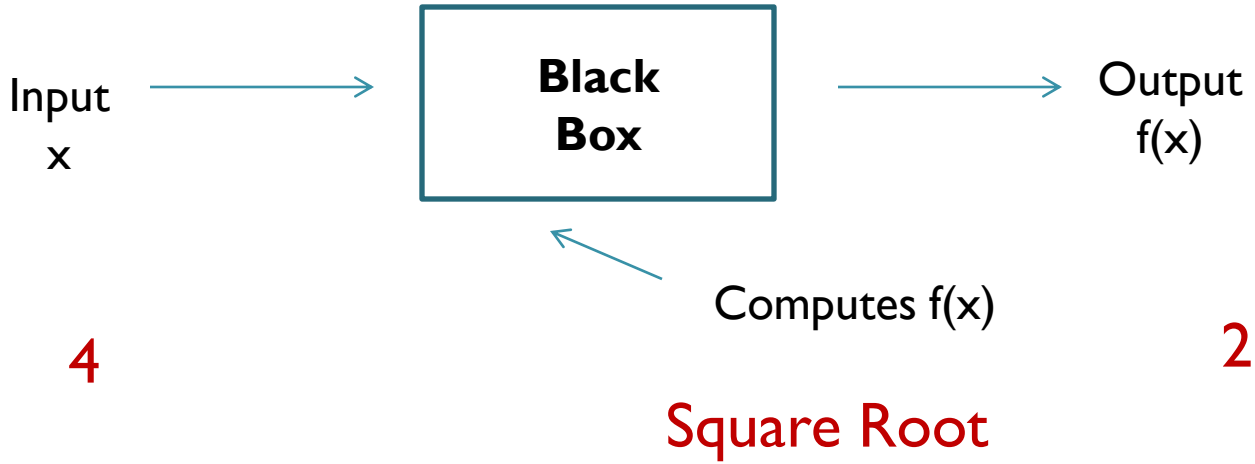
# Contents of the Course

- Basics of functional and imperative models
  - Recursive and iterative models
- Data directed programming
  - Strings, lists, arrays, trees, abstract data types
- Design and analysis of simple algorithms
  - Iterative style, divide and conquer, Dynamic programming based, randomized algorithms

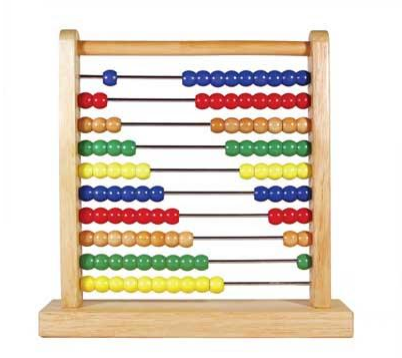
# What is Computer Science?

Computer Science: "Science of Computers"

A fundamental notion of "Computing"



# Various Computing Devices



Input/output has to be in a certain format

# Computing Devices

What is inside the black box?

Rules of Computing

Input
x

**Black Box**

Output
f(x)

# Algorithm

- Algorithm: A step by step procedure (sequence of lines) for solving a given computing problem.
- Example: How do you make tea?
  - Take the Pan
  - Heat up Water
  - Add Milk, Sugar and Tea
  - Boil
- How do you find 5*4?

# Model of Computation

- Need to specify the primitives
  - 5 * 4 = ?
  - 5 * 4 = 20 (multiplication as primitive)
  - 5 * 4 = 5 + 5 + 5 + 5 =20 (addition as primitive)
- Given a model of computation, write an algorithm to solve the problem

# Correctness and Efficiency

- Sound: An algorithm is sound if it always computes the "correct" answer
  - 5*4 = 21 – not sound
- Complete: An algorithm is complete if it calculates "an" answer for every given input
  - 5.1*4.1 = no answer – not complete
- Correctness = Soundness + Completeness

# Efficient Algorithm

- A notion of how much "space" and "time" algorithm uses given a model of computation

$5 * 4 (2+ 3 + 5) = 20 * 10 = 200$ (4 steps)

$5 * 4 (2 + 3 + 5) = 5 * 4 * 2 + 5 * 4 * 3 + 5 * 4 * 6 = 40 + 60 + 120 = 200$ (8 steps)

# Program

- Encoding of an algorithm in the syntax of a “programming language”
- Necessary so that machine can execute and given an output

# From Problem to Solution

Problem Specification (Concise Definition)

Should be true to original vague formulation

↓

Algorithm (Sequence of Steps)

Should be correct and efficient

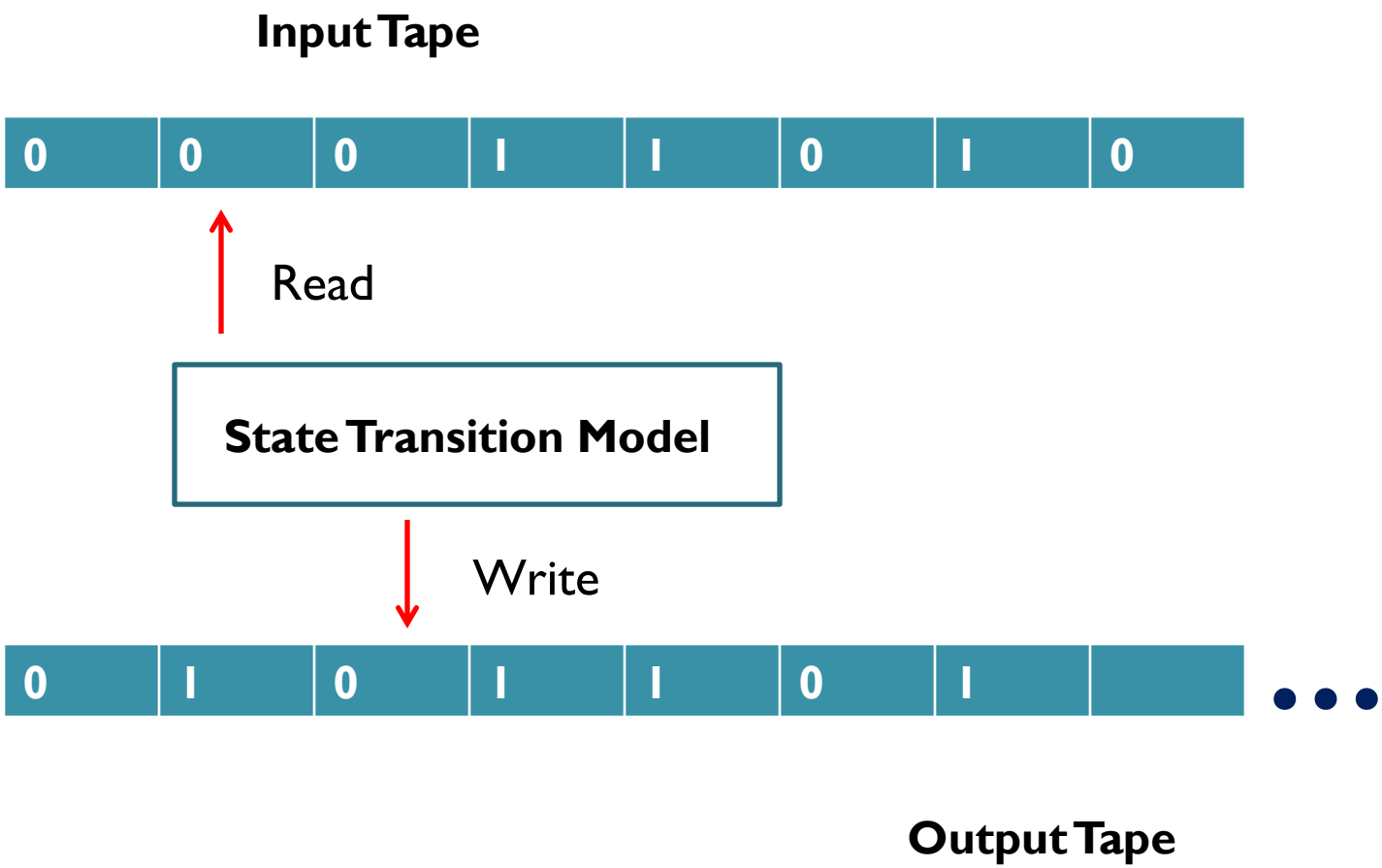
↓

Program (Actual Execution Happens)

Should execute exactly what algorithm specifies

# Turing Machine

Due to Alan Turing [1948]




Can Simulate any Modern Day Computer!

# Mathematical Preliminaries

# Sets

- A set is a collection of different objects
  - Example: Students in the class of CSL101
- Denoted by upper case letters – A, B, S

- Notation:

  A = {bb5120013, me2120796, bb5120021…}

- A and B are equal if they have the same elements

# Sets

- Two different ways to specify a set
  - Explicit Enumeration
    - {2, 4, 6, 8, 10}
  - Implicit Definition
    - {x | x is even positive integer less than equal to 10}
    - {x | x is an Integer and x % 2 = 0 and $1 \leq x \leq 10$}
    - A way to specify infinite sets

# Sets

- A set can have another set as an element
  - $S = \{\{a,b,c\}, d\}$
- $x \in S$ denotes x is an element of set S
- $A \subseteq B$ if $x \in B$ whenever $x \in A$
- Empty set $\Phi = \{\ \}$ (also known as null set)

# Sets

- Cardinality of a set S = |S| = number of elements in S

- Universe of discourse = U
  - Set of all possible values
  - Every set is a subset of U
- Properties
  - $\Phi \subseteq A, \ \forall A$ ($\forall$ - for all symbol)
  - $A \subseteq U, \ \forall A$

# Sets

- Union
  - $A \cup B = \{x \mid x \in A$ **or** $x \in B\}$
  - set of all the elements which belong to A or B
- Intersection
  - $A \cap B = \{x \mid x \in A$ **and** $x \in B\}$
  - Set of all the elements which belong to A and B
- Properties
  - $A \cup \Phi = A$
  - $A \cup U = U$
  - $A \cap \Phi = \Phi$
  - $A \cap U = A$

# Sets

- Cartesian Product of A and B
  - The set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$
  - $S = A \times B = \{ (a, b) \mid a \in A \text{ and } b \in B\}$

- $A^n$ is the set of all ordered tuples $(a_1, a_2, a_3, \ldots, a_n)$ such that $a_i \in A$
  - $A^n = A \times A \times A, \ldots, A$ (n times)

# Some Useful Sets

- N = Natural Numbers = {0,1,2,3,…}
- P = Positive Integers = {1,2,3,4,5..}
- Z = Integers = {… -2, -1, 0, 1, 2…}
- R = Real Numbers
- B = Boolean Set = {true, false}

# Relations and Functions

- Binary relation R over sets A, B is a subset of A x B
  - $R = \{(a_1, b_1), (a_2,b_2) \ldots (a_k,b_k)\}$
  - Example: Binary relations over natural numbers
  - $=, \neq, <, >$ , 'square'

# Relations and Functions

- Binary relation R over sets A, B is a subset of A x B
  - $R = \{(a_1, b_1), (a_2,b_2) \ldots (a_k,b_k)\}$
  - Example: Binary relations over natural numbers
  - $=, \neq, <, >$ , 'square'
- Example: $\{(0,0), (1,1), (2,2), (3,3) \ldots\}$ are the elements of $=$ relation

# Relations and Functions

- Binary relation R over sets A, B is a subset of A x B
  - $R = \{(a_1, b_1), (a_2,b_2) \ldots (a_k,b_k)\}$
  - Example: Binary relations over natural numbers
  - $=, \neq, <, >$ , 'square'
- Example: {(0,0), (1,1), (2,2), (3,3)…} are the elements of = relation
- Example: {(1,1),(2,4), (3,9)…} are elements of the 'square' relation

# Relations and Functions

- Binary relation R over sets A, B is a subset of A x B
  - $R = \{(a_1, b_1), (a_2,b_2) \ldots (a_k,b_k)\}$
  - Example: Binary relations over natural numbers
  - $=, \neq, <, >$ ,'square'
- Example: {(0,0), (1,1), (2,2), (3,3)…} are the elements of = relation
- Example: {(1,1),(2,4), (3,9)…} are elements of the 'square' relation
- N-ary relation defined over n sets $A_1, A_2, A_3, \ldots, A_n$ is a subset of $A_1 x A_2 x A_3 .. x A_n$

# Functions

- Function $R: A \rightarrow B$ is a binary relation over A, B such that
  - $\forall$ a $\in$ A, $\exists$ a unique b $\in$ B such that (a,b) $\in$ R
  - R(a) = b
- A is domain of R
- B is co-domain of R
- Range (R) = { b$\in$B | $\exists$ a, R(a) = b}

  ($\exists$ is there exists symbol)

# Functions

- ‘square’ is a function from $Z \rightarrow N$
- Addition and multiplication are functions over natural numbers.
  - $+: N \times N \rightarrow N$
  - $*: N \times N \rightarrow N$
- Any binary relation R on A, B can be seen as defining a function
  - $f_R$: $A \times B \rightarrow \{true, false\}$
  - $f_R(a, b)$ = true if $(a, b) \in R$, false otherwise
  - Example: $=, \neq, <, >$
  - (5=3) is false, whereas (5=5) is true

# Mathematical Induction

# Inductive Argument

- Generalizes a property from a finite set of instances to the whole set


© 3poD * www.ClipartOf.com/16528

# Mathematical Induction – A Tool

- Defining functions and relations over natural numbers
- Proving correctness of theorems defined over natural numbers as arguments
- Useful for proving correctness of programs

# Defining Functions

- Function f
- Basis Step:
  - $f(0) = k$
- Induction Step:
  - $f(n+1) = g(f, n)$

second order function

A finite definition structure!

# Defining Functions

- Factorial function
  - fac: $N \longrightarrow N$ (written as fac(n)=n!)
- Basis Step:
  - $fac(0) = 1$
- Induction Step:
  - $fac(n+1) = (n+1) * fac(n), \forall\ n \geq 0$

# Composition

- Let R, S be binary relations defined on A x A
  - $R \circ S = \{(a,c) \mid \exists\, b, (a,b) \in R \text{ and } (b, c) \in S\}$
- N-fold composition
- Basis:
  - $R^1 = R$
- Induction Step:
  - $R^{n+1} = R \circ R^n$
- Useful for finding nodes n-hopes away in a graph

# Proof by Induction

- Version 1:
- To Prove:
  - A property P holds for all natural numbers.
- Basis Step:
  - Property holds for n = 0
- Induction Step:
  - If P holds for an arbitrary n ($n \geq 0$), then P holds for n+1

Induction Hypothesis

A finite proof structure!

# Example

- To Prove:
  - $g(n) = (n^3 + 2n)$ is divisible by 3
- Proof:
- Basis Step:
  - $n = 0 \Rightarrow g(0) = 0$ is divisible by 3.
- Induction Hypothesis:
  - $g(n)$ is divisible by 3 $(n \geq 0)$

# Example

- Induction Step:
  - $g(n+1) = (n+1)^3 + 2 * (n+1)$

$$= (n^3 + 1 + 3n^2 + 3n) + (2n + 2)$$

$$= (n^3 + 2n) \quad + \quad (3n^2 + 3n + 1 + 2)$$

$$= (n^3 + 2n) \quad + \quad 3(n^2 + 3n + 1)$$

Divisible by 3 due to induction hypothesis

Divisible by 3

$\Rightarrow g(n+1)$ is divisible by 3

# Example

How??

- Induction Step:
  - $g(n+1) = (n+1)^3 + 2 * (n+1)$

$$= (n^3 + 1 + 3n^2 + 3n) + (2n + 2)$$

$$= (n^3 + 2n) + (3n^2 + 3n + 1 + 2)$$

$$= (n^3 + 2n) + 3(n^2 + 3n + 1)$$

Divisible by 3 due to induction hypothesis

Divisible by 3

$\Rightarrow g(n+1)$ is divisible by 3

# Proof by Induction

- Version 2:
- To Prove:
  - A property P holds for all natural numbers.
- Basis Step:
  - Property holds for $n = k$ ($k \geq 0$)
- Induction Step:
  - If P holds for an arbitrary n ($n \geq k$), then P holds for n+1

Version 1 – A special case ($k = 0$)

# Example

- To Prove:
  - Two stamp denominations of Rs 3 and Rs 5.
  - Any stamp value $\geq 8$ can be constructed using them.
  - Formally: $\forall n \geq 8, n = 3i + 5j$, for some $i, j \geq 0$
- Proof:
- Basis Step:
  - $n = 8 \Rightarrow 8 = 3*1 + 5*1$ $(i = 1, j = 1)$
- Induction Hypothesis:
  - $n = 3i + 5j,\ n \geq 8, i, j \geq 0$

# Example

- Induction Step:
- $n + 1 = 3i + 5j + 1$
- If $(j > 0)$
  - $3i + 5j + 1 = 3i + 5(j - 1) + 5 + 1$
    $= 3i + 3*2 + 5(j-1)$
    $= 3i + 6 + 5(j-1)$
    $= 3(i + 2) + 5(j-1)$
- If $(j = 0)$
  - $3i + 5j + 1 = 3i + 1$ ($i \geq 3$, since $n \geq 8$)
    $= 3(i -3) + 9 + 1$
    $= 3(i-3) + 5*2$
    $= 3(i-3) + 5j$ (where $j=2$)

# Example

How??

- $n + 1 = 3i + 5j + 1$
- If $(j > 0)$
  - $3i + 5j + 1 = 3i + 5(j - 1) + 5 + 1$

    $= 3i + 3*2 + 5(j-1)$

    $= 3i + 6 + 5(j-1)$

    $= 3(i + 2) + 5(j-1)$
- If $(j = 0)$
  - $3i + 5j + 1 = 3i + 1$ ($i \geq 3$, since $n \geq 8$)

    $= 3(i -3) + 9 + 1$

    $= 3(i-3) + 5*2$

    $= 3(i-3) + 5j$ (where $j=2$)

# Example

How??

- $n + 1 = 3i + 5j + 1$
- If $(j > 0)$
  - $3i + 5j + 1 = 3i + 5(j - 1) + 5 + 1$

    $= 3i + 3*2 + 5(j-1)$

    $= 3i + 6 + 5(j-1)$

    $= 3(i + 2) + 5(j-1)$

Base Case

- If $(j = 0)$
  - $3i + 5j + 1 = 3i + 1$ $(i \geq 3$, since $n \geq 8)$

    $= 3(i -3) + 9 + 1$

    $= 3(i-3) + 5*2$

    $= 3(i-3) + 5j$ (where $j=2$)

# Proof by Induction

- Version 3:
- To Prove:
  - A property P holds for all natural numbers.
- Basis Step:
  - Property holds for $n = k$ ($k \geq 0$)
- Induction Step:
  - If P holds for all m, $k \leq m \leq n$ for an arbitrary n ($n \geq k$), then P holds for $n+1$

Version 1, 2 – Special cases

# Example

- Let $F_0=0$, $F_1=1$, $F_2=1$ … be the Fibonacci sequence where $F_n=F_{n-1}+F_{n-2}$
  - $\Phi = (1 + \text{sqrt}(5))/2$
- To Prove:
  - $F_n \leq \Phi^{n-1}$ for $n > 0$
- Proof:
- Basis:
  - $F_1 = 1 = \Phi^0$

# Example

- Induction Hypothesis:
  - $F_m \leq \Phi^{m-1}, \forall\ 1 \leq m \leq n$
- Induction Step:
  - $F_{n+1} = F_n + F_{n-1}$

    $\leq \Phi^{n-1} + \Phi^{n-2}$

    $\leq \Phi^{n-2}(1 + \Phi)$
  - $1 + \Phi = 1 + (1 + \text{sqrt}(5))/2 = (3 + \text{sqrt}(5))/2$
  - $\Phi^2 = (1+\text{sqrt}(5))^2/4 = (6+2*\text{sqrt}(5))/4=$

    $= (3 + \text{sqrt}(5))/2 = 1 + \Phi$
  - $\Rightarrow F_{n+1} \leq \Phi^{n-2}(1 + \Phi) = \Phi^{n-2} * \Phi^2 = \Phi^n$

# Example

- Induction Hypothesis:
  - $F_n \leq \Phi^{n-1}, \forall\ 1 \leq m \leq n$
- Induction Step:
  - $F_{n+1} = F_n + F_{n-1}$

    $\leq \Phi^{n-1} + \Phi^{n-1}$

    $\leq \Phi^{n-2}(1 + \Phi)$
  - $1 + \Phi = 1 + (1 + \text{sqrt}(5))/2 = (3 + \text{sqrt}(5))/2$
  - $\Phi^2 = (1+\text{sqrt}(5))^2/4 = (6+2*\text{sqrt}(5))/4=$

    $= (3 + \text{sqrt}(5))/2 = 1 + \Phi$
  - $\Rightarrow F_{n+1} \leq \Phi^{n-2}(1 + \Phi) = \Phi^{n-2} * \Phi^2 = \Phi^n$

How??

# Proof by Induction

- Version 0:
- A property P can be thought of as defining a set S such that
  - $S = \{x \mid x \text{ satisfies } P\}$
- To Prove:
  - A property P holds for the set of natural numbers.
- Basis Step:
  - $0 \in S$
- Induction Step:
  - If $\underline{n \in S}$, then $n+1 \in S$

# Programming Language

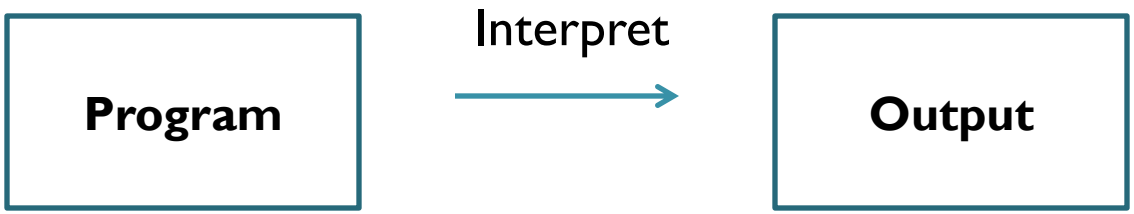
# Syntax, Semantics, Debugging

- Syntax
  - The rules to write the program
  - a = 5 + 3
  - print a
  - print(a)
- Semantics
  - What does the program actually accomplish?
  - The program prints the result of adding 5 and 3
- Debugging
  - Correcting the syntax and semantics

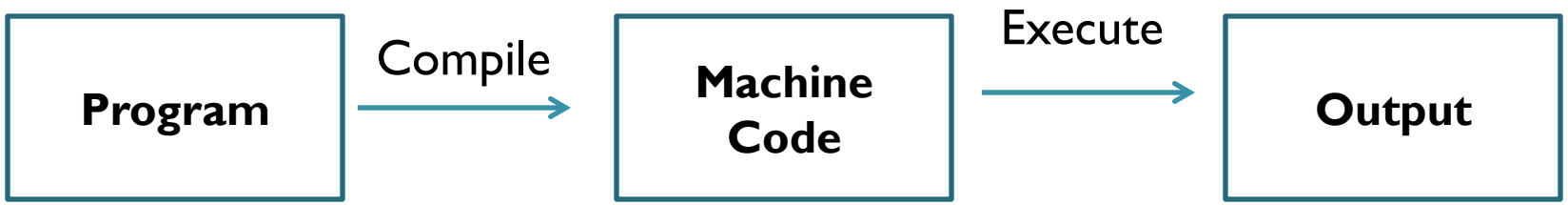
# Python

- A high level language (vs low level)
  - C, C++, Java
- Interpreted (vs compiled)

Interpreted Language




Compiled Language



# Modes of Operation

- Interactive Mode
  - [ Terminal@] python
  - >>>>print 5
  - 5
- Script mode
  - Create “myfile.py”
    - a=5
    - print a
  - [ Terminal@] python myfile.py
    - 5

# Printing a String

- >>>> print ‘Hello World’
- >>>> Hello World

# Variables and Constants

- Variables
  - Can hold any value
  - Example: a, b, x2, y4, myvar
- Constant
  - Has a fixed value
  - 5, 6, ‘Hello’, 5.4

# Data Types

- int
  - Integer valued numbers
- float
  - Floating point numbers (decimal numbers)
- str
  - Characters, Strings
- bool
  - {True, False}
- >>>> a=3
- >>>> type(a)
- >>>> <type, 'int'>

Loosely typed language

# Operators

- >>>> 2+3
- >>>> 5
- >>>> 5 * 7
- >>>> 35
- >>>> 8/5
- >>>> 1
- >>>> 8%5
- >>>>> 3

# Precedence

- >>>> 5*8+4/2=42
- Precedence order:
  - /, *, +,-
- Use of Brackets
  - >>>> (5*8)+(4/2)

# Assignment

- >>>> n = 2+3
- >>>> print n
- >>>> 5
- >>>> a = 2
- >>>> b =3
- >>>> c = a + b
- >>>> print c
- >>>> 5
- >>>>c = c + 1
- print c

# Assignment

- >>>> n = 2+3
- >>>> print n
- >>>> 5
- >>>> a = 2
- >>>> b =3
- >>>> c = a + b
- >>>> print c
- >>>> 5
- >>>>c = c + 1
- print c
- 6

# Program

- Write a program to print the number of hours, minutes and seconds in the year 2012.

# Input Function

- >>>> n = input(‘Please input a number:’)
  5
- >>>> print ‘n is ’, n
- >>>> n is 5

# Comparison Operators

- >, <, <=,  >=, ==, !=
- Return ‘True’ or ‘False’ (Boolean type)
- >>>>a = 5
- >>>> a == 5
- True

# Boolean Type

- Two values: True, False
- >>>> a = (5 > 3)
- >>>> print a
- >>>> True
- Boolean Operators
- and
  - >>>> a = (5 > 3) and (5 > 6)
  - >>>> print a
  - >>>>False
- or
  - >>>> a = (5 > 3) or (5 > 6)
  - >>>> print a
  - >>>>True

# Boolean Type

- not
  - >>>> not (5 == 5)
  - >>>>True
  - >>>> a = 6
  - >>>> b = not (a == 6)
  - >>>> print b

# Boolean Type

- not
  - >>>> not (5 == 5)
  - >>>>True
  - >>>> a = 6
  - >>>> b = not (a == 6)
  - >>>> print b
  - >>>> False

# Conditionals

- Defined over Boolean variables
- If < condition > :
  - <Code>
- else:
  - <Code>
- Indentation is the key

# Conditionals

- Example Program (script mode)

- a = input(‘Input a number:’)
- if (a == 3) :
  - print ‘a is equal to 3’
- else:
  - print ‘a is not equal to 3’

# Program

- Write a program which takes as input an year and prints the number of hours, minutes and seconds in it (Remember to correctly handle the case for a leap year).