1 Background

Key terms: grammar comment

Exercise: Introduce yourself to your peers in the course forum.

Reading: Course policies[1]

Reading: Course philosophy[2]

CS131B is an introduction to the famous Python language, which was developed starting in 1989 by the Dutch mathematician and computer scientist Guido van Rossum. The advisory for this course is CS110A, which is a more basic introduction to programming. CS131B has no enforced prerequisites, but you will need some basic skills and resources: a conversational level of English; the ability to use the web LMS and to log into the UNIX server; and the willingness to read carefully.

Programs exist to compute at greater scale than is otherwise possible. They are used in nearly every field of endeavor. Your goal is to develop fluency in Python and notice related computer science topics. A dedicated novice completing this course is fluent enough to reach the major milestone of writing a useful program.

A Python program is composed of "plain" text. Some of the characters used are at hand on full-size computer keyboards, but harder to call up using tiny or on-screen input devices. The characters consist of vaguely English-language names written in all lowercase letters, the familiar arithmetic symbols of a calculator, matched pairs of several types of enclosures, and a few other elements we won't address here. A suggested style is formally guided by a document called `PEP-008`.[3]

All languages use grammar to encode semantic meaning. This means any messages not conforming to that grammar, whose meaning is inherently unclear, must be rejected. In natural language, we call these unintelligible constructs "gibberish" or "word salad", but we cannot always reject them out of hand. By tightly constraining program validity, computers can reliably do very complex operations which would be impractical to describe in natural language. In practice this means that many texts which are draft versions of Python programs do absolutely nothing, because they are not grammatically valid. They produce syntax errors, and when that happens, the programmer's first order of business is making the program valid again so that work can proceed on its logic.

Interleaved with rigid programmatic statements are sections of prose called "comments," which allow more flexible expression, because they are not subject to any grammar at all. Their main use is for you to store notes for yourself; you'd be surprised how confusing the source code you wrote last year can be. A description of a program's operation in comments is a kind of functional specification, which can be used to assess the program's correctness in testing. Noone regrets writing too many comments. They can also seal off problematic code and disable it on a temporary basis. A hash, `#`, makes anything between it and the next newline a comment. A matched pair of triple quotes, `'''` or `"""`, though not formally a comment, similarly disables any statements between them.

```
# Anything after the hash mark is a free text comment.

''' Strings can also be used as comments, in cases
where the content must span multiple lines.'''
```

Next, we will address the environment in which your programs will run.

[1] `https://fog.ccsf.edu/~abrick/policies.html`
[2] `https://fog.ccsf.edu/~abrick/pedagogy.html`
[3] `https://peps.python.org/pep-0008/`