Lecture 1 - Introduction to Programming

Adnan Ferdous Ashrafi

Stamford University Bangladesh

Introduction

Thinking like a computer scientist

The single most important skill for a computer scientist is problem solving. Problem solving means the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately. As it turns out, the process of learning to program is an excellent opportunity to practice problem-solving skills.

What is a Program?

A **program** is a sequence of instructions that specifies how to perform a computation. The computation might be something mathematical, such as solving a system of equations or finding the roots of a polynomial, but it can also be a symbolic computation, such as searching and replacing text in a document or something graphical, like processing an image or playing a video.

What is a Program?

The details look different in different languages, but a few basic instructions appear in just about every language:

input: Get data from the keyboard, a file, the network, or some other device.
output: Display data on the screen, save it in a file, send it over the network, etc.
math: Perform basic mathematical operations like addition and multiplication.
conditional execution: Check for certain conditions and run the appropriate code.

List of Programming Languages

These are the programming languages that you will learn in different courses in Stamford University Bangladesh.

- Python
- C
- C++
- Java
- C#
- PHP
- JavaScript
- and at least a thousand more.

Python Programming Language

Python

The Python **interpreter** is a program that reads and executes Python code.

There are two versions of Python, called Python 2 and Python 3. They are very similar, so if you learn one, it is easy to switch to the other. In fact, there are only a few differences you will encounter as a beginner.

Installing Python

The steps that you need to do for installing python interpreter and IDE (integrated development environment) are as follows:

- Go to Anaconda Individual Edition Download
- Install Anaconda
- Go to Start Menu $\rightarrow$ Python 3.7 $\rightarrow$ Python 3.7 (32/64 bit)
- Go to Start Menu $\rightarrow$ Anaconda3(64 bit) $\rightarrow$ Jupyter Notebook (Anaconda3)
- You are done!

Installing Python

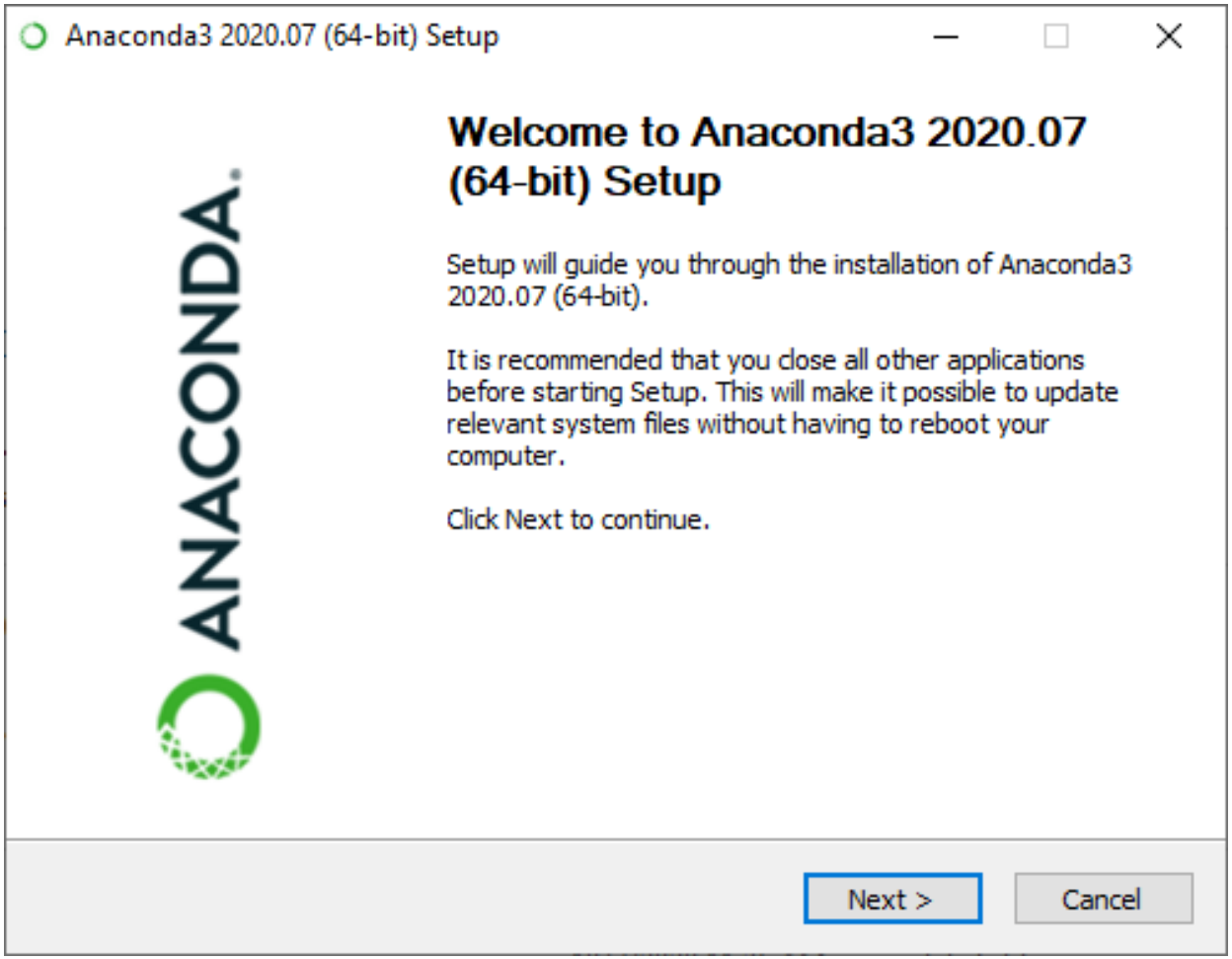

Figure 1: Installing Anaconda

Installing Python

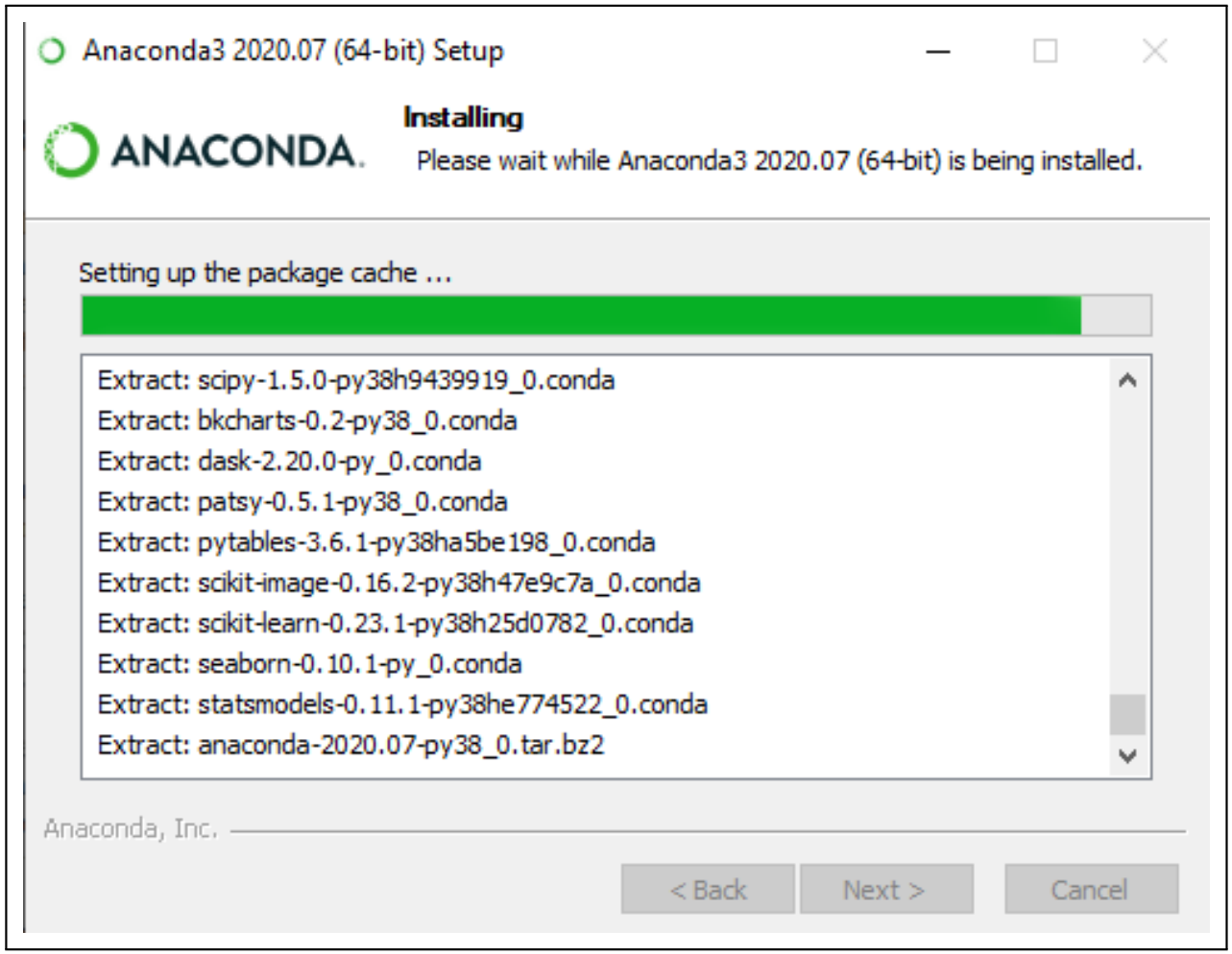

Figure 2: Installing Anaconda

Installing Python

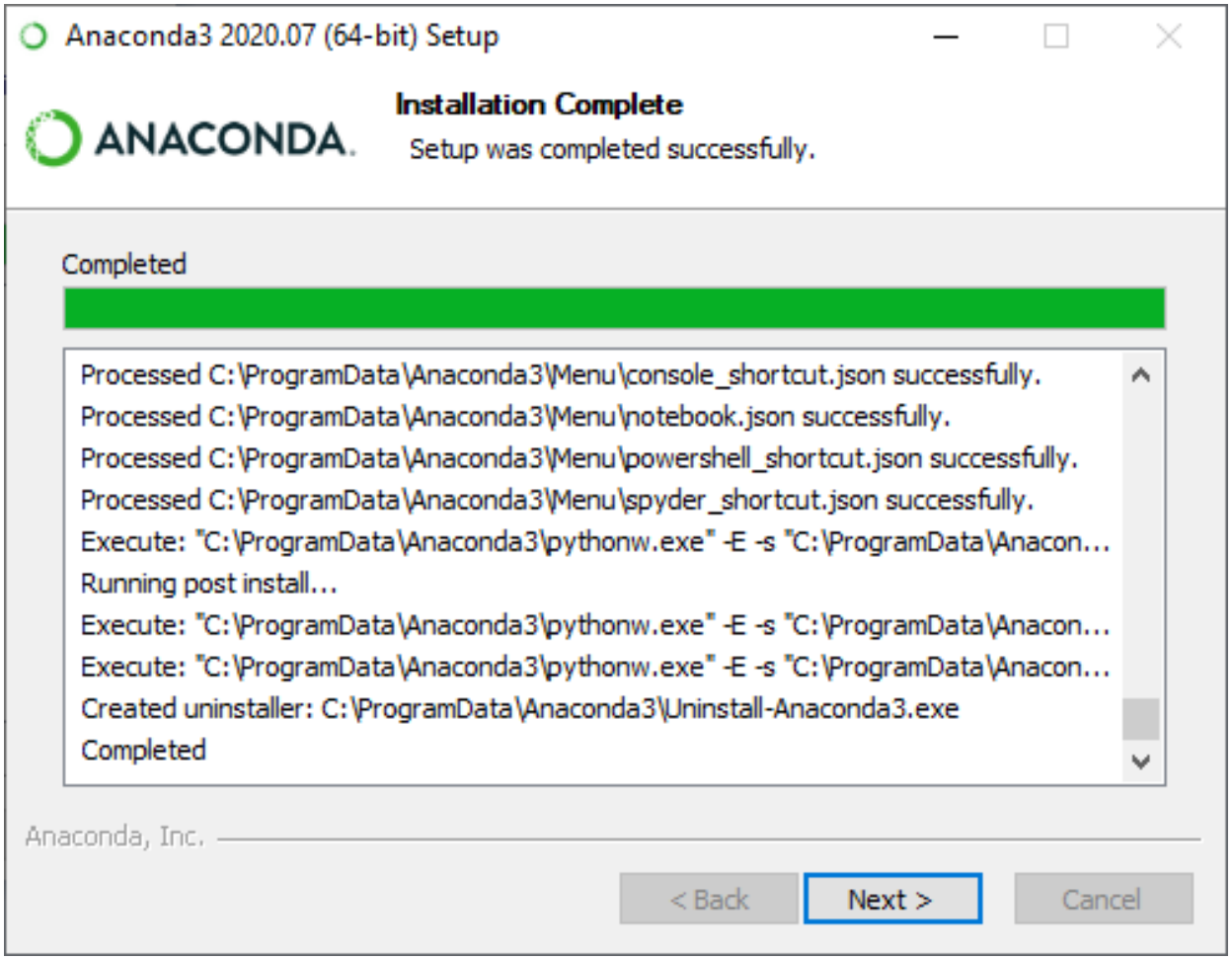

Figure 3: Installing Anaconda

Starting the Python interpreter

You can run python program in two environments. These are:

- **Python 3.7 Command Line**: Go to Start Menu $\rightarrow$ Python 3.7 $\rightarrow$ Python 3.7 (32/64 bit) and start typing your code directly into the command line.
- **Jupyter Notebook**: Go to Start Menu $\rightarrow$ Anaconda3 $\rightarrow$ Jupyter Notebook (Anaconda3) $\rightarrow$ New $\rightarrow$ Python 3 and start typing your code into the web browser.

Starting the Python interpreter

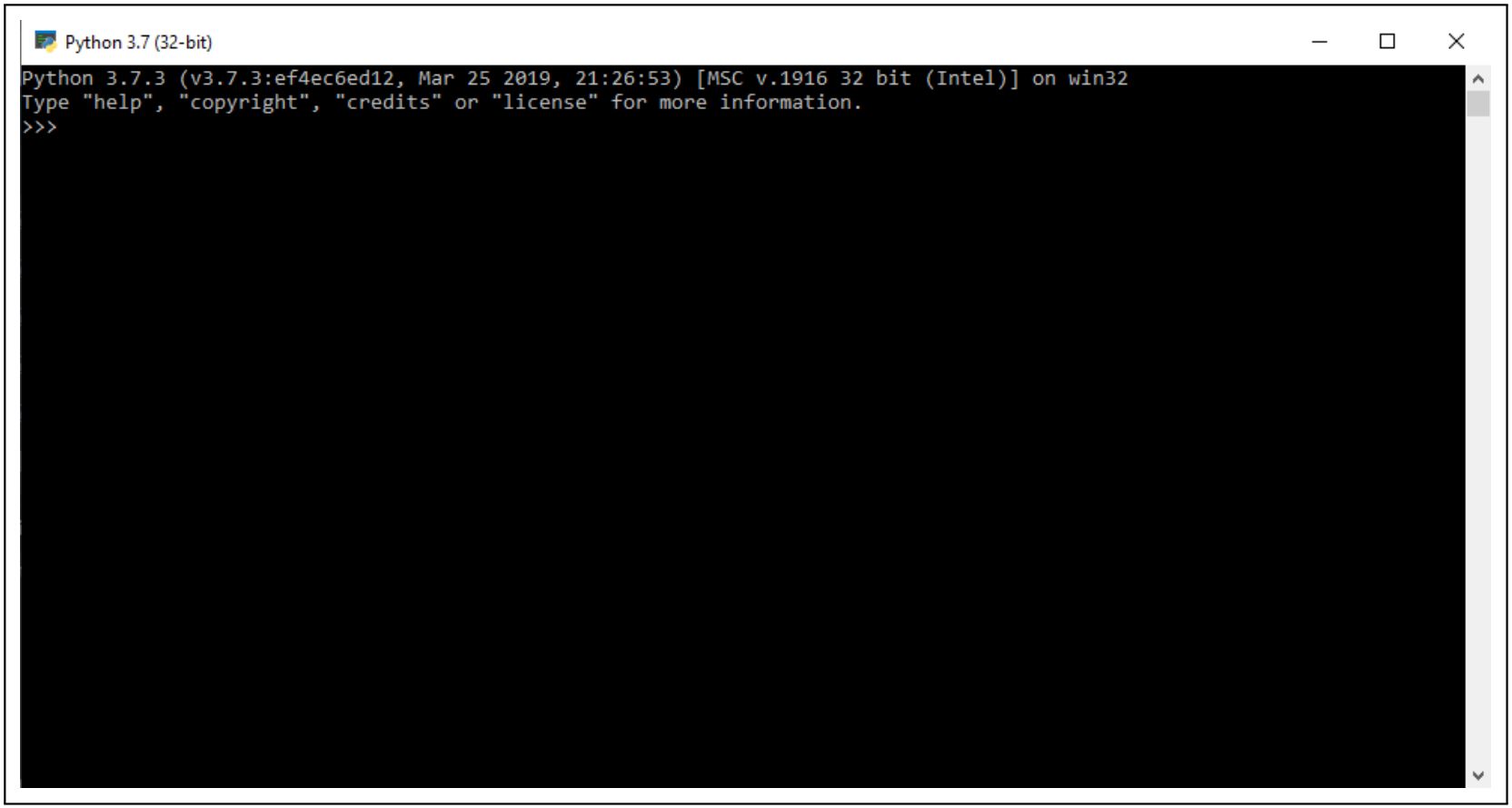

Figure 4: Running Python 3 command line

Starting the Python interpreter

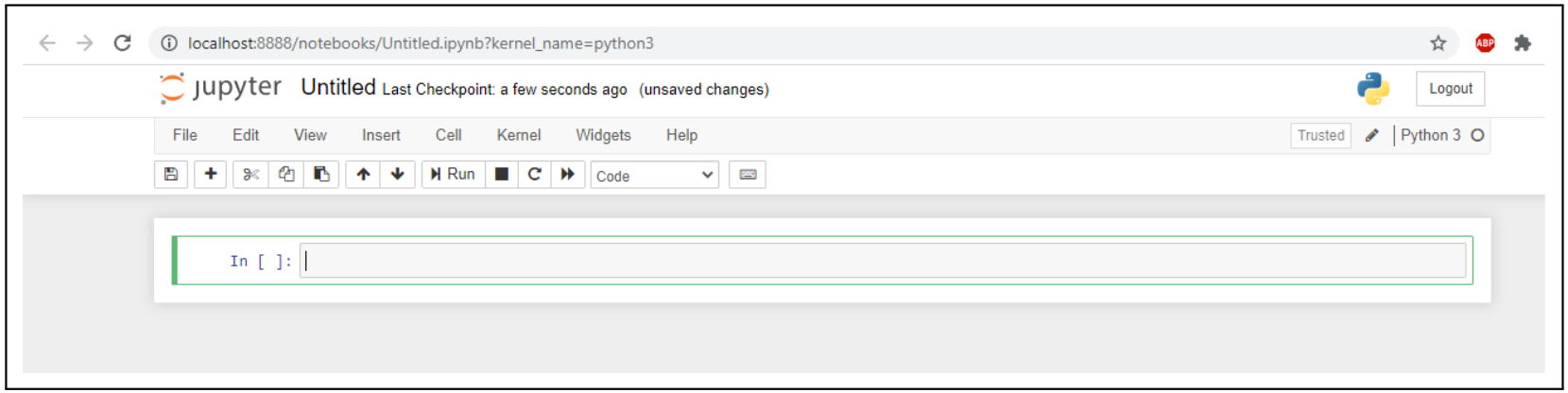

Figure 5: Running Jupyter Notebook from browser

Running your first program

```
1 print("Hello World")
2 print("This is my first program in Python")
```

Figure 6: Output of first program

Example of printing prime numbers

```
1  import math as m
2  lower = 1
3  upper = 1000
4  print("Prime numbers between", lower, "and", upper, "are:")
5
6  for num in range(lower, upper + 1):
7      # all prime numbers are greater than 1
8      if num > 1:
9          for i in range(2, int(m.sqrt(num))):
10             if (num % i) == 0:
11                 break
12         else:
13             print(num)
```

Can you find the output yourself?

Formal vs. Natural Languages

Natural languages are the languages people speak, such as English, Spanish, and French. They were not designed by people (although people try to impose some order on them); they evolved naturally.

Formal languages are languages that are designed by people for specific applications. For example, the notation that mathematicians use is a formal language that is particularly good at denoting relationships among numbers and symbols. Chemists use a formal language to represent the chemical structure of molecules. And most importantly:

Programming Languages

Programming languages are formal languages that have been designed to express computations.

Syntax, Tokens and structures

Formal languages tend to have strict **syntax** rules that govern the structure of statements. For example, in mathematics the statement $3+3=6$ has correct syntax, but $3+=3\$6$ does not. In chemistry H_2O is a syntactically correct formula, but ${}_2Zz$ is not.

Syntax rules come in two flavors, pertaining to **tokens** and **structure**. Tokens are the basic elements of the language, such as words, numbers, and chemical elements. One of the problems with $3+=3\$6$ is that \$ is not a legal token in mathematics (at least as far as I know). Similarly, ${}_2Zz$ is not legal because there is no element with the abbreviation Zz.

The second type of syntax rule pertains to the way tokens are combined. The equation $3+/3$ is illegal because even though $+$ and $/$ are legal tokens, you can't have one right after the other. Similarly, in a chemical formula the subscript comes after the element name, not before.

Parsing

Let's see two interesting examples:

This is @ well-structured Engli$h sentence with invalid t∗kens in it.

This sentence all valid tokens has, but invalid structure with.

When you read a sentence in English or a statement in a formal language, you have to figure out the structure (although in a natural language you do this subconsciously). This process is called **parsing**.

Differences between formal and natural language

Although formal and natural languages have many features in common—tokens, structure, and syntax—there are some differences:

ambiguity: Natural languages are full of ambiguity, which people deal with by using contextual clues and other information. Formal languages are designed to be nearly or completely unambiguous, which means that any statement has exactly one meaning, regardless of context.

redundancy: In order to make up for ambiguity and reduce misunderstandings, natural languages employ lots of redundancy. As a result, they are often verbose. Formal languages are less redundant and more concise.

literalness: Natural languages are full of idiom and metaphor. If I say, "You are the apple of your parents eyes", there is probably no apple and nothing related to eyes (this idiom means that you are the favourite person to your parents). Formal languages mean exactly what they say.

Debugging

Definition

Programmers make mistakes. For whimsical reasons, programming errors are called **bugs** and the process of tracking them down is called **debugging**.

Learning to debug can be frustrating, but it is a valuable skill that is useful for many activities beyond programming.

Glossary

problem solving: The process of formulating a problem, finding a solution, and expressing it.
high-level language: A programming language like Python that is designed to be easy for humans to read and write.
low-level language: A programming language that is designed to be easy for a computer to run; also called "machine language" or "assembly language".
portability: A property of a program that can run on more than one kind of computer.
interpreter: A program that reads another program and executes it.
prompt: Characters displayed by the interpreter to indicate that it is ready to take input from the user.
program: A set of instructions that specifies a computation.

Glossary

print statement: An instruction that causes the Python interpreter to display a value on the screen.
operator: A special symbol that represents a simple computation like addition, multiplication, or string concatenation.
value: One of the basic units of data, like a number or string, that a program manipulates.
type: A category of values. The types we have seen so far are integers (type int), floatingpoint numbers (type float), and strings (type str).
integer: A type that represents whole numbers.
floating-point: A type that represents numbers with fractional parts.
string: A type that represents sequences of characters.

Glossary

natural language: Any one of the languages that people speak that evolved naturally.
formal language: Any one of the languages that people have designed for specific purposes, such as representing mathematical ideas or computer programs; all programming languages are formal languages.
token: One of the basic elements of the syntactic structure of a program, analogous to a word in a natural language.
syntax: The rules that govern the structure of a program.
parse: To examine a program and analyze the syntactic structure.
bug: An error in a program.
debugging: The process of finding and correcting bugs.

Further Reading

- Chapter 1 of Think Python(2nd Edition) - Allen B. Downey
- Chapter 1 of Python Crash Course - Eric Matthes
- Python Official Website
- Lecture 1 Jupyter Notebook

Thank you.
Any Questions?