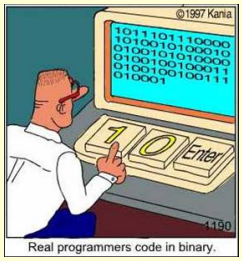
What is programming?

The act of writing a program.

What is a program?

An algorithm written in a language that can be understood by computers.

Real programming?



Real programmers code in binary.

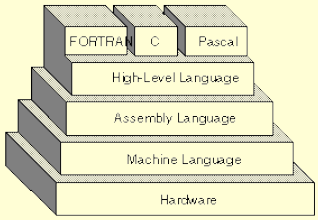
http://www.cengizhan.com/

High-level languages

In practice, programs are written in *high-level languages* that can be understood by both human and computers.

Such programs are then translated to binary codes (in a machine language).

Hierarchy of languages



http://www.webopedia.com/

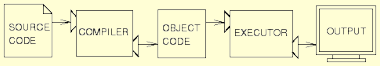
High-level languages

In practice, programs are written in *high-level languages* that can be understood by both human and computers.

Such programs are then translated to binary codes (in a machine language).

A compiler is a software system to perform such translation.

Compiler



- Translates a program into binary/object code completely before execution.
- Program execution is done by running binary/object code.
- The first compiler was invented by

Grace Murray Hopper (1906–1992)

http://www.wikipedia.org/

High-level languages

In practice, programs are written in *high-level languages* that can be understood by both human and computers.

Such programs are then translated to binary codes (in a machine language).

An *interpreter* provides another way to perform such translation.

Interpreter

- Translates and executes a program one line at a time.
- Unlike compilers, programs must be translated every time they are executed.
- Inefficient in general, but convenient!

Modern high-level languages

- Compiled: C/C++, Fortran, Java, Pascal, …
- Interpreted: JavaScript, Perl, Python, Ruby, Scheme, …

Python and Java

In this course, you will be learning how to program in two languages.

- Python: A simple language that supports a user-friendly interpreter.
- Java: The industry standard. A bit harder to learn than Python but very powerful for developing large-scale software, involving many programmers.

Python

```
Print 'Hello, World!'
```


Java

```
class HelloWorld {
  static public void main(String args[]) {
    System.out.print("Hello, World!");
  }
}
```


C++

```
#include <iostream.h>

main() {
  cout << "Hello, World!";
  return 0;
}
```

Scheme

```
(define helloworld
  (lambda ()
    (display ”Hello, World!”)))
```


Software development process

Software development process

In a nut shell, this consists of the following three basic steps:

- Problem analysis
- Algorithm design
- Implementation

Problem analysis

Analyze and formalize the problem to be solved. Determine the *Input/Output (I/O) specification* (which in turn determines what the program has to accomplish).

Algorithm design

Formulate the overall structure of the program. Design algorithms that meet the specification. Algorithms are written informally in pseudocode.

Implementation

Translate algorithms into program codes. When completed, test and debug the program.

Software development process

In a nut shell, this consists of the following three basic steps:

- Problem analysis
- Algorithm design
- Implementation

Most people skip analysis and design and jump into implementation — a very bad idea!

Python's basic constructs

Values

Numbers, letters, words, … are collectively called *values*.

Example. The following are all values.

```
1823
3.141592653
'Trinity College'
```

Types

Values are categorized into different *types*: integers, floating-point (real) numbers, strings, …

Each value belongs to a single type (i.e., no value belongs to more than one type).

Types

Python has a type command that tells you the type of a given value:

Example.

```
>>> type(1823)
<type 'int'>
>>> type(3.141592653)
<type 'float'>
>>> type('Trinity College')
<type 'str'>
```


Variables

A *variable* is a name that refers to a value.

An *assignment* statement creates a variable and assigns a value to it.

Example.

```
>>> n = 1823
>>> print n
1823
```

Variables

A *variable* is a name that refers to a value.

An *assignment* statement creates a variable and assigns a value to it.

Example.

```
>>> s = 'Trinity College'
>>> print s
Trinity College
```


Variables and types

The *type* of a variable is the type of its value.

Example.

```
>>> n = 1823
>>> type(n)
<type 'int'>
```


Variables and types

The *type* of a variable is the type of its value.

Example.

```
>>> s = 'Trinity College'
>>> type(s)
<type 'str'>
```

Variable names

A variable can be named by any word (except for certain reserved words).

A variable name can contain any letter or digit, but it must begin with a letter.

To name a variable using more than one word, it is customary to use _ to connect words.

Variable names

Example.

```
temp = 33
exam3 = 89
my_college = 'Trinity College'
```


Variable names

Here are some of the *reserved words* that cannot be used to name variables:

```
and, as, assert, break, class,
continue, def, del, elif, else,
except, exec, finally, for, from,
global, if, import, in, is, lambda,
not, or, pass, print, raise, ...
```

Statements

A *statement* is the smallest unit of code that can be executed as a stand-alone instruction.

Example.

```
print 'Hello, World!'
```


Statements

A *statement* is the smallest unit of code that can be executed as a stand-alone instruction.

Example.

```
10 + 11
```


Statements

A *statement* is the smallest unit of code that can be executed as a stand-alone instruction.

Example.

```
n = 1823
```

Expressions

An *expression* is a code fragment that has a value.

Example.

```
1823
```


Expressions

An *expression* is a code fragment that has a value.

Example.

`1823 + n` (assuming `n` has a value)

Expressions

An *expression* is a code fragment that has a value.

Example.

```
'Trnity College'
```

Expressions

An *expression* is a code fragment that has a value.

Non example.

```
n = 1823
```


Expressions

An *expression* is a code fragment that has a value.

Non example.

```
print 'Trinity College'
```


Statement and expressions

An *expression* is usually part of a statement.

Example. This statement

```
n = 11 * (2 + 3)
```

contains an expression `2 + 3`.

Statement and expressions

An *expression* is usually part of a statement.

Example. This statement

```
print 'Trinity College'
```

contains an expression `'Trinity College'`.