PYTHON NOTES

print statement displays a result on the screen

The quotation marks in the program mark the beginning and end of the text to be displayed; they don’t appear in the result.

The parentheses eg. '' indicate that print is a function

A value is one of the basic things a program works with, like a letter or a number. Some values we have seen so far are 2, 42.0, and 'Hello, World!'.

These values belong to different types: 2 is an integer, 42.0 is a floating-point number, and 'Hello, World!' is a string, so-called because the letters it contains are strung together.

If you are not sure what type a value has, the interpreter can tell you:

>>> type(2)

<class 'int'>

>>> type(42.0)

<class 'float'>

>>> type('Hello, World!')

<class 'str'>

What about values like '2' and '42.0'? They look like numbers, but they are in quotationmarks like strings.

>>> type('2')

<class 'str'>

>>> type('42.0')

<class 'str'>

They’re strings.

When you type a large integer, you might be tempted to use commas between groups of digits, as in 1,000,000. This is not a legal integer in Python, but it is legal:

>>> 1,000,000

(1, 0, 0)

That’s not what we expected at all! Python interprets 1,000,000 as a comma-separated sequence of integers.

Programming languages are formal languages that have been designed toexpress computations.

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 H2O 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.

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.

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, “The penny dropped”, there is probably no penny and nothing dropping (this idiom means that someone understood something after a period of confusion). Formal languages mean exactly what they say.

1.9. Exercises

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. 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.