Chapter 1 Introduction

This chapter addresses the question "What is computer science?" We begin by introducing the essence of computational problem solving via some classic examples. Next, computer algorithms, the heart of computational problem solving, are discussed. This is followed by a look at computer hardware (and the related issues of binary representation and operating systems) and computer software (and the related issues of syntax, semantics, and program translation). The chapter finishes by presenting the process of computational problem solving, with an introduction to the Python programming language.

Motivation

Computing technology has changed, and is continuing to change the world. Essentially every aspect of life has been impacted by computing. Computing related fields in almost all areas of study are emerging.

Various Computational-Related Fields		
Computational Biology	Computational Medicine	Computational Journalism
Computational Chemistry	Computational Pharmacology	Digital Humanities
Computational Physics	Computational Economics	Computational Creativity
Computational Mathematics	Computational Textiles	Computational Music
Computational Materials Science	Computational Architecture	Computational Photography
Computer-Aided Design	Computational Social Science	Computational Advertising
Computer-Aided Manufacturing	Computational Psychology	Computational Intelligence

What is Computer Science?

Computer science is fundamentally about computational problem solving.

Programming and computers are only tools in the field of computing. The field has tremendous breadth and diversity. Areas of study include:

- Programming language Design
- Systems Programming
- Computer Architecture
- Human–Computer Interaction
- Robotics
- Artificial Intelligence
- Software Engineering
- Database Management / Data Mining
- Computer Networks
- Computer Graphics
- Computer Simulation
- Information Security

Computational Problem Solving

Two things that are needed to perform computational problem solving:

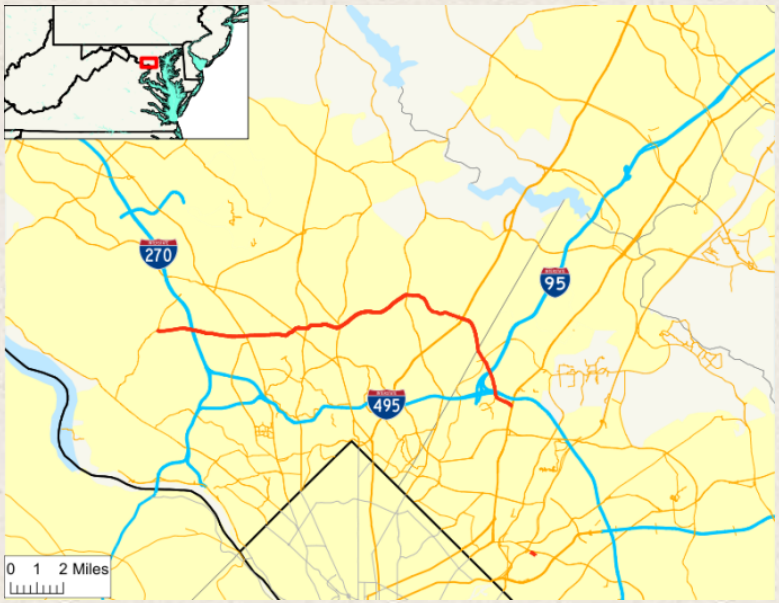
- **a representation** that captures all the relevant aspects of the problem
- **an algorithm** that solves the problem by use of the representation

Thus, computational problem solving finds a solution within a representation that translates into a solution for what is being represented.

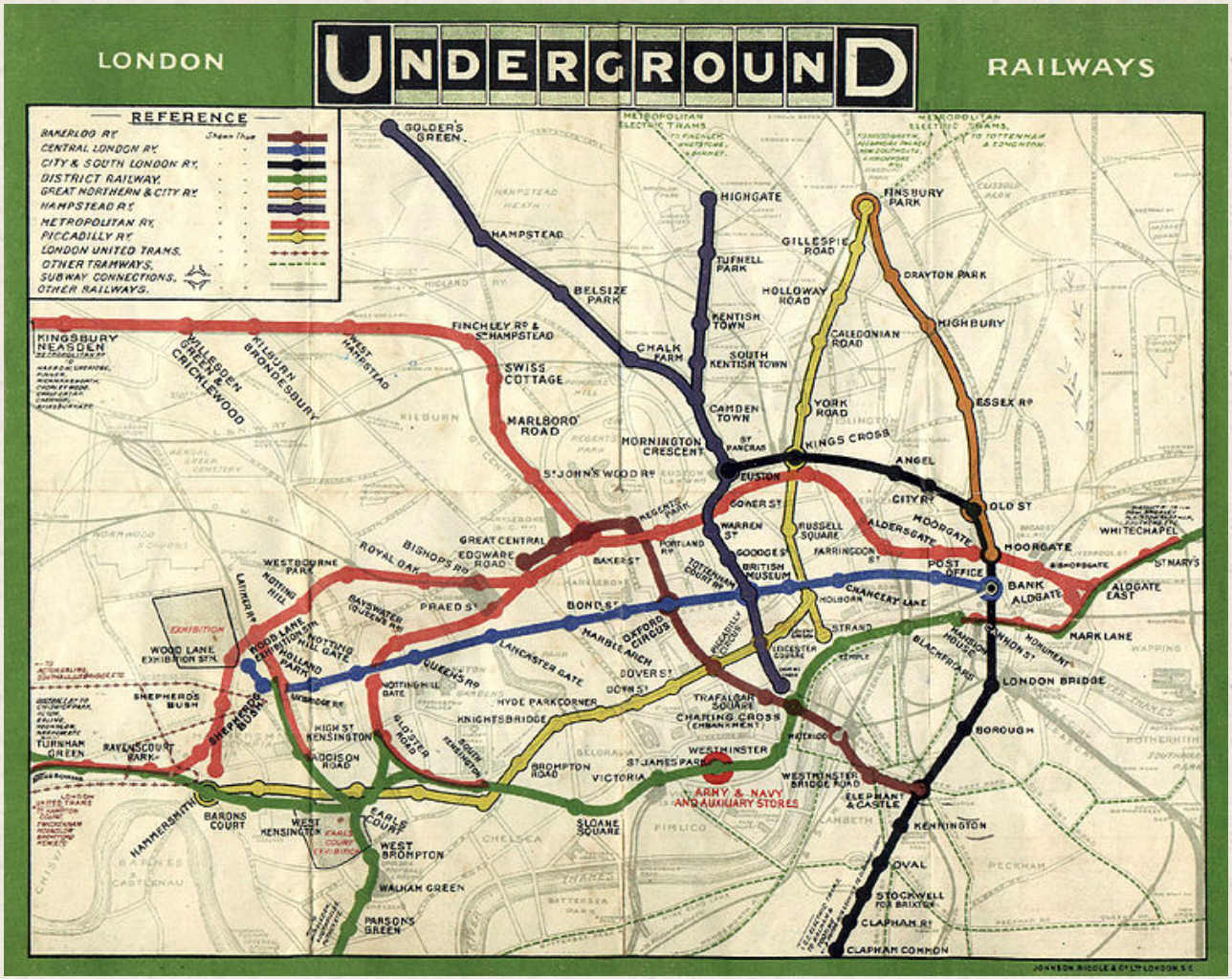
The Use of Abstraction in Computational Problem Solving

A representation that leaves out detail of what is being represented is a form of abstraction.

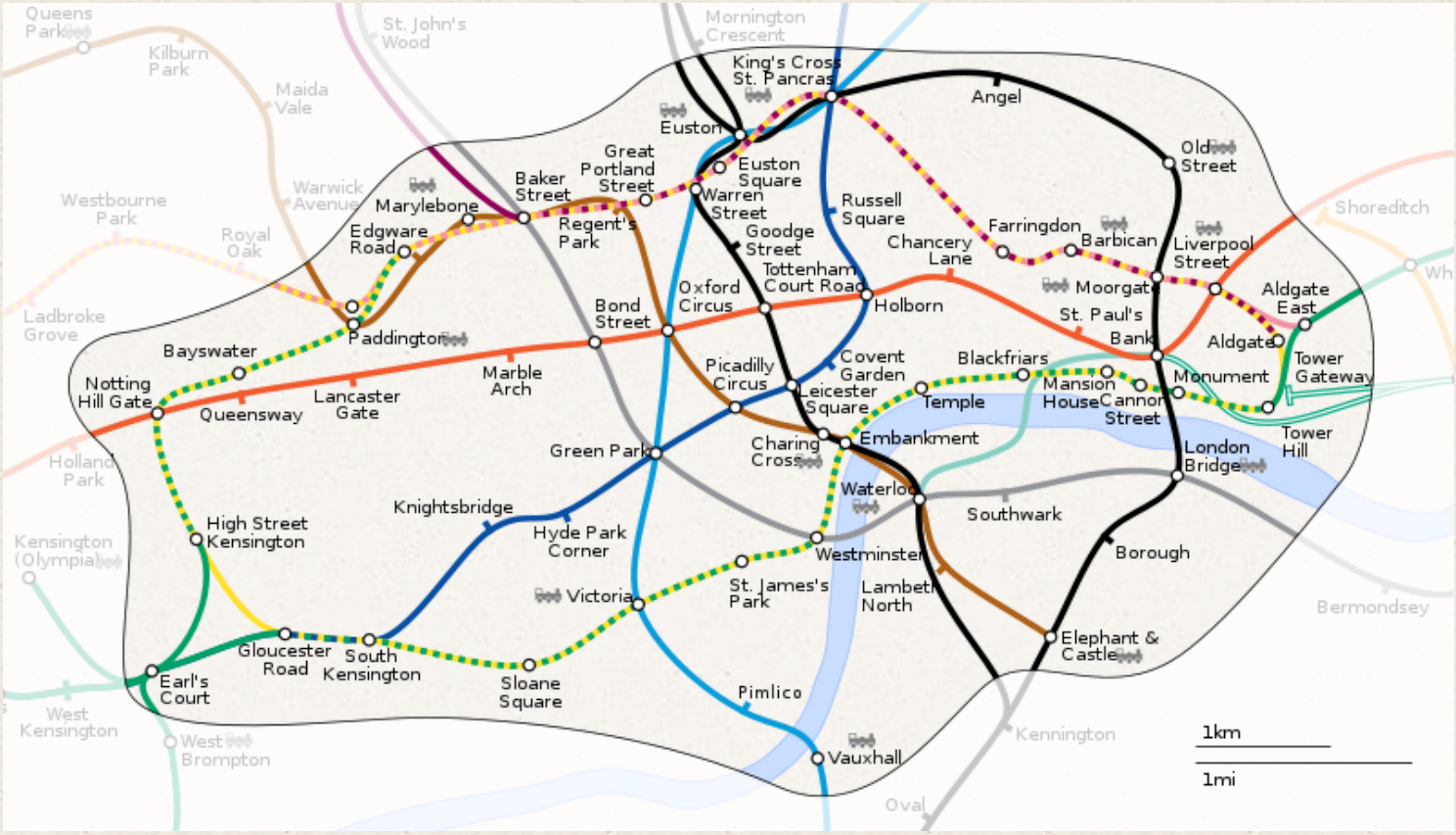
Abstraction is prevalent in the everyday world. For example, **maps are abstractions**.



Below is the original 1908 map of the London Underground (Subway).



Below is a more abstract, but topologically correct map of the London Underground subway system showing the bends and curves of each track.



This map contains too much information for its purpose – to find out where each subway line leads, and where the connections are between lines.

Below is a more abstract representation of the subway system, developed by Harry Beck in 1931. The track lines are straightened out where the track curves are irrelevant for subway riders. This is a simpler, easier to read, and thus a better representation for its purpose.

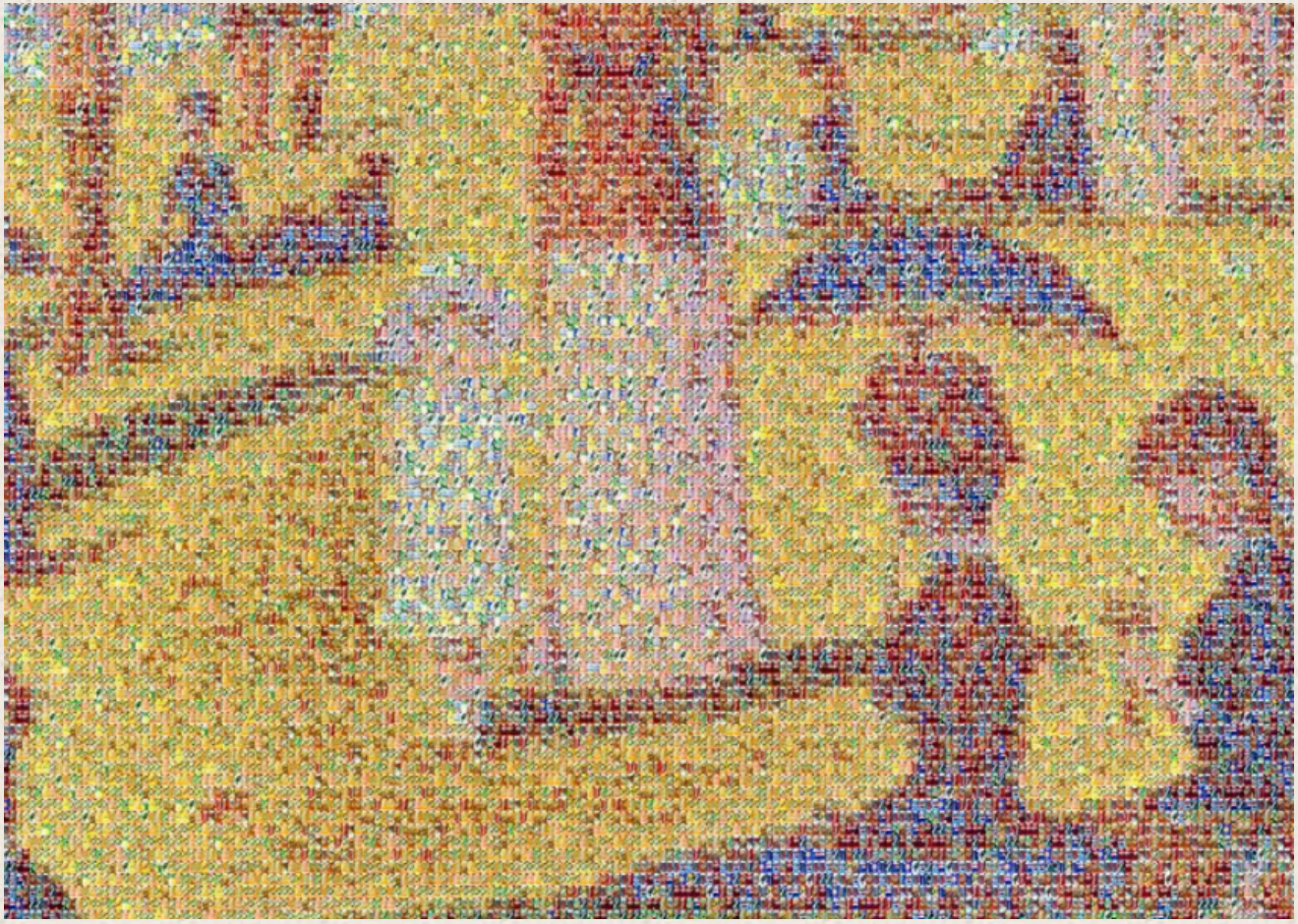
This particular abstraction is still in use today.

Washington D.C. Metro Map

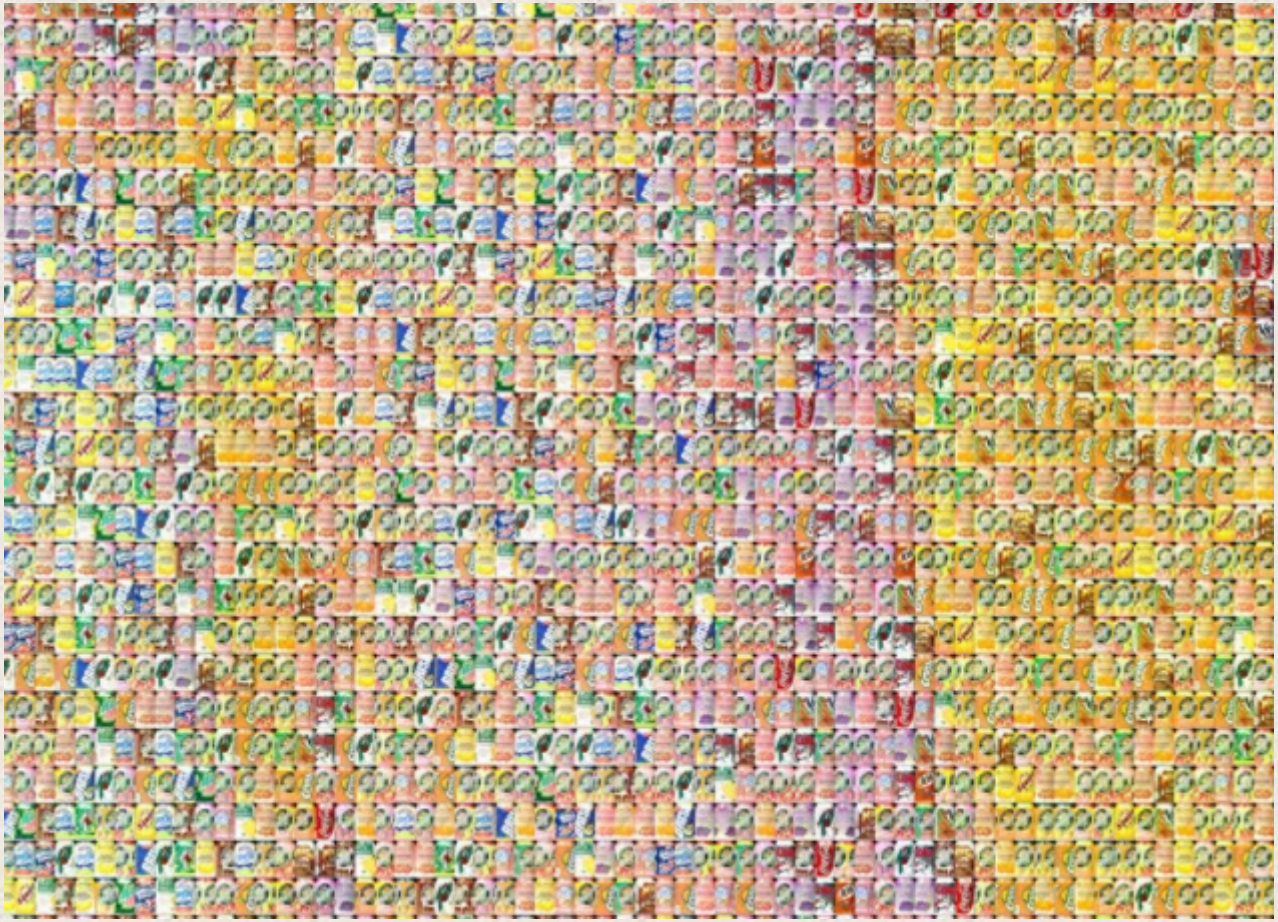
An interesting form of abstraction is the creative work of Chris Jordan, which allows the viewer to control the degree of abstraction of the work.

Cans Seurat, 2007 60x92“ Copyright Chris Jordan

Depicts 106,000 aluminum cans, the number used in the U.S. every thirty seconds.



Clicking on the image causes the picture to zoom in, showing more detail.
(Copyright Chris Jordan)



Zooming in closer …

(Copyright Chris Jordan)

(Copyright Chris Jordan)

More Images

Abstraction in Computing

Abstraction is intrinsic to computing and computational problem solving.

- The concept of "1s" and "0s" in digital computing is an abstraction.
  Digital information is actually represented as a high or low voltage levels, magnetic particles oriented one of two ways, pits on an optical disk, etc.

- Programming languages are an abstraction.
  The instructions and data of a computer program is an abstract representation of the underlying machine instructions and storage.

- Programming design involves the use of abstraction.
  Programs are conceptualized as various modules that work together.

Man, Cabbage, Goat, Wolf Problem

A man lives on the east side of a river. He wishes to bring a cabbage, a goat, and a wolf to a village on the west side of the river to sell. However, his boat is only big enough to hold himself, and either the cabbage, goat, or wolf. In addition, the man cannot leave the goat alone with the cabbage because the goat will eat the cabbage, and he cannot leave the wolf alone with the goat because the wolf will eat the goat. How does the man solve his problem?

There is a simple algorithmic approach for solving this problem by simply trying all possible combinations of items that may be rowed back and forth across the river.

Trying all possible solutions is referred to as a ***brute force approach***.

What would be an appropriate representation for this problem? Whatever representation we use, only the aspects of the problem that are relevant for its solution need to be represented. Should we include in the representation the …

- color of the boat?
- name of the man?
- width of the river?

The only information relevant for this problem is where each particular item is at each step in the problem solving. Therefore, by the use of **abstraction**, we define a representation that captures only this needed information.

For example, we could use a sequence to indicate where each of the objects currently are,

man	cabbage	goat	wolf	boat	village
[e a s t ,	w e s t ,	e a s t ,	w e s t ,	e a s t ,	w e s t]

where it is understood that the first item in the sequence is the location of the man, the second item in the sequence is the location of the cabbage, etc.

Note that the village is always on the west side of the river – it doesn't move. Its location is fixed and therefore does not need to be represented.

Also, the boat is always in the same place as the man. So representing the location of both the man and the boat is redundant information. The relevant, **minimal representation** is given below (replacing "east"/"west" with single characters).

man	cabbage	goat	wolf
[E ,	W ,	E ,	E]

The actual problem is to determine how the man can row objects across the river, with certain constraints on which pairs of objects cannot be left alone.

The computational problem is to find a way to convert the representation of the **start state** of the problem, when all the object are on the east side of the river,

man	cabbage	goat	wolf
[E ,	E ,	E ,	E]

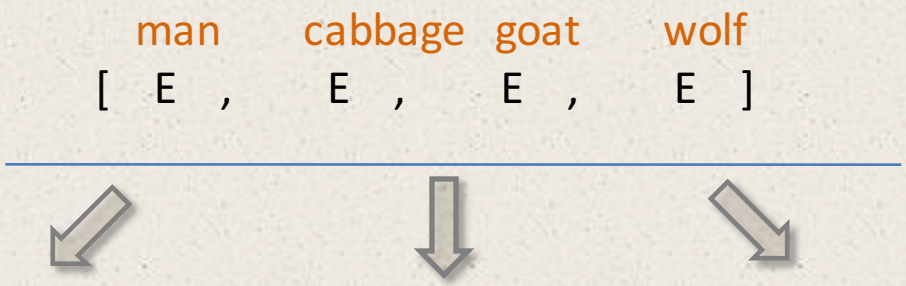
to the **goal state**, when all objects on the west side of the river,

man	cabbage	goat	wolf
[W ,	W ,	W ,	W]

with the constraint that certain **invalid states** should never be used.

Thus, in a computational problem solving approach, a solution is found within the representation used, which must translate into a solution of the actual problem.

For example, from the start state, there are three possible moves that can be made, only one of which results in a valid state.



[W , W , E , E]

Man rows cabbage across

INVALID STATE

Goat left alone with wolf

[W , E , W , E]

Man rows goat across

VALID STATE

Cabbage left alone with wolf

[W , E , E , W]

Man rows wolf across

INVALID STATE

Cabbage left alone with goat

We check if the new problem state is the goal state. If so, then we solved the problem in one step! (We know that cannot be so, but the algorithmic approach that we are using does not.)

man cabbage goat wolf

[E , E , E , E] **START STATE**

[W , E , W , E]

Man rows goat across

Is goal state [W , W , W , W] ? **No**

Therefore we continue searching from the current state.

Since the man can only row across objects on the same side of the river as himself, and the only object currently with him is the goat, **there are only two possible moves from here**,

man cabbage goat wolf

[W , E , W , E]

INTERMEDIATE STATE

[E , W , E , E]

Man rows back alone

VALID STATE

Cabbage left alone on west side

Man, Goat and Wolf on east side

[E , E , E , E]

Man rows goat across

VALID STATE (but unproductive)

Man and all others on east side

Previously in this state. It is the start state. Therefore no progress made!

This would continue until the goal state is reached,

Thus, **the computational problem of generating the goal state from the start state translates into a solution of the actual problem** since each transition between states has a corresponding "real-world" action – of the man rowing across the river with (or without) a particular object.

The Importance of Algorithms

MAY 2012

Sun	Mon	Tues	Wed	Thur	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

As another example computational problem, suppose that you needed to write a program that displays a calendar month for any given month and year.

The representation of this problem is rather straightforward. Only a few values are needed:

- the month and year
- number of days in each month
- names of the days of the week
- day of the week that the first day of the month falls on

The month and year, number of days in a month, names of the days of the week can be easily handled. **The less obvious part is how to determine the day of the week that a particular date falls on**.

How would you do that?

> *Start with a known day of the week for a past date and calculate forward from there?*

That would not be a very efficient way of solving the problem.

Since calendars are based on cycles, there must be a more direct method for doing this. No matter how good a programmer you may be, without knowledge of the needed algorithm, you could not write a program that solves the problem.

The Limits of Computational Problem Solving

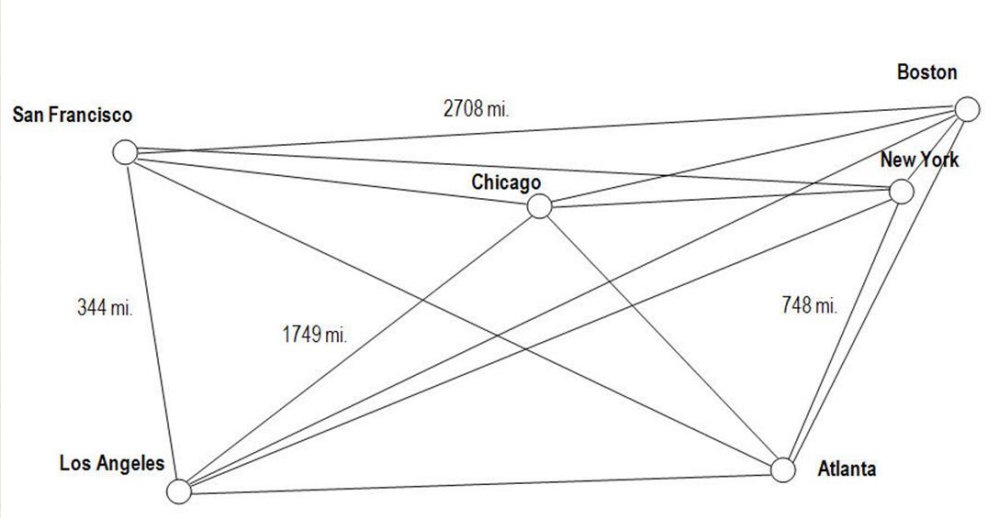
Once an algorithm for a given problem is developed or found, an important question is "**Can a solution to the problem be found in a reasonable amount of time**?"

"But aren't computers very fast, and getting faster all the time?"

Yes, but some problems require an amount of time to compute a solution that is astronomical compared to the capabilities of current computing devices.

A classic problem in computer science that demonstrates this is the **Traveling Salesman problem**.

The Traveling Salesman Problem



A salesman needs to visit a set of cities. He wants to find the shortest route of travel, starting and ending at any city for a given set of cities. What route should he take?

The algorithm for solving this problem is a simple one. Determine the lengths of all possible routes that can be taken, and find the shortest one – a **brute force approach**. **The computational issue, therefore, is for a given set of cities, how many possible routes are there to consider?**

If we consider a route to be a specific sequence of names of cities, then **how many permutations of that list are there?**

New York, Boston, Chicago, San Francisco, Los Angeles, Atlanta

New York, Boston, Chicago, San Francisco, Atlanta, Loa Angeles

New York, Boston, Chicago, Los Angeles, San Francisco, Atlanta

etc.

Mathematically, **the number of permutations for n entities is n!** (n factorial).

How big a number is that for various number of cities?

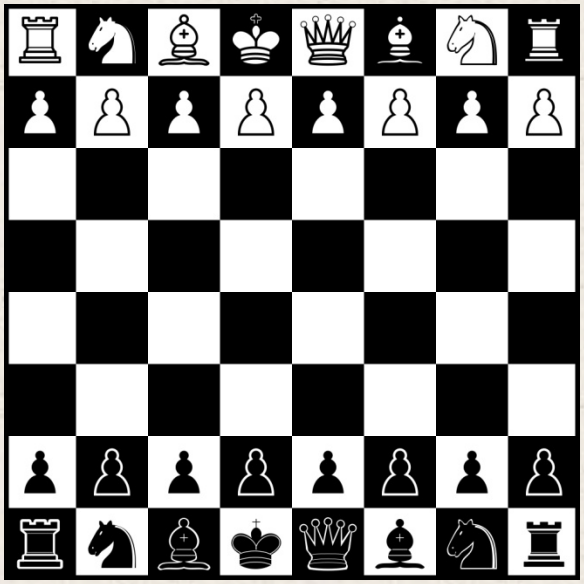
Below are the number of permutations (and thus the number of routes) there are for varies numbers of cities:

Ten Cities	10!	3,628, 800 (over three million)
Twenty Cities	20!	2,432,902,008,176,640,000
Fifty Cities	50!	over 10^{64}

If we assume that a computer could compute the routes of one million cities per second:

- for **twenty cities**, it would take **77,000 years**
- for **fifty cities**, it would take **longer than the age of the universe!**

The Game of Chess



When a computer plays chess against a human opponent, both have to "think ahead" to the possible outcomes of each move it may make.

Therefore, a brute force approach can also be used for a computer playing a game of chess of "looking ahead" at all the possible moves that can be made, each ending in a win, loss, or draw. It can then select the move each time leading to the most number of ways of winning.

(Chess masters, on the other hand, only think ahead a few moves, and "instinctively" know the value of each outcome.)

There are approximately 10^{120} possible chess games that can be played. This is related to the average number of look-ahead steps needed for deciding each move.

There are approximately,

10^{80} atoms in the observable universe

and an estimated

3×10^{90} grains of sand to fill the universe solid

Thus, there are *more possible chess games that can be played than grains of sand to fill the universe solid!*

Therefore, for problems such as this and the Traveling Salesman problem in which a brute-force approach is impractical to use, **clever and more efficient problem-solving methods must be discovered that find either an** exact **or an** approximate solution to the problem.

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS:

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS: 1. False,

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS: 1. False 2. (d),

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS: 1. False, 2. (d) 3. algorithm,

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS: 1. False, 2. (d), 3. algorithm, 4. abstraction,

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS: 1. False, 2. (d), 3. algorithm, 4. abstraction 5. (b),

Self-Test Questions

1. A good definition of computer science is "the science of programming computers." (TRUE/FALSE)

2. Which of the following areas of study are included within the field of computer science?
   (a) Software engineering
   (b) Database management
   (c) Information security
   (d) All of the above

3. In order to computationally solve a problem, two things are needed: a representation of the problem, and an _____ that solves it.

4. Leaving out detail in a given representation is a form of _____.

5. A "brute-force" approach for solving a given problem is to:
   (a) Try all possible algorithms for solving the problem.
   (b) Try all possible solutions for solving the problem.
   (c) Try various representations of the problem.
   (d) All of the above

6. For which of the following problems is a brute-force approach practical to use?
   (a) Man, Cabbage, Goat, Wolf problem
   (b) Traveling Salesman problem
   (c) Chess-playing program
   (d) All of the above

ANSWERS: 1. False, 2. (d), 3. algorithm, 4. abstraction, 5. (b), 6. (a)

Computer Algorithms

An algorithm is a finite number of clearly described, unambiguous "doable" steps that can be systematically followed to produce a desired result for given input in a finite amount of time (that is, it eventually terminates).

The word "algorithm" is derived from the ninth-century Arab mathematician, Al-Khwarizmi who worked on "written processes to achieve some goal."

Algorithms and Computers: A Perfect Match

Computer algorithms are central to computer science. They provide step-by-step methods of computation that a machine can carry out.

Having high-speed machines (computers) that can consistently follow a given set of instructions provides a reliable and effective means of realizing computation. However, **the computation that a given computer performs is only as good as the underlying algorithm used.**

Because computers can execute a large number of instructions very quickly and reliably without error, algorithms and computers are a perfect match!

Euclid's Algorithm
One of the Oldest Known Algorithms

Euclid's Algorithm is an algorithm for computing the greatest common denominator (GCD) of two given integers. It is one of the oldest numerical algorithms still in common use.

1. Assign M the value of the larger of the two values.

2. Divide M by N, call the remainder R.

3. If R is not 0, then assign M the value of N, assign N the value of R, and go to step 2.

4. The greatest common divisor is N.

Example Use

Finding the GCD of 18 and 20

1. **Assign M the value of the larger of the two values, and N the smaller.**
   M ← 20 N ← 18

Example Use

Finding the GCD of 18 and 20

1. Assign M the value of the larger of the two values, and N the smaller.
   M ← 20 N ← 18

2. **Divide M by N, call the remainder R.**
   M/N = 20 / 18 = 1, with R ← 2

Example Use

Finding the GCD of 18 and 20 (**first time** through, **second time** through)

1. Assign M the value of the larger of the two values, and N the smaller.
   M ← 20 N ← 18

2. Divide M by N, call the remainder R.
   M/N = 20 / 18 = 1, R ← 2

3. **If R is not 0, assign M the value of N, assign N the value of R, and go to step 2.**
   M ← 18, N ← 2.

Example Use

Finding the GCD of 18 and 20 (**first time** through, **second time** through)

1. Assign M the value of the larger of the two values, and N the smaller.
   M ← 20 N ← 18

2. **Divide M by N, call the remainder R.**
   M/N = 18 / 2 = 9, R ← 0

3. If R is not 0, assign M the value of N, assign N the value of R, and go to step 2.
   M ← 18, N ← 2.

Example Use

Finding the GCD of 18 and 20 (**first time** through, **second time** through)

1. Assign M the value of the larger of the two values, and N the smaller.
   M ← 20 N ← 18

2. Divide M by N, call the remainder R.
   M/N = 20 / 18 = 1, with R ← 2
   M/N = 18 / 2 = 9, with R ← 0

3. **If R is not 0, assign M the value of N, assign N the value of R, and go to step 2.**
   R is 0. Therefore, proceed to step 4.

Example Use

Finding the GCD of 18 and 20 (**first time** through, **second time** through)

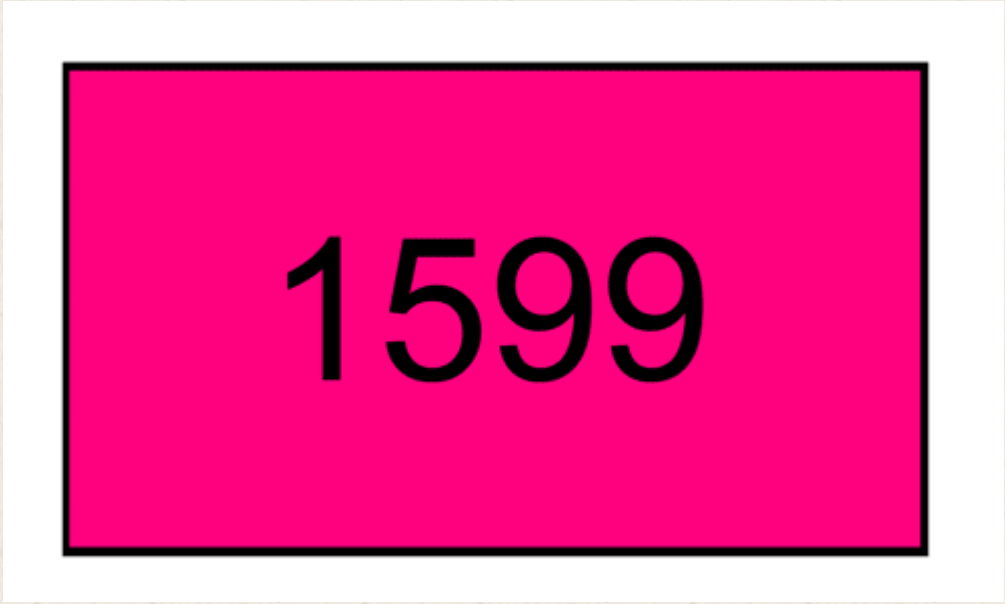
1. Assign M the value of the larger of the two values, and N the smaller.
   M ← 20 N ← 18

2. Divide M by N, call the remainder R.
   M/N = 20 / 18 = 1, with R ← 2
   M/N = 18 / 2 = 9, with R ← 0

3. If R is not 0, assign M the value of N, assign N the value of R, and go to step 2.
   R = 2. Therefore, M ← 18, N ← 2. Go to step 2.
   R is 0. Therefore, proceed to step 4.

4. **The greatest common divisor is N.**
   GCD is 2 (the value of N)

Animation of Euclid's Algorithm



Following is an example algorithm for determining the day of the week for any date between January 1, 1800 and December 31, 2099

To determine the day of the week for a given **month**, **day**, and **year**:

1. Let **century_digits** be equal to the first two digits of the year.
2. Let **year_digits** be equal to the last two digits of the year.
3. Let **value** be equal to **year_digits +** floor(**year_digits** / 4)
4. If **century_digits** equals 18, then add 2 to **value**, else
   if **century_digits** equals 20, then add 6 to **value**.
5. If the **month** is equal to January and **year** is not a leap year,
   then add 1 to **value**, else,

   if the **month** is equal to February and the **year** is a leap year, then
   add 3 to **value**; if not a leap year, then add 4 to **value,** else,

   if the **month** is equal to March or November, then add 4 to **value**, else,

   if the **month** is equal to April or July, then add 0 to **value**, else,

   if the **month** is equal to May, then add 2 to **value**, else,

   if the **month** is equal to June, then add 5 to **value**, else,

   if the **month** is equal to August, then add 3 to **value**, else,

   if the **month** is equal to October, then add 1 to **value**, else,

   if the **month** is equal to September or December, then add 6 to **value**,
6. Set **value** equal to (**value** + **day**) mod 7.
7. If **value** is equal to 1, then the day of the week is Sunday; else

   if **value** is equal to 2, day of the week is Monday; else

   if **value** is equal to 3, day of the week is Tuesday; else

   if **value** is equal to 4, day of the week is Wednesday; else

   if **value** is equal to 5, day of the week is Thursday; else

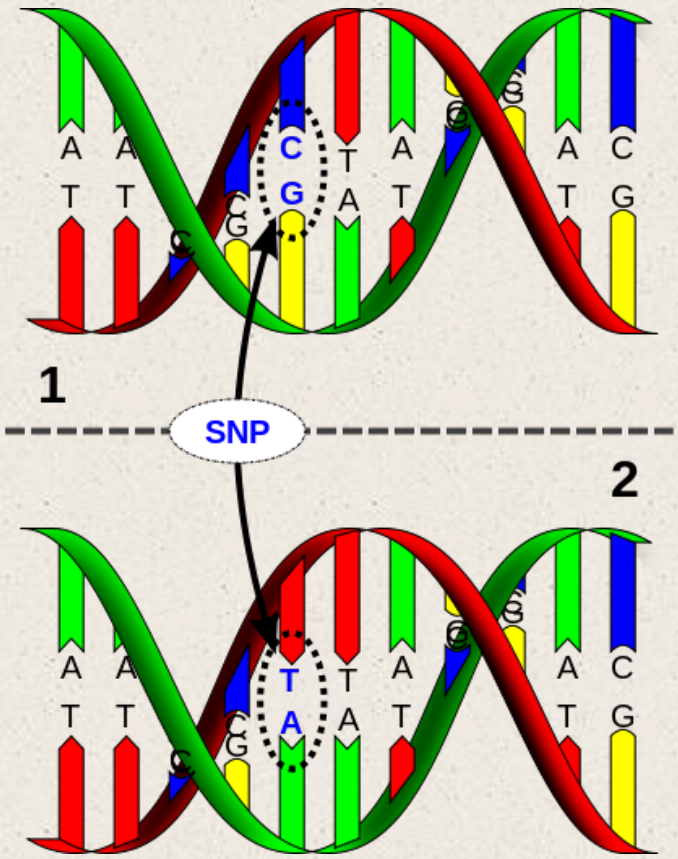
   if **value** is equal to 6, day of the week is Friday; else

   if **value** is equal to 0, day of the week is Saturday

Notable Contemporary Algorithms

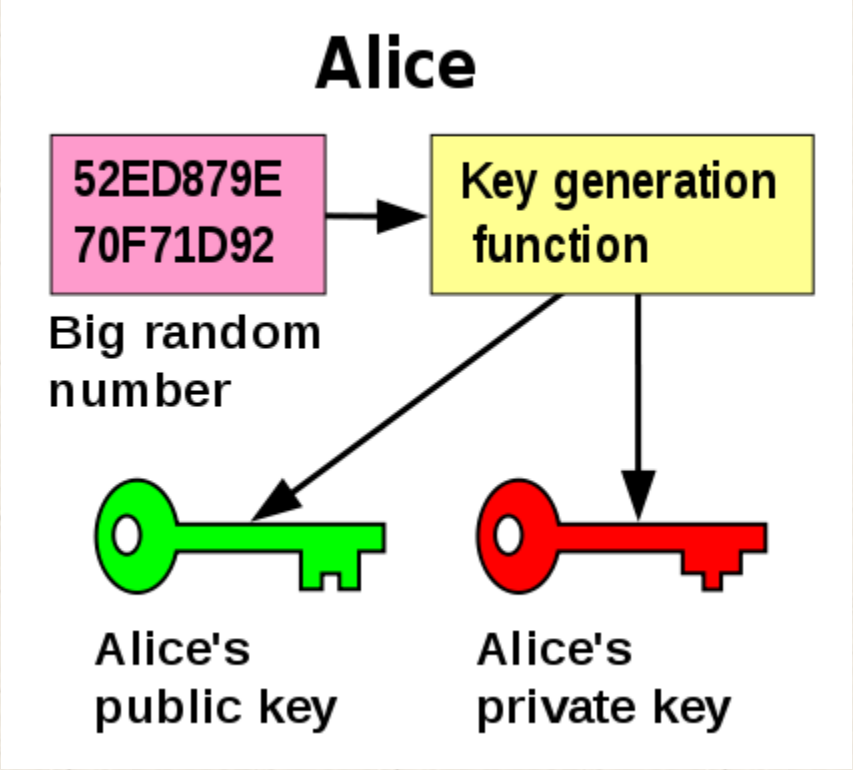
BLAST Algorithm

The **sequencing of the human genome** was dependent on the development of fast, efficient algorithms for comparing and matching DNA sequences.



RSA Algorithm

The **RSA algorithm** is the basis of **public key encryption**. It is requires the factorization of large prime numbers to break, which for large enough primes, is considered impossible. It is the method used for secure web communication.



Self-Test Questions

1. Which of the following are true of an algorithm?
   (a) Has a finite number of steps
   (b) Produces a result in a finite amount of time
   (c) Solves a general problem
   (d) All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a "perfect match" because: (Select all that apply.)
   (a) Computers can execute a large number of instructions very quickly.
   (b) Computers can execute instructions reliably without error.
   (c) Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
   (a) A means of sorting any list of numbers
   (b) Directions for getting from your home to a friend's house
   (c) A means of finding the shortest route from your house to a friend's house.

ANSWERS:

Self-Test Questions

1. Which of the following are true of an algorithm?
   (a) Has a finite number of steps
   (b) Produces a result in a finite amount of time
   (c) Solves a general problem
   (d) All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a "perfect match" because: (Select all that apply.)
   (a) Computers can execute a large number of instructions very quickly.
   (b) Computers can execute instructions reliably without error.
   (c) Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
   (a) A means of sorting any list of numbers
   (b) Directions for getting from your home to a friend's house
   (c) A means of finding the shortest route from your house to a friend's house.

ANSWERS: 1. (d),

Self-Test Questions

1. Which of the following are true of an algorithm?
   (a) Has a finite number of steps
   (b) Produces a result in a finite amount of time
   (c) Solves a general problem
   (d) All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a "perfect match" because: (Select all that apply.)
   (a) Computers can execute a large number of instructions very quickly.
   (b) Computers can execute instructions reliably without error.
   (c) Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
   (a) A means of sorting any list of numbers
   (b) Directions for getting from your home to a friend's house
   (c) A means of finding the shortest route from your house to a friend's house.

ANSWERS: 1. (d) 2. False,

Self-Test Questions

1. Which of the following are true of an algorithm?
   (a) Has a finite number of steps
   (b) Produces a result in a finite amount of time
   (c) Solves a general problem
   (d) All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a "perfect match" because: (Select all that apply.)
   (a) Computers can execute a large number of instructions very quickly.
   (b) Computers can execute instructions reliably without error.
   (c) Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
   (a) A means of sorting any list of numbers
   (b) Directions for getting from your home to a friend's house
   (c) A means of finding the shortest route from your house to a friend's house.

ANSWERS: 1. (d), 2. False, 3. (a,b)

Self-Test Questions

1. Which of the following are true of an algorithm?
   (a) Has a finite number of steps
   (b) Produces a result in a finite amount of time
   (c) Solves a general problem
   (d) All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a "perfect match" because: (Select all that apply.)
   (a) Computers can execute a large number of instructions very quickly.
   (b) Computers can execute instructions reliably without error.
   (c) Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
   (a) A means of sorting any list of numbers
   (b) Directions for getting from your home to a friend's house
   (c) A means of finding the shortest route from your house to a friend's house.

ANSWERS: 1. (d), 2. False, 3. (a,b) 4. Friday

Self-Test Questions

1. Which of the following are true of an algorithm?
   (a) Has a finite number of steps
   (b) Produces a result in a finite amount of time
   (c) Solves a general problem
   (d) All of the above

2. Algorithms were first developed in the 1930–1940s when the first computing machines appeared. (TRUE/FALSE)

3. Algorithms and computers are a "perfect match" because: (Select all that apply.)
   (a) Computers can execute a large number of instructions very quickly.
   (b) Computers can execute instructions reliably without error.
   (c) Computers can determine which algorithms are the best to use for a given problem.

4. Given that the year 2016 is a leap year, what day of the week does April 15th of that year fall on? Use the algorithm in Figure 1-8 for this.

5. Which of the following is an example of an algorithm? (Select all that apply.)
   (a) A means of sorting any list of numbers
   (b) Directions for getting from your home to a friend's house
   (c) A means of finding the shortest route from your house to a friend's house.

ANSWERS: 1. (d), 2. False, 3. (a,b) 4. Friday, 5. (a,c)

Computer Hardware

Computer hardware comprises the physical part of a computer system. It includes the all-important components of the **central processing unit** (CPU) and **main memory**. It also includes peripheral components such as a keyboard, monitor, mouse, and printer.

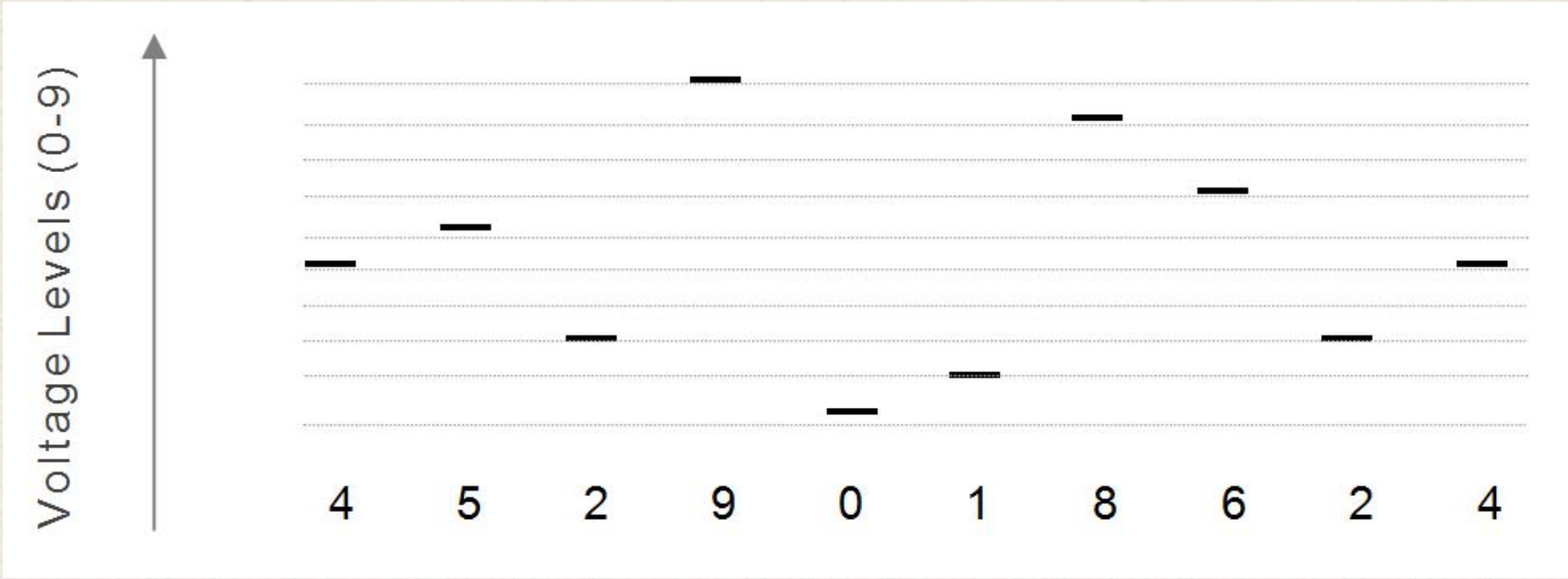
Digital Computing: Its all About Switches

It is essential that computer hardware be reliable and error free. If the hardware gives incorrect results, then any program run on that hardware may give incorrect results as well.

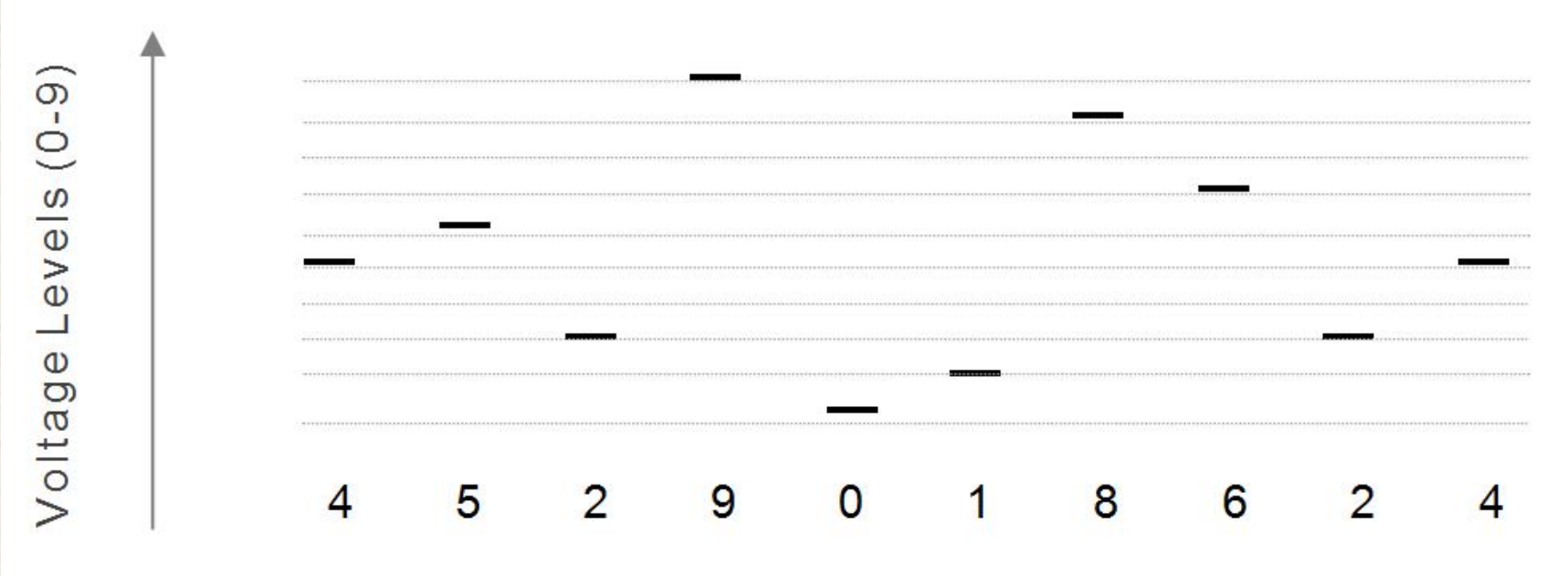
The key to developing reliable systems is to keep the design as simple as possible. In digital computing, all information is represented as a series of digits, in which each digit is either "0" or "1".

Decimal Digitalization

In electronic computing, digital values are represented by discrete voltage levels. Suppose that computers represented numbers as we are used to, in base ten.



In electronic computers, each digit is represented by a different voltage level. The more voltage levels (digits) that the hardware must utilize and distinguish between, the more complex the hardware becomes to design. This results in greater chance of hardware design errors.



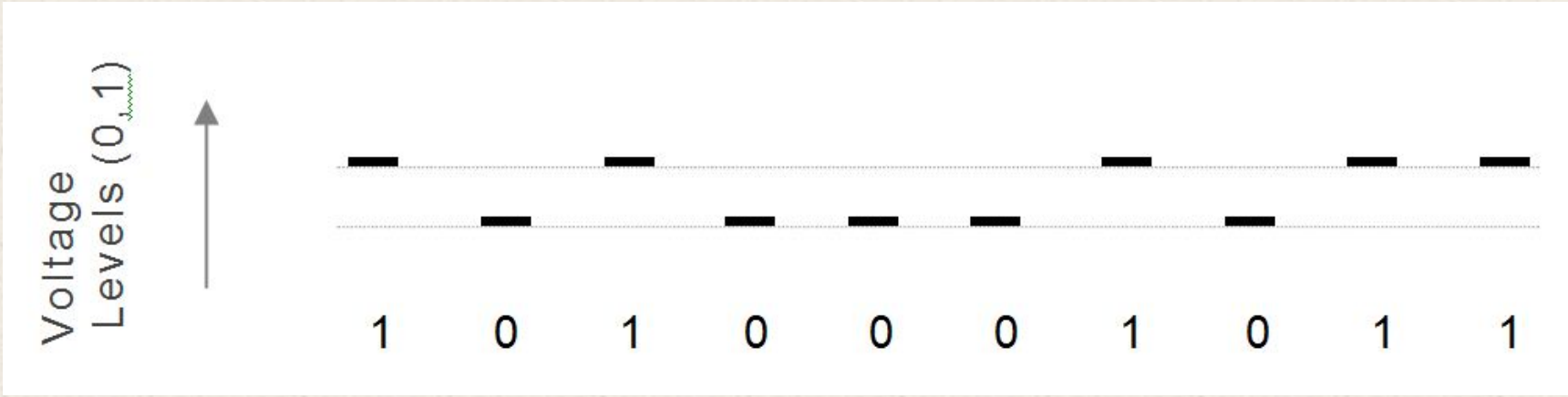
Information Theory

The **Fundamental Theorem of Information Science** of **Claude Shannon** (known as the "Father of Information Theory") states that **all information can be represented by the use of only two symbols**, e.g., 0 and 1.

This is referred to as **binary representation.**

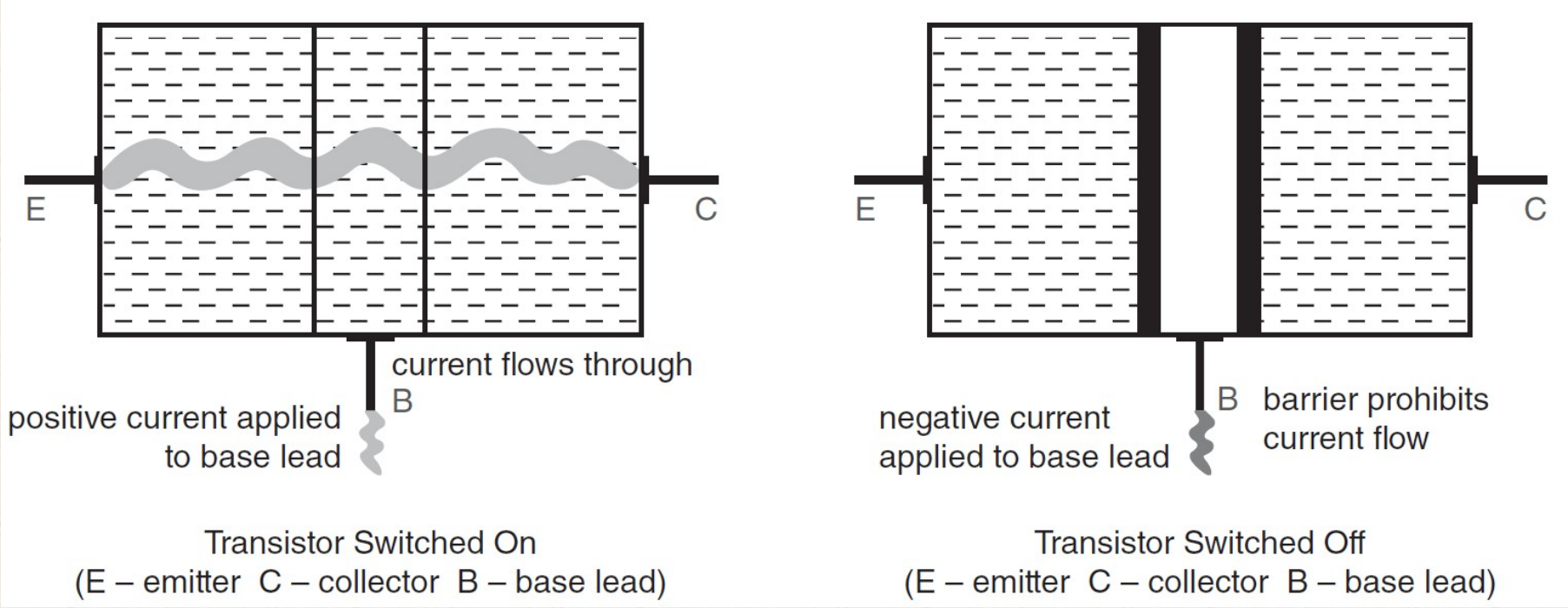
Binary Digitalization

In a binary representation, each digit can be one of only two possible values, similar to a light switch that can be either on or off. Computer hardware, therefore, is based on the use of simple electronic "on/off" switches called **transistors** that can be switched at essentially the speed of light.



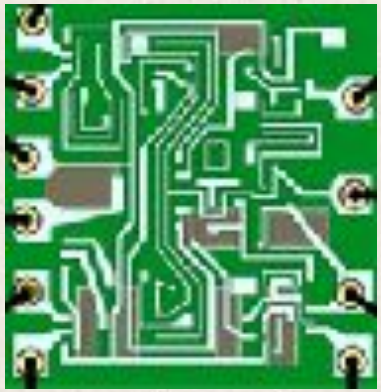
The Transistor

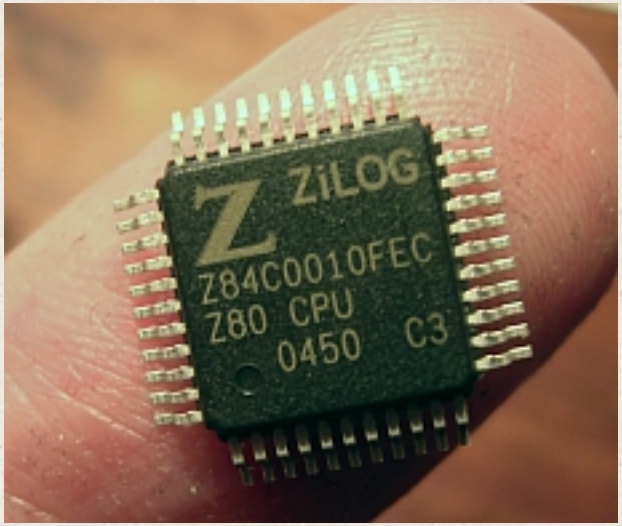
The transistor, an electronic switching component developed in 1947, revolutionized electronics. Its invention it what has allowed for all of the dramatic advances in computing technology that we continue to see today. (For the historical development of the transistor, see Chapter 12.)



Integrated Circuits

Integrated circuits ("chips"), **the building blocks of computer hardware**, are comprised of millions or even billions of transistors. It solved the problem of how to "wire together" this multitude of components. (For the historical development of the integrated circuit, see Chapter 12.)





The Binary Number System

For representing numbers, any base (radix) can be used. For example, **in base 10**, there are ten possible digits (0, 1, ..., 9), in which column values are a power of ten:

10,000,000	1,000,000	100,000	10,000	1,000	100	10	1
10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
						9	9 = **99**

For representing numbers in base 2, there are two possible digits (0, 1) in which column values are a power of two:

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	0	0	0	1	1

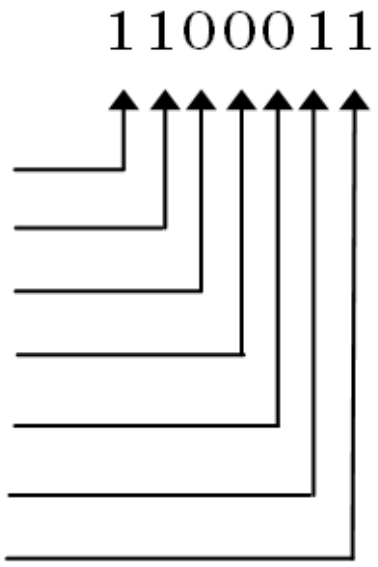
0 + 64 + 32 + 0 + 0 + 0 + 2 + 1 = **99**

Although values represented in base 2 are significantly longer than those in base 10, **binary representation is used in digital computing because of the resulting simplicity of hardware design**

Bits and Bytes

Each **bi**nary **d**igit is referred to as a **bit**. A group of (usually) eight bits is called a **byte**. Converting a base ten number to base two involves the successive division of the number by 2.

```
99/2   =   49, with remainder 1
49/2   =   24, with remainder 1
24/2   =   12, with remainder 0
12/2   =   6,  with remainder 0
6/2    =   3,  with remainder 0
3/2    =   1,  with remainder 1
1/2    =   0,  with remainder 1
```



Fundamental Hardware Components

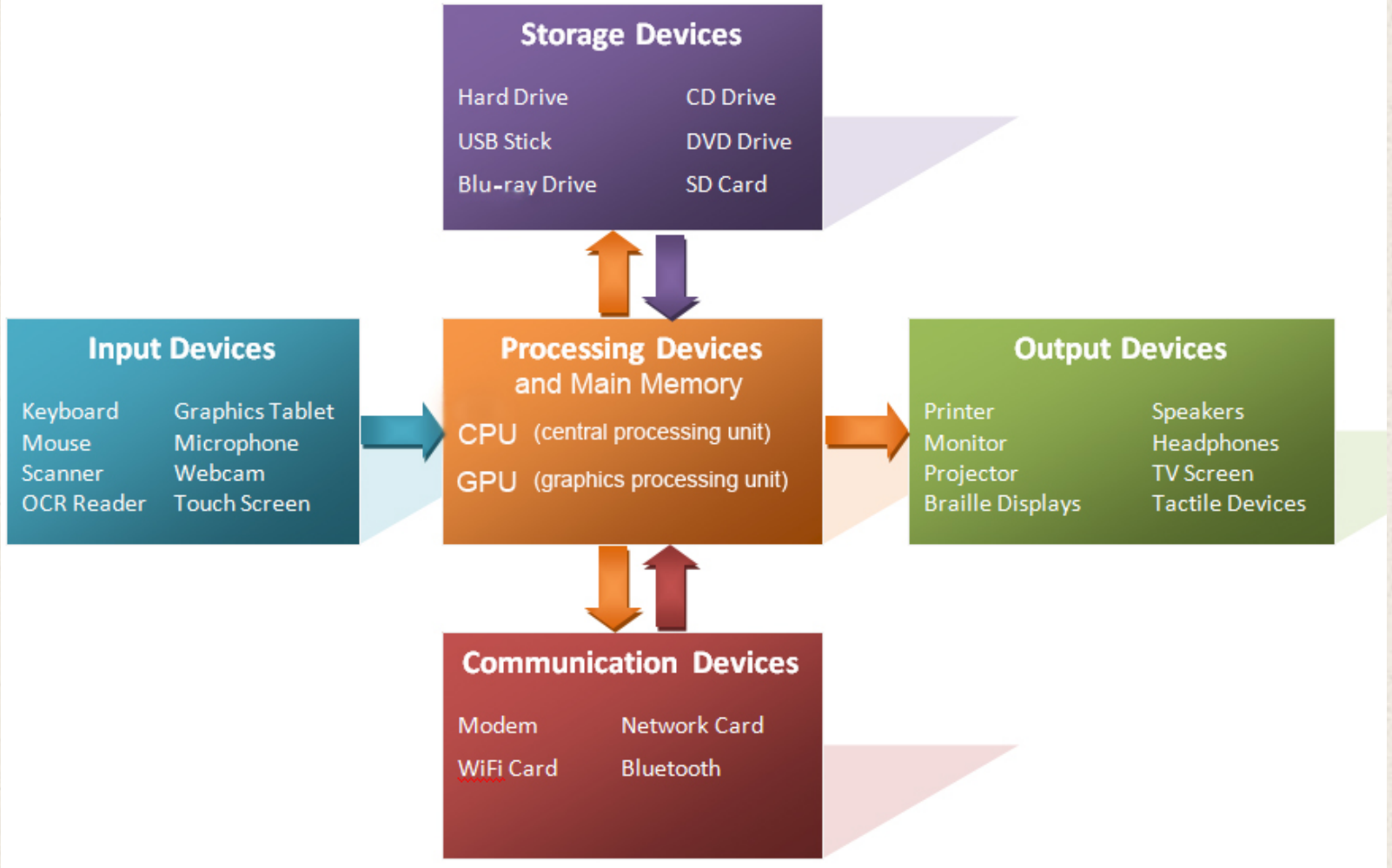
Central processing unit (CPU) – the "brain" of a computer system. Interprets and executes instructions.

Main memory – is where currently executing programs reside. It is *volatile*, the contents are lost when the power is turned off.

Secondary memory – provides long-term storage of programs and data. N*onvolatile*, the contents are retained when power is turned off. Can be magnetic (hard drive), optical (CD or DVD), or flash memory (USB drive).

Input/output devices – mouse, keyboard, monitor, printer, etc.

Buses – transfer data between components within a computer system. System bus (between CPU and main memory)


Storage Devices
Hard Drive
CD Drive
USB Stick
DVD Drive
Blu-ray Drive
SD Card
Input Devices
Keyboard
Graphics Tablet
Mouse
Microphone
Scanner
Webcam
OCR Reader
Touch Screen
Processing Devices
and Main Memory
CPU (central processing unit)
GPU (graphics processing unit)
Output Devices
Printer
Speakers
Monitor
Headphones
Projector
TV Screen
Braille Displays
Tactile Devices
Communication Devices
Modem
Network Card
WiFi Card
Bluetooth

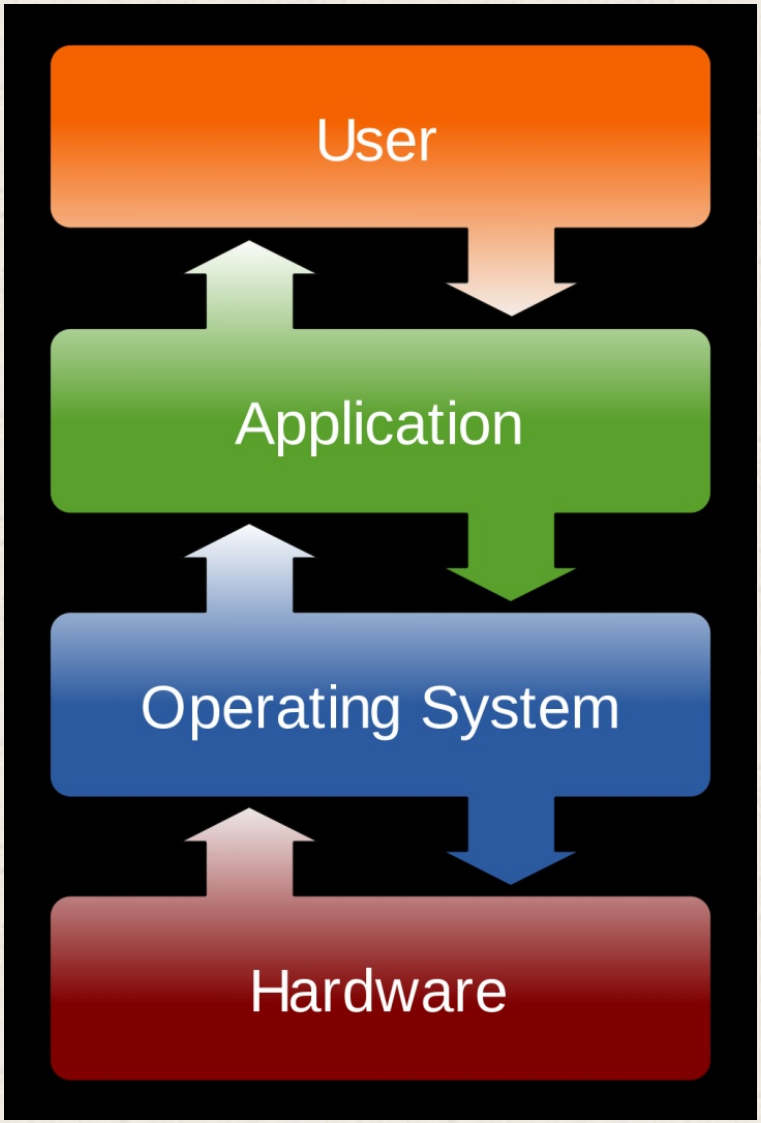
Operating Systems

An **operating system** is **software that manages and interacts with the hardware resources** of a computer. It acts as the "middle man" between the hardware and executing application programs. For example, it controls the allocation of memory for the various programs that may be executing on a computer.

Because an operating system is **intrinsic to the operation of a computer**, it is referred to as **system software**.

Operating systems also provide a particular user interface.

It is the operating system installed on a computer that determines the "look and feel" of the user interface and how the user interacts with the system, and not the particular model computer.


User
Application
Operating System
Hardware

Limits of Integrated Circuits Technology: Moore's Law

In 1965, **Gordon E. Moore**, one of the pioneers in the development of integrated circuits and cofounder of Intel Corporation, **predicted that the number of transistors that would be able to be put on a silicon chip would double roughly every two years**, allowing the complexity and therefore the capabilities of integrated circuits to grow exponentially. **This prediction became known as Moore's Law**.

Amazingly, to this day that prediction has held true. While this doubling of performance cannot go on indefinitely, it has not yet reached its limit.

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS:

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS 1. True,

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True 2. transistors,

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True, 2. transistors 3. (c),

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True, 2. transistors, 3. (c), 4. binary digit, 5. False,

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True, 2. transistors, 3. (c), 4. binary digit, 5. False, 6. (c),

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True, 2. transistors, 3. (c), 4. binary digit, 5. False, 6. (c) 7. CPU,

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True, 2. transistors, 3. (c), 4. binary digit, 5. False, 6. (c), 7. CPU, 8. True,

Self-Test Questions

1. All information in a computer system is in binary representation. (TRUE/FALSE)
2. Computer hardware is based on the use of electronic switches called _____.
3. How many of these electronic switches can be placed on a single integrated circuit, or "chip"?
   (a) Thousands
   (b) Millions
   (c) Billions
4. The term "bit" stands for _____.
5. A bit is generally a group of eight bytes. (TRUE/FALSE)
6. What is the value of the binary representation 0110.
   (a) 12
   (b) 3
   (c) 6
7. The _____ interprets and executes instructions in a computer system.
8. An operating system manages the hardware resources of a computer system, as well as provides a particular user interface. (TRUE/FALSE)
9. Moore's Law predicts that the number of transistors that can fit on a chip doubles about every ten years. (TRUE/FALSE)

ANSWERS: 1. True, 2. transistors, 3. (c), 4. binary digit, 5. False, 6. (c), 7. CPU, 8. True, 9. False

Computer Software

What is computer software?

Computer software is a set of program instructions, including related data and documentation, that can be executed by computer. This can be in the form of instructions on paper, or in digital form.

While **system software** is intrinsic to a computer system, **application software** fulfills users' needs, such as a photo-editing program.

The First Computer Programmer

The first computer programs ever written were for a mechanical computer designed by **Charles Babbage** in the mid-1800s. The person who wrote these programs was a woman, **Ada Lovelace**, who was a talented mathematician. Thus, she is referred to as "**the first computer programmer**." (For more on this history, see Chapter 12).

Syntax, Semantics and Program Translation

Programming languages (called **"artificial languages"**) are languages just as **"natural languages"** such as English and Mandarin (Chinese). *Syntax* **and** *semantics* **are important concepts that apply to all languages.**

Syntax

The **syntax** of a language is a **set of characters and the acceptable sequences (arrangements)** of those characters.

English, for example, includes the letters of the alphabet, punctuation, and properly spelled words and properly punctuated sentences. The **following is a syntactically correct sentence** in English,

"Hello there, how are you?"

The **following, however, is not syntactically correct**,

"Hello there, hao are you?"

The sequence of letters "hao" is not a word in the English language.

Semantics

The **semantics** of a language is the **meaning associated with each syntactically correct sequence of characters**.

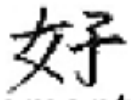
Consider the following sentence:

"Colorless green ideas sleep furiously."

This sentence is syntactically correct, but has no meaning. Thus, it is *semantically incorrect.*

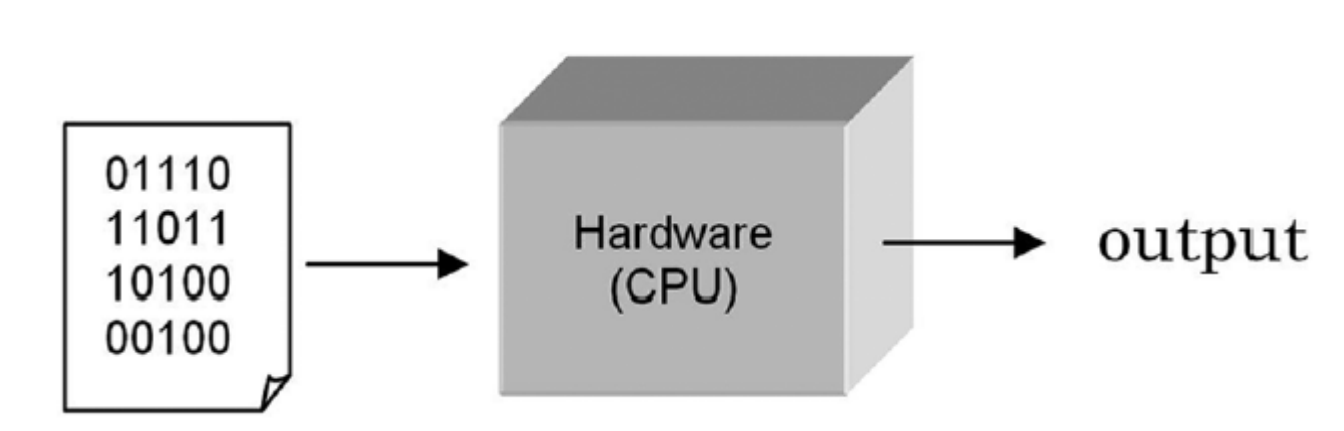
Every language has its own syntax and semantics.

For example, in English "Hao" is syntactically incorrect. In Mandarin (Chinese), however, "Hao" is a valid word meaning "good." ("Hao" is from a system called pinyin, which uses the Roman alphabet rather than Chinese characters for writing Mandarin.)

ENGLISH	MANDARIN (pinyin)	MANDARIN (Chinese Characters)
Syntax Hao	**Syntax** Hao	**Syntax** 
Semantics No meaning *(syntactically incorrect)*	**Semantics** *"Good"*	**Semantics** *"Good"*

Program Translation

A central processing unit (CPU) is **designed to interpret and execute a specific set of instructions represented in binary form** (i.e., 1s and 0s) called **machine code**. Only programs in machine code can be executed by a CPU.

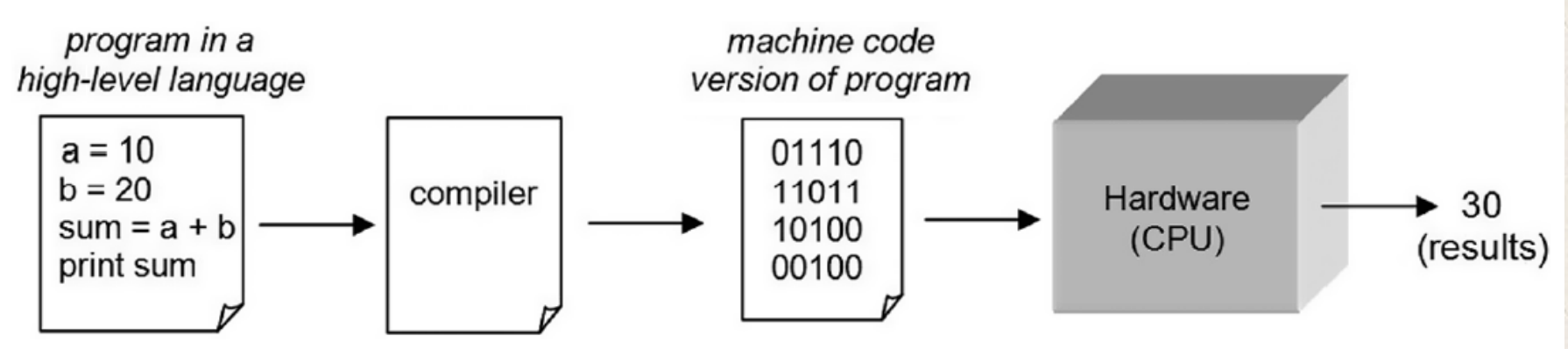


Writing programs at this "low level" is tedious and error-prone. Therefore, most programs are written in a "high-level" programming language such as Python. Since the instructions of such programs are not in machine code that a CPU can execute, a translator program must be used.

There are two fundamental types of translators:

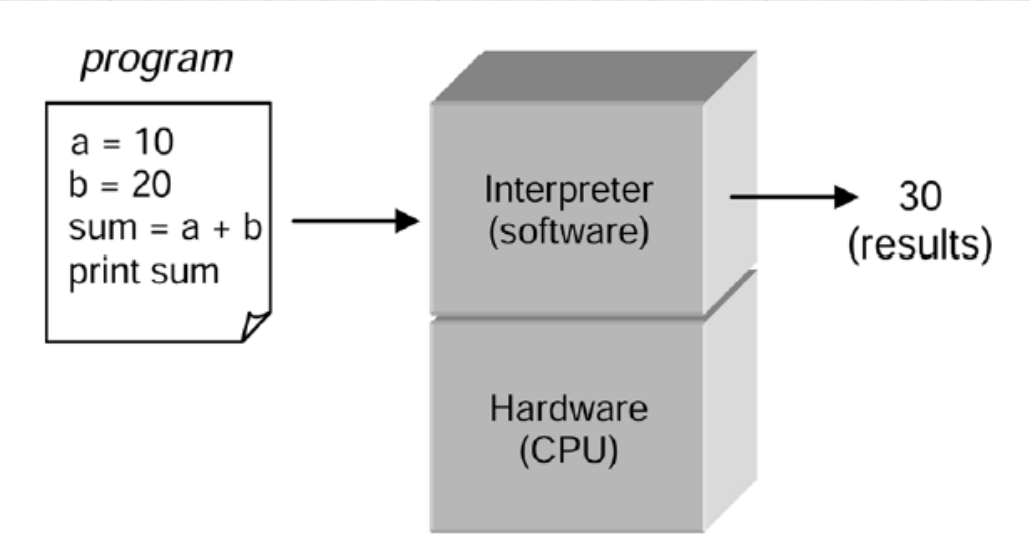
- **Compiler** software that translates programs into machine code to be executed by the CPU
- **Interpreter** software that executes programs in place of the CPU

Compiler



Compiled programs generally execute faster than interpreted programs.

Interpreter



An interpreter can immediately execute instructions as they are entered. This is referred to as **interactive mode**. This is a very useful feature for program development. **Python, as we shall see, is executed by an interpreter**.

Program Debugging: Syntax Errors vs. Semantic Errors

Program debugging is the process of finding and correcting errors (**"bugs"**) in a computer program. Programming errors are inevitable during program development.

Syntax errors are caused by invalid syntax (for example, entering `prnt` instead of `print`).

Since **a translator cannot understand instructions containing syntax errors**, translators terminate when encountering such errors indicating where in the program the problem occurred.

In contrast, **semantic errors** (generally called logic errors) **are errors in program logic**. Such errors cannot be automatically detected, since translators cannot understand the intent of a given computation. Therefore, programs are not terminated when containing logic errors, but give incorrect results.

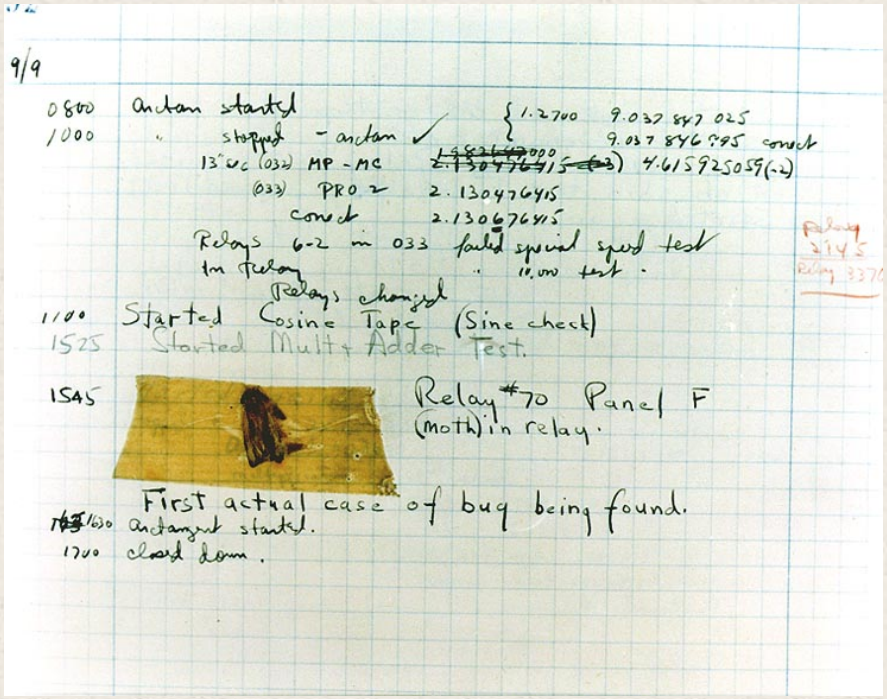
For example, if a program computed the average of three numbers as follows,

```
(num1 + num2 + num3) / 2.0
```

a translator would have no means of determining that the divisor should be 3 and not 2. **Computers do not understand what a program is meant to do, they only follow the instructions given**. *It is up to the programmer* to detect such errors.

Program debugging is not a trivial task, and constitutes much of the time of program development.

The First Actual Computer "Bug"



In 1947, engineers working on the **Mark II** computer **at Harvard University** found a moth stuck in one of the components, causing the machine to malfunction. They taped the insect in their logbook and labeled it "**first actual case of bug being found**."

The term program "bug" **(and** "debugging"**) has become a standard part of the language of computer programmers**. The log book, complete with the attached bug, is on display at the Smithsonian Institution in Washington, D.C.

Procedural vs. Object-Oriented Programming

Programming languages fall into a number of programming paradigms. **The two major programming paradigms in use today are procedural (imperative) programming and object-oriented programming**. Each provides a different way of thinking about computation.

While most programming languages only support one paradigm, **Python supports both procedural and object-oriented programming**. We will start with the procedural aspects of Python.

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS:

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application.

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application, 2. (c),

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application, 2. (c) 3. False,

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application, 2. (c), 3. False, 4. machine code.

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application, 2. (c), 3. False, 4. machine code, 5. compilers, interpreters,

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application, 2. (c), 3. False, 4. machine code, 5. compilers, interpreters, 6. program debugging,

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

ANSWERS: 1. application, 2. (c), 3. False, 4. machine code, 5. compilers, interpreters, 6. program debugging, 7. (a),

Self-Test Questions

1. Two general types of software are system software and _____ software.
2. The syntax of a given language is,
   (a) the set of symbols in the language.
   (b) the acceptable arrangement of symbols.
   (c) both of the above
3. The semantics of a given language is the meaning associated with any arrangement of symbols in the language. (TRUE/FALSE)
4. CPUs can only execute instructions that are in binary form called _____.
5. The two fundamental types of translation programs for the execution of computer programs are _____ and _____.
6. The process of finding and correcting errors in a computer program is called _____.
7. Which kinds of errors can a translator program detect?
   (a) Syntax errors
   (b) Semantic errors
   (c) Neither of the above
8. Two major programming paradigms in use today are _____ programming and _____ programming.

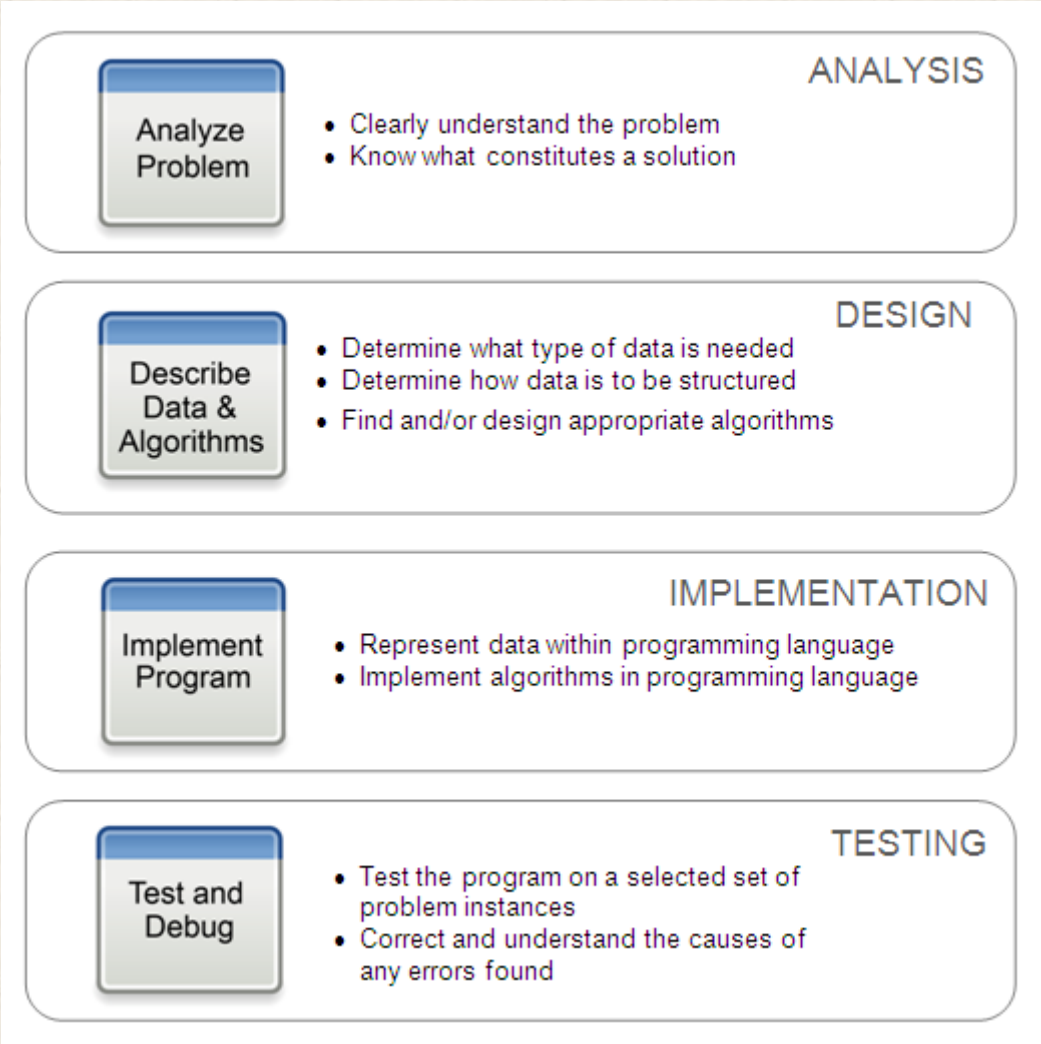
ANSWERS: 1. application, 2. (c), 3. False, 4. machine code, 5. compilers, interpreters, 6. program debugging, 7. (a) 8. procedural, object-oriented

The Process of Computational Problem Solving

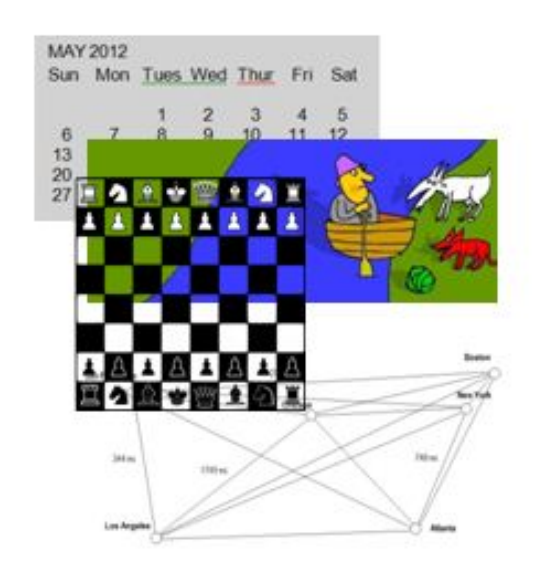
Computational problem solving does not simply involve the act of computer programming. It is a *process*, with programming only one of the steps.

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



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 either,

- **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, since going from Atlanta to Boston or Boston to Atlanta, the distance is the same

Representation for Chess Playing Program

Below are two possible ways to represent the pieces on a chess board.

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. Those are decisions 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.

The Python Programming Language

The Python Programming Language was created by **Guido van Rossum**. It was first released in the early 1990s.

Its name comes from a 1970s British comedy sketch show *called Monty Python's Flying Circus.* (An example of their work is The Argument Clinic).

Companies and organizations that use Python include YouTube, Google, Yahoo and NASA.

Python Features

- **Simple Syntax**
  Python programs are clear and easy to read

- **Interpreted Language**
  Python instructions can be executed interactively

- **Powerful Programming Features**
  Can accomplish significant computation with few instructions

- **Numerous Python Modules Provide Additional Capabilities**

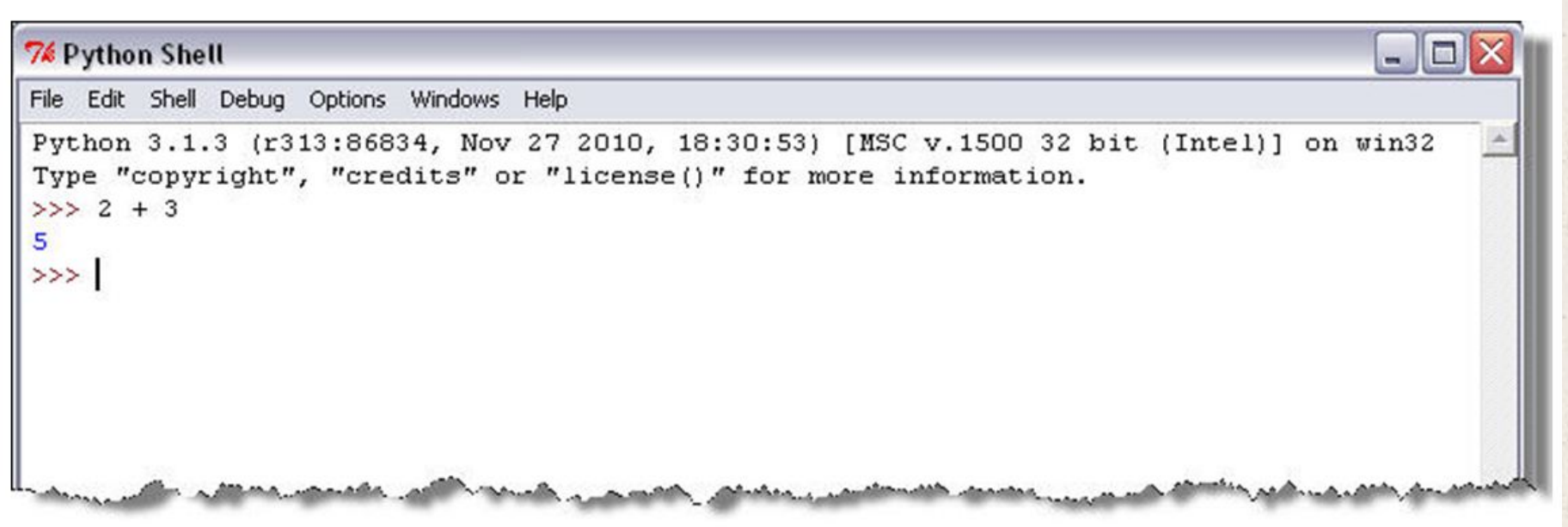
The IDLE Python Development Environment

IDLE is an **integrated development environment (IDE)**. An **IDE** is a bundled set of software tools for program development. This typically includes,

- **an editor**
  for creating and modifying programs
- **a translator**
  for executing programs, and
- **a program debugger**
  for taking control of the execution of a program to aid in finding program errors

The Python Shell

Python can be executed interactively in the **Python shell**. **In this mode, executing Python is similar to using a calculator**.



The **>>> symbol is the shell prompt**. Here, typing 2 + 3 at the prompt outputs the result 5, redisplaying the prompt in wait of another instruction.

The Python Standard Library

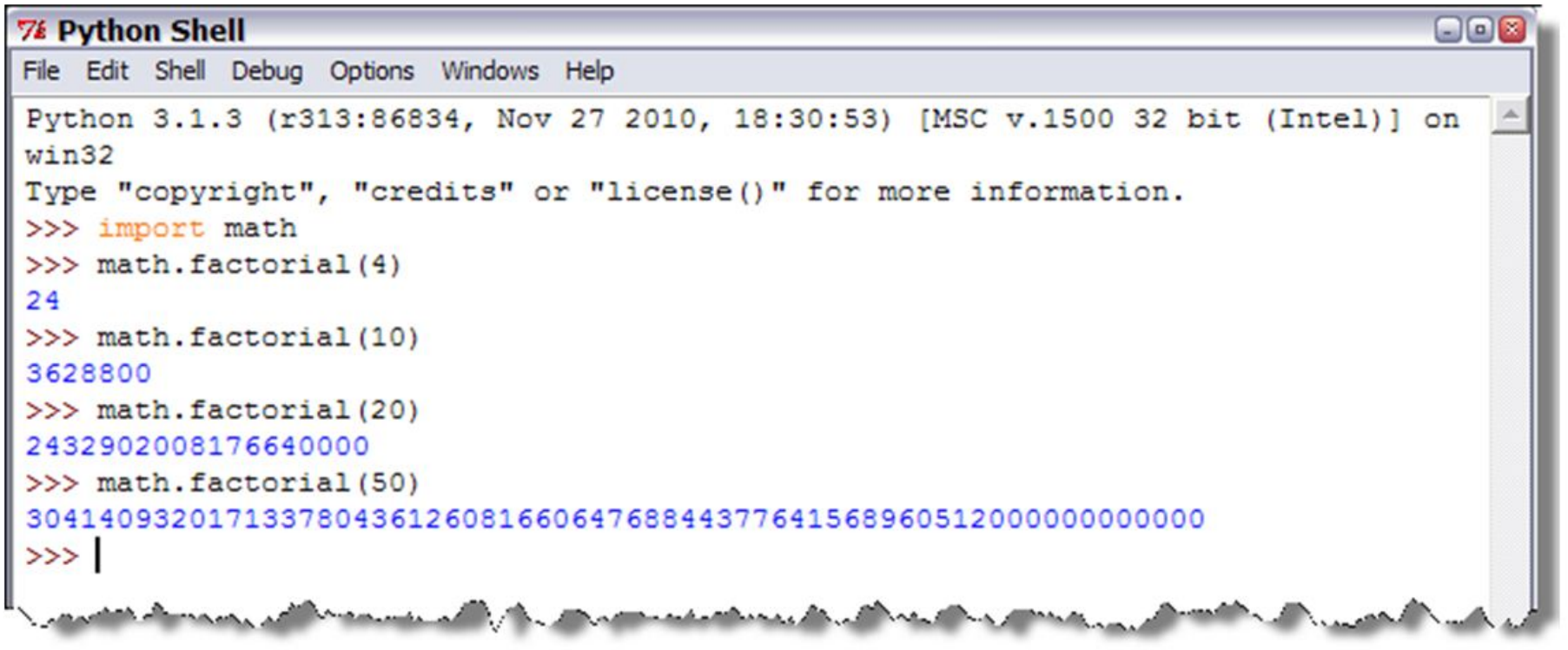
The **Python Standard Library** is a collection of **built-in modules**, *each providing specific functionality* beyond what is included in the "core" part of Python.

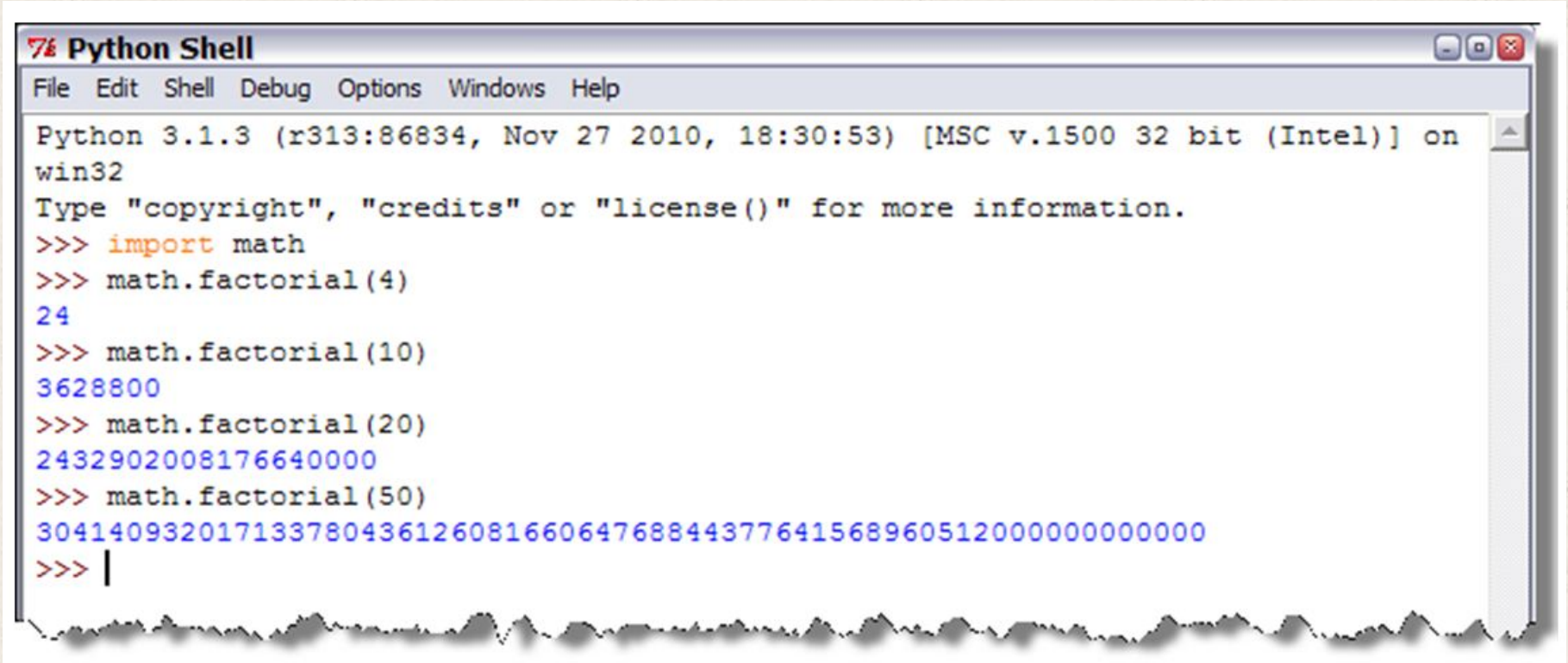
For example, the math module provides additional mathematical functions. The random module provides the ability to generate random numbers, useful in programming, as we shall see.

Other Python modules are described in the Python 3 Programmers' Reference at the end of the book.

Importing a Library Module

In order to utilize the capabilities of modules in a specific program, an **import** statement is used as shown.





Because the factorial function is from the math module, the function is called with the name of the module prepended:

e.g., **`math.factorial(20)`**

A Bit of Python

We introduce a bit of Python, just enough to begin writing some simple programs.

Since all computer programs,

- input data
- process the data
- output results

We look at the notion of a variable, how to perform some simple arithmetic calculations, and how to do simple input and output.

Variables

One of the most fundamental concepts in programming is that of a *variable*.

A **variable** is "a name that is assigned to a value," as shown below,

```
n = 5
```

variable `n` is assigned the value 5

Thus, whenever variable `n` appears in a calculation, the **current value of** `n` is used, as in the following,

```
n + 20
```

(5 + 20) → 25

If variable `n` is assigned a new value, then the **same expression will produce a different result**,

```
n = 10
n + 20
```

(10 + 20) → 30

Some Basic Arithmetic Operators

The **common arithmetic operators** in Python are,

+ (addition)	* (multiplication)	** (exponentiation)
– (subtraction)	/ (division)	

Addition, subtraction, and division use standard mathematical notation,

```
10 + 20          25 - 15          20 / 10
```

(Also, `//` for truncated division)

For multiplication and exponentiation, the asterisk (*) is used,

`5 * 10` (5 times 10) `2 ** 4` (2 to the 4th power)

Multiplication is never denoted by the use of parentheses,

`10 * (20 + 5)` **CORRECT** `10(20 + 5)` **INCORRECT**

Note that parentheses may be used to denote subexpressions.

Basic Input

The programs that we will write request and get information from the user. In Python, the **input** function is used for this purpose,

```
name = input('What is your name?: ')
```

Characters within quotes are called strings. This particular use of a string, for requesting input from the user, is called a **prompt**.

The input function displays the string on the screen to prompt the user for input,

```
What is your name?: Charles
```

(The underline is used here to indicate the user's input.)

Basic Output

The print function is used to display information on the screen in Python. This may be used to display a message (string),

```
>>> print('Welcome to My First Program!')
Welcome to My First Program!
```

or used to output the value of a variable,

```
>>> n = 10
>>> print(n)
10
```

Can also display a combination of strings and variables,

```
>>> name = input('What is your name?: ')
What is your name?: Charles

>>> print('Hello', name)
Hello Charles
```

Note that a comma is used to separate the individual items being printed, which causes a space to appear between each when displayed. Thus, the output of the print function in this case is

```
Hello Charles
```

and not

```
HelloCharles
```

We will soon learn more about variables, operators, and input/output in Python.

Using IDLE

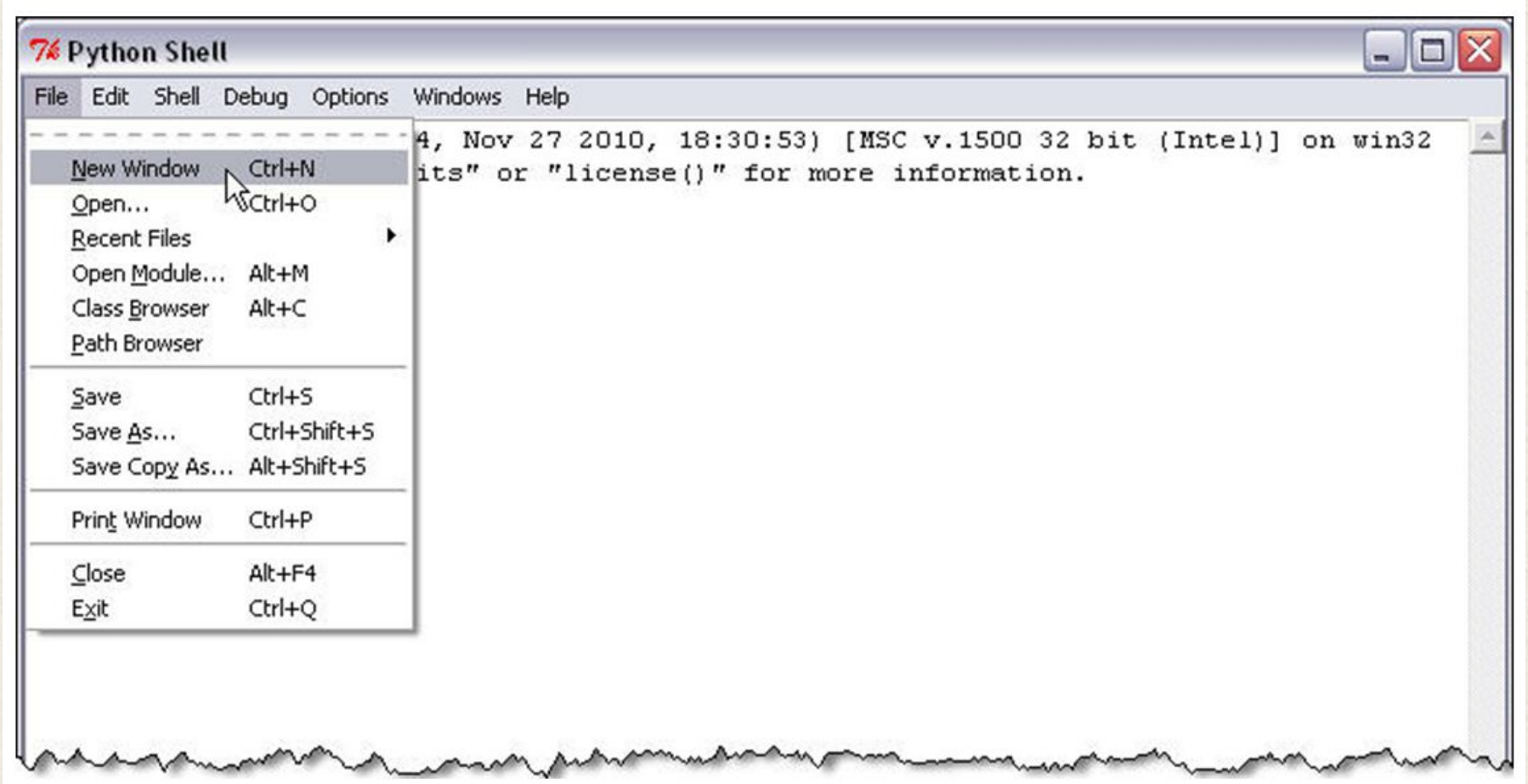
In order to become familiar with writing your own Python programs using IDLE, we will create a simple program that asks the user for their name, and then responds with a greeting.

This program utilizes the following concepts:

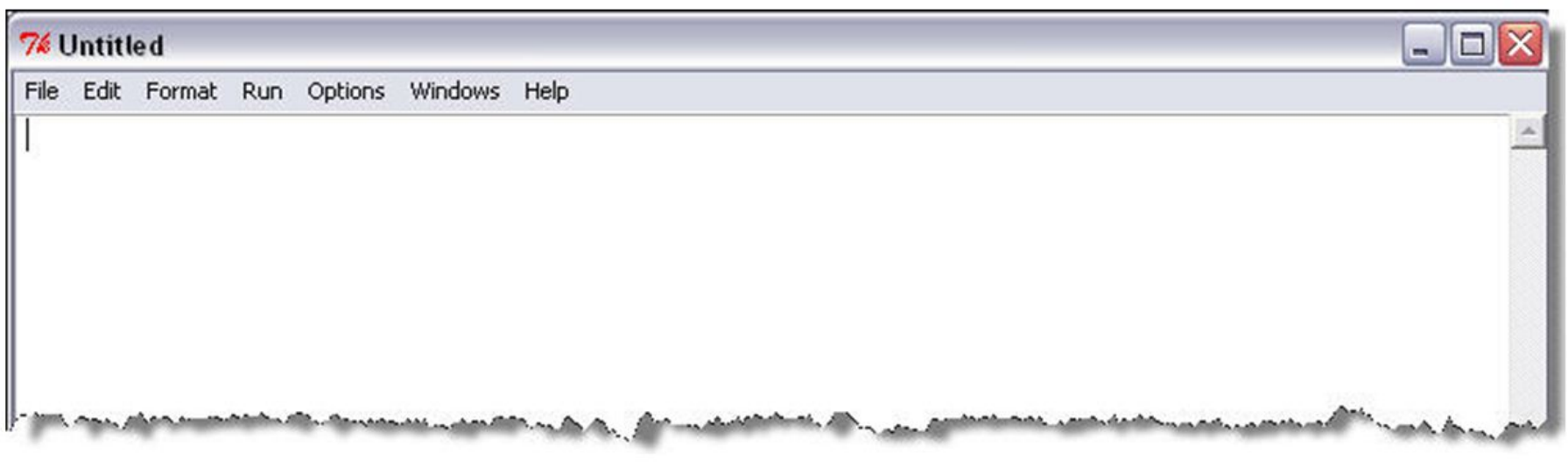
- **Creating and executing Python programs**
- **Input and print functions**

Creating a New Python Program

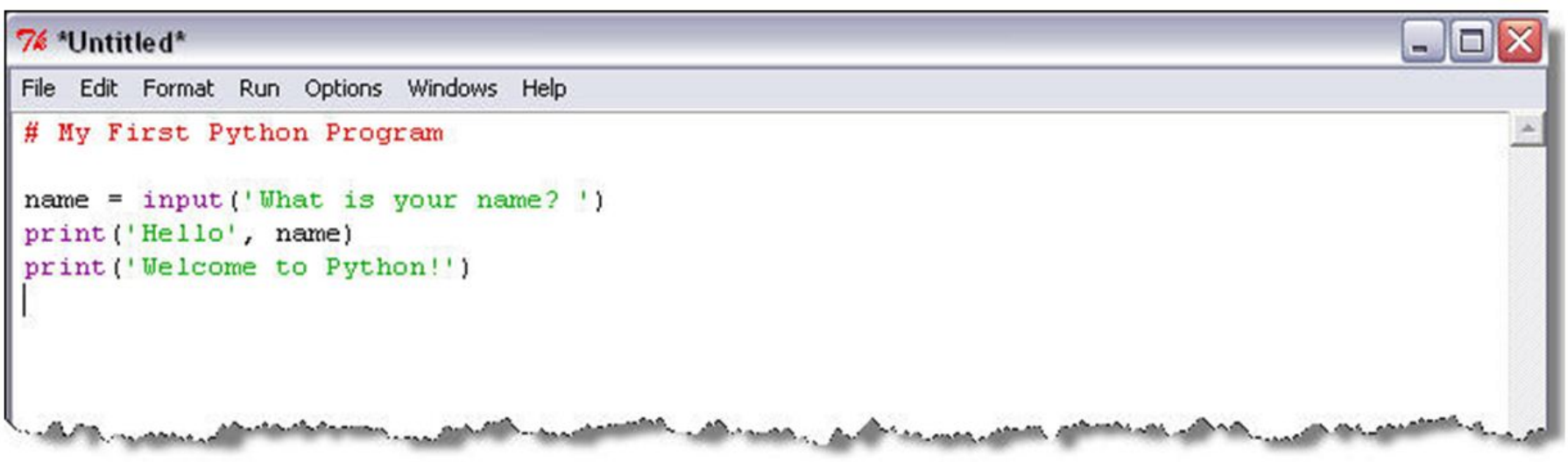
To create a Python program file, **select** New Window **from the File menu** in the Python shell,



A **new, untitled window** will appear,

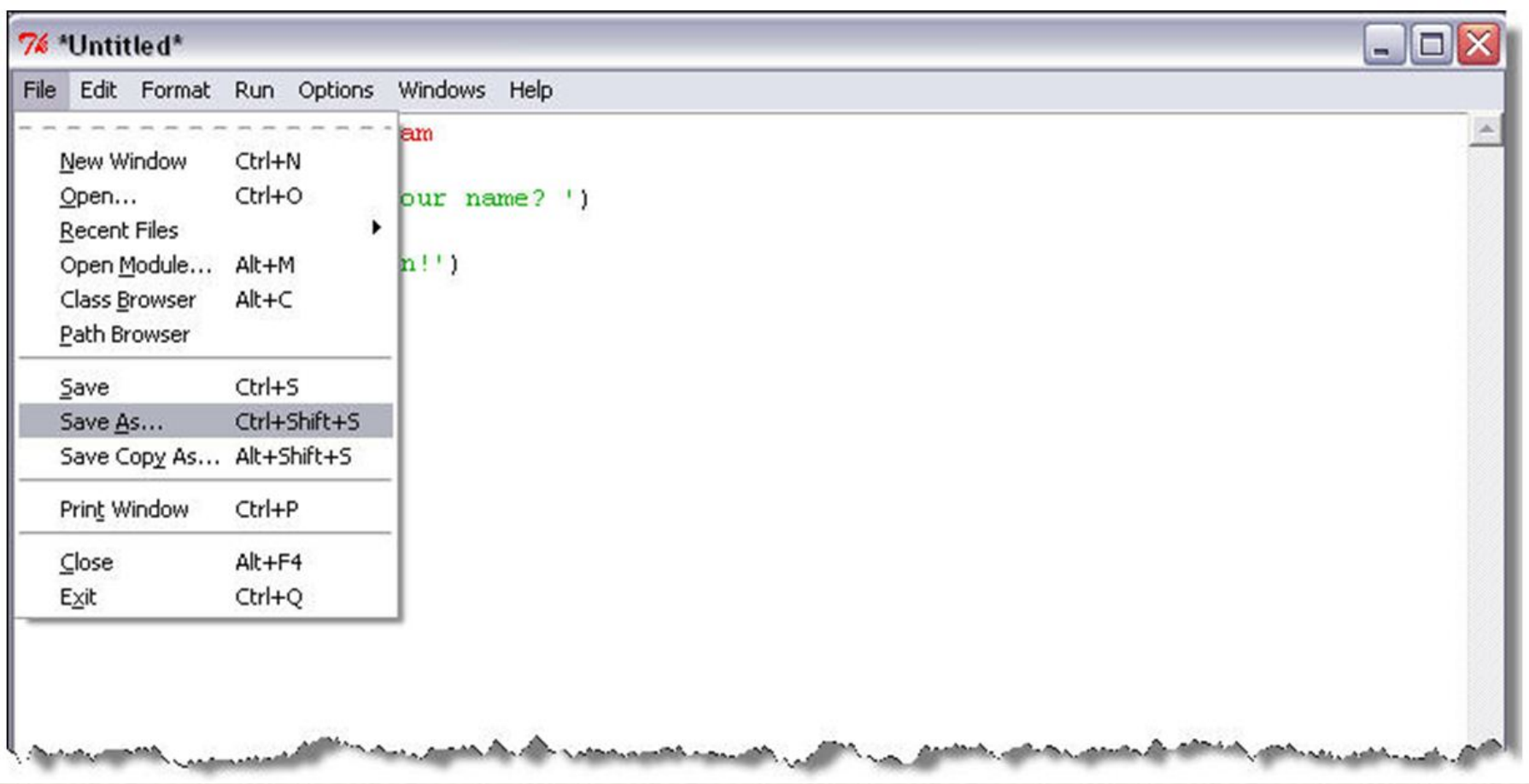


Now can **begin entering lines** of a program without them being immediately executed, as in the Python shell.



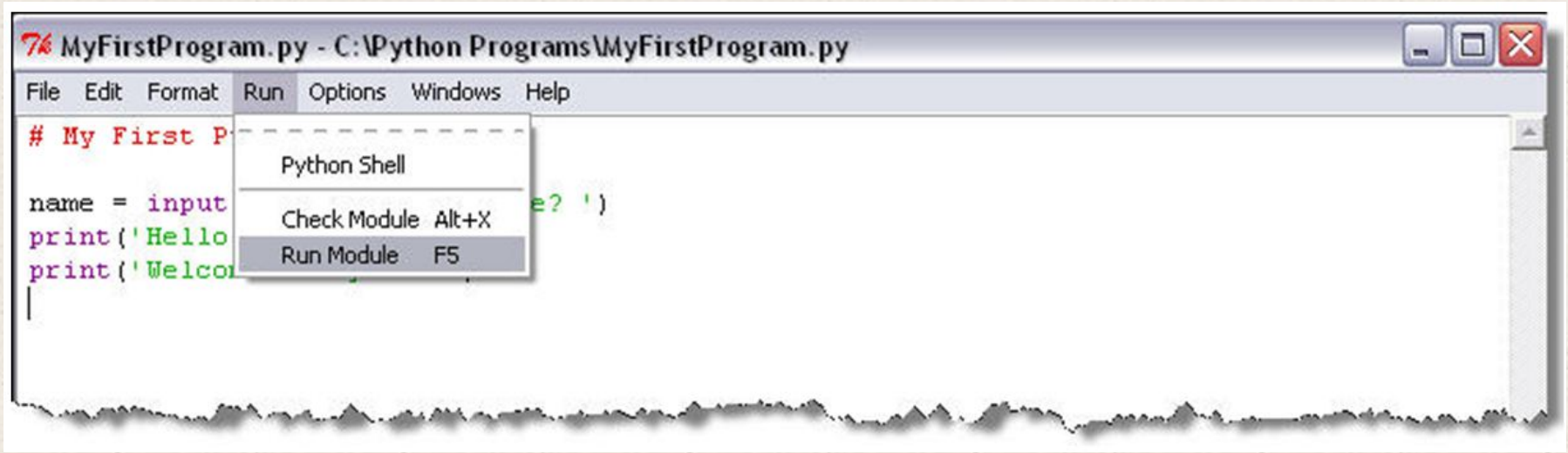
Note that parts of the program lines are displayed in a certain color. Since `print` and `input` are predefined function names in Python, they are colored purple. The strings in the program are colored green. The statement in red is a comment statement. **Comment statements are for those reading the program, and are ignored when the program is executed**.

When finished, **save the program file** by selecting Save As (under the File menu) and save in the appropriate folder with the name MyFirstProgram.py.

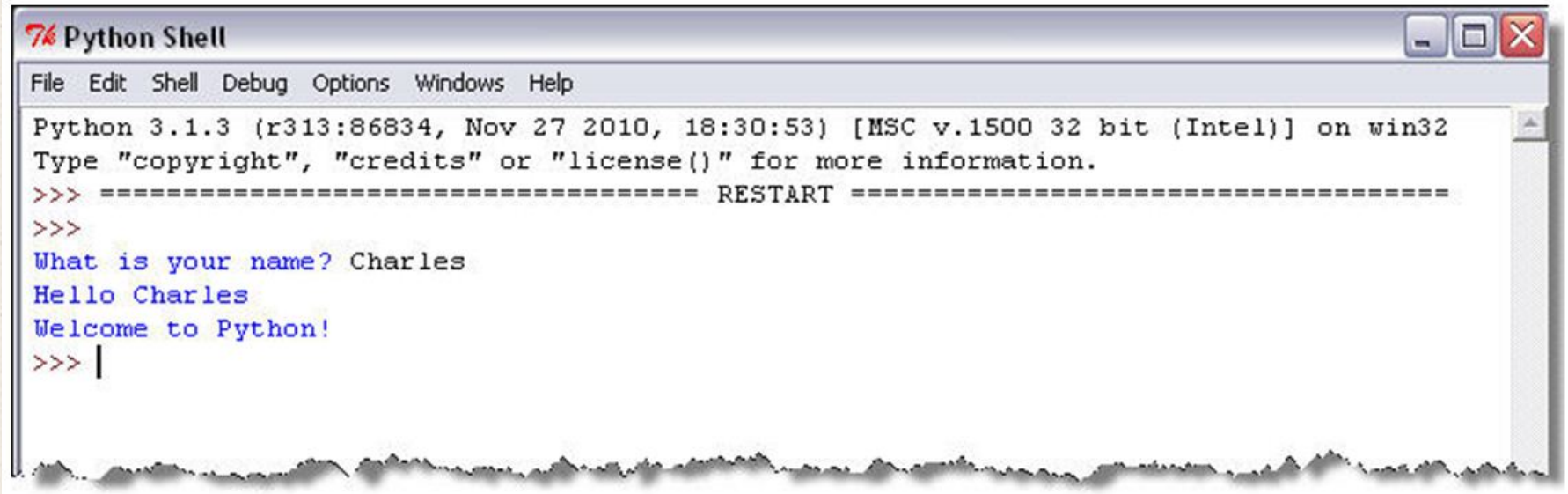


Executing a Python Program

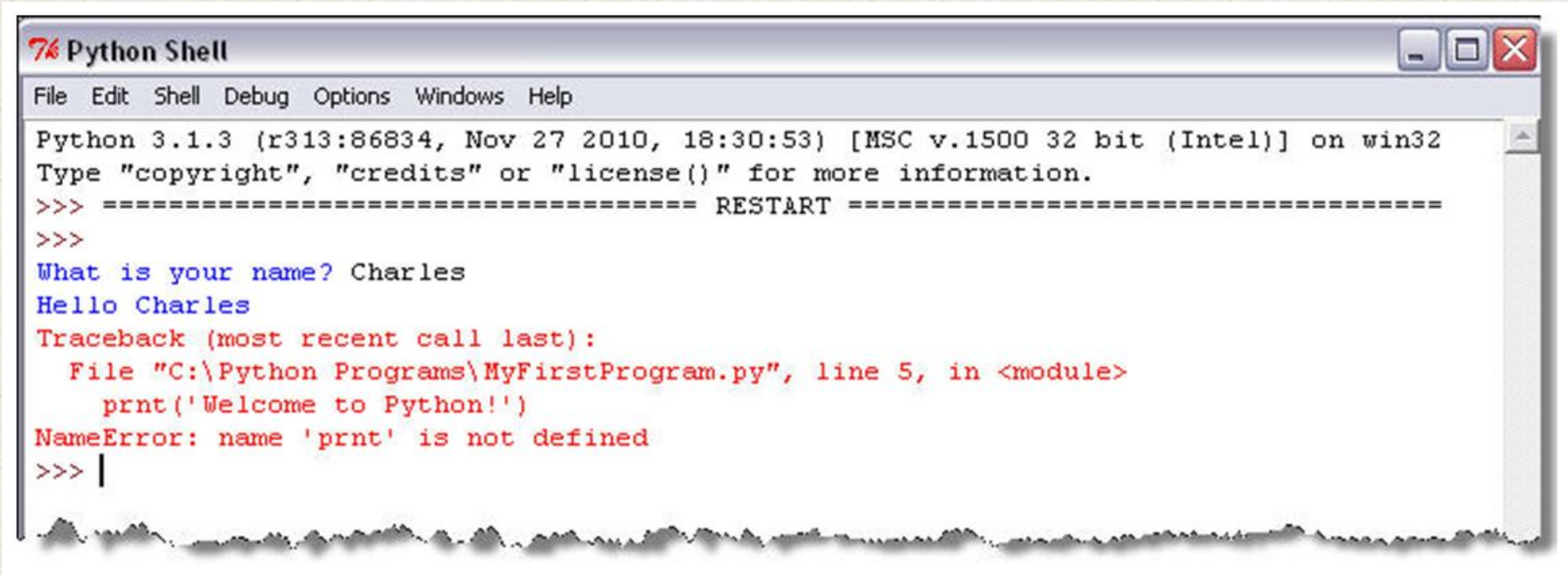
To **run a Python program**, select Run Module from the Run menu (or simply hit function key F5).



If you have entered the program code correctly, the program should execute as shown



However, if you have mistyped part of the program resulting in a syntax error (such as mistyping `print`), you will get an error message.



In such instances, you need to go back to the program window and make the needed corrections, the re-save and re-execute the program. **You may need to go through this process a number of times until all the syntax errors have been corrected**.

A First Program
Calculating the Drake Equation

Dr. Frank Drake conducted the first search for radio signals from extraterrestrial civilizations in 1960. This established SETI (Search for Extraterrestrial Intelligence), a new area of scientific inquiry.

In order to estimate the number of civilizations that may exist in our galaxy that we may be able to communicate with, he developed what is now called the *Drake equation*.

The Drake equation accounts for a number of different factors. Some of the values are the result of scientific study, while others are only the result of an "intelligent guess." The factors consist of,

R, the average rate of star creation per year in our galaxy
p, the percentage of those stars that have planets
n, the average number of planets that can potentially support life for each star with planets
f, the percentage of those planets that actually go on to develop life
I, the percentage of those planets that go on to develop intelligent life
c, the percentage of those that have the technology communicate with us and
L, the expected lifetime of civilizations (the period that they can communicate).

The Drake equation is simply the multiplication of all these factors, giving **N**, the estimated number of detectable civilizations there are at any given time,

$$N = R * p * n * f * i * c * L$$

The following table shows those parameters in the Drake equation that have some consensus as to their correct value, as well as the values that Drake himself has used.

Drake Equation Factor Values		Drake's Values	Current Values
Rate of star creation	R	10	7†
Percentage of stars with planets	p	50%	40%
Average number of planets that can potentially support life for each star with planets	n	2	(no consensus)
Percentage of those that go on to develop life	f	100%	13%
Percentage of those that go on to intelligent develop life	i	1%	(no consensus)
Percentage of those willing and able to communicate	c	1%	(no consensus)
Expected lifetime of civilizations	L	10,000	(no consensus)

† Estimate of NASA and the European Space Agency

Calculating the Drake Equation

The Problem

The value of 7 for R, the rate of star creation, is the least disputed value in the Drake equation today.

Given the uncertainty of the remaining factors, develop a program that allows a user to enter their own estimated values for the remaining six factors (p, n, f, i, c, and L) and displays the calculated result.

Calculating the Drake Equation

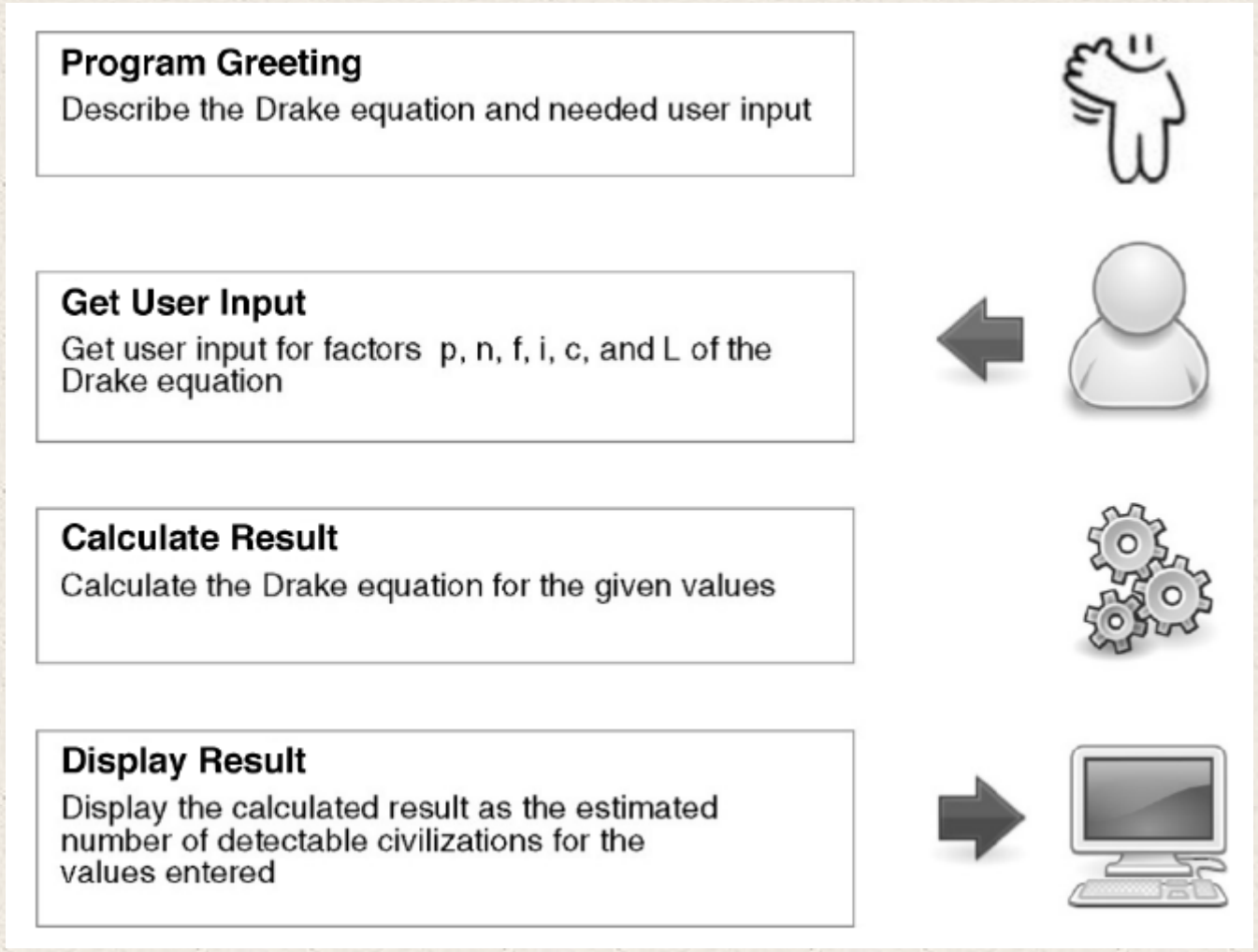
Problem Analysis

The problem is very straightforward. We only need to understand the equation provided.

Calculating the Drake Equation

Program Design

The program design for this problem is straightforward. The data to be represented consist of numerical values, with the Drake equation as the algorithm.



The Overall Steps of the Program

Calculating the Drake Equation

Program Implementation

The implementation of this program is fairly simple. The only programming elements needed are input, assignment, and print, along with the use of arithmetic operators.

```
# SETI Program
#
# The Drake equation, developed by Frank Drake in the 1960s, attempts to
# estimate how many extraterrestrial civilizations, N, may exist in our
# galaxy at any given time that we might come in contact with,
#
#    N = R * p * n * f * i * c * L
#
# where,
#
#    R ... estimated rate of star creation in our galaxy
#    p ... estimated percent of stars that have planets
#    n ... estimated average number of planets that can potentially support
#          life for each star with planets
#    f ... estimated percent of those planets that actually go on to develop life
#    i ... estimated percent of those planets go on to develop intelligent life
#    c ... estimated percent of those that are willing and able to communicate
#    L ... estimated expected lifetime of such civilizations
#
# Given that the value for R, 7 per year, is the least disputed of the values,
# the user will be prompted to enter estimated values for the remaining six
# factors. The estimated number of civilizations that may be detected in our
# galaxy will then be displayed.

# display program welcome
print('Welcome to the SETI program')
print('This program will allow you to enter specific values related to')
print('the likelihood of finding intelligent life in our galaxy. All')
print('percentages should be entered as integer values, e.g., 40 and not .40')
print()

# get user input
p = int(input('What percentage of stars do you think have planets?: '))
n = int(input('How many planets per star do you think can support life?: '))
f = int(input('What percentage do you think actually develop life?: '))
i = int(input('What percentage of those do you think have intelligent life?: '))
c = int(input('What percentage of those do you think can communicate with us?: '))
L = int(input('Number of years you think civilizations last?: '))

# calculate result
num_detectable_civilizations = 7 * (p/100) * n * (f/100) * (i/100) * (c/100) * L

# display result
print()
print('Based on the values entered ...')
print('there are an estimated', round(num_detectable_civilizations),
      'potentially detectable civilizations in our galaxy')
```

```
24
25  # display program welcome
26  print('Welcome to the SETI program')
27  print('This program will allow you to enter specific values related to')
28  print('the likelihood of finding intelligent life in our galaxy. All')
29  print('percentages should be entered as integer values, e.g., 40 and not .40')
30  print()
31
32  # get user input
33  p = int(input('What percentage of stars do you think have planets?: '))
34  n = int(input('How many planets per star do you think can support life?: '))
35  f = int(input('What percentage do you think actually develop life?: '))
36  i = int(input('What percentage of those do you think have intelligent life?: '))
37  c = int(input('What percentage of those do you think can communicate with us?: '))
38  L = int(input('Number of years you think civilizations last?: '))
39
40  # calculate result
41  num_detectable_civilizations = 7 * (p/100) * n * (f/100) * (i/100) * (c/100) * L
42
43  # display result
44  print()
45  print('Based on the values entered ...')
46  print('there are an estimated', round(num_detectable_civilizations),
47        'potentially detectable civilizations in our galaxy')
```

Calculating the Drake Equation

Python Issues to Point Out

- **Comment statements** start with a hash (#) sign
- Comments at the start of the program give an overall program description
- Comments within code serve as section headers

- **Print function** is used for program welcome (lines 26-29)
- Empty print function calls cause a skipped line on the screen (line 44)
- Print function also used to display results (lines 45-47)

- **Input function** is used for getting factors from user for Drake Equation
- The input function always returns a string value
- `int(input('.....'))` used to convert input string to an integer value

Calculating the Drake Equation

Example Execution

Program Execution ...

```
Welcome to the SETI program
This program will allow you to enter specific values related to
the likelihood of finding intelligent life in our galaxy. All
percentages should be entered as integer values, e.g., 40 and not .40

What percentage of stars do you think have planets?: 40
How many planets per star do you think can support life?: 2
What percentage do you think actually develop life?: 5
What percentage of those do you think have intelligent life?: 3
What percentage of those do you think can communicate with us?: 5
Number of years you think civilizations last?: 10000

Based on the values entered...
there are an estimated 4.2 potentially detectable civilizations in
our galaxy
>>>
```


Calculating the Drake Equation

Program Testing

To test the program, we can calculate the Drake equation for various other values using a calculator, providing a set of ***test cases***.

A **test case** is a **set of input values and expected output** of a given program.

A **test plan** consists of **a number of test cases** to verify that a program meets all requirements.

A good strategy is to include "average," as well as "extreme" or "special" cases in a test plan.

Test Plan (with results)

Input Values	p	n	f	i	c	L	Expected Results	Actual Results	Evaluation
Extreme Cases (no chance of contacting intelligent life)									
Zero Planets per Star	0	2	100%	1%	1%	10,000	0	0	passed
Zero percent of Planets Support Life	50%	0	100%	1%	1%	10,000	0	0	passed
Average Cases									
Average Case 1	30%	3	75%	1%	5%	5,000	12	12	passed
Average Case 2	50%	6	80%	5%	10%	10,000	840	840	passed
Extreme Cases (great chance of contacting intelligent life)									
Extreme Case 1	100%	10	100%	100%	100%	10,000	700,000	700,000	passed
Extreme Case 2	100%	12	100%	100%	100%	100,000	8,400,000	8,400,000	passed