Outline

1. Formal Languages
   - syntax and semantics
   - Backus-Naur Form
2. Strings, Lists, and Tuples
   - composite data types
   - shared references

MCS 260 Lecture 6
Introduction to Computer Science
Jan Verschelde, 16 June 2023

languages, strings, lists, and tuples

1. Formal Languages
   - syntax and semantics
   - Backus-Naur Form

2. Strings, Lists, and Tuples
   - composite data types
   - shared references

Programming Languages

classified into 4 generations

1. machine language
   instructions are encoded as bit sequences
2. assembly language
   a mnemonic system for representing programs
3. high level programming languages
   similar to writing algorithms in pseudo code
   early examples: FORTRAN, lisp, Cobol, C
   object oriented languages: Ada, C++
4. framework languages, problem solving environments
   environment helps in discovery of algorithms,
   mostly limited to one specific problem area,
   Python comes with batteries included.

Formal Languages

syntax and semantics

Every language is defined through syntax and semantics.

- Syntax states rules how to compose valid sentences.
- Semantics define the meaning of the sentences.

Syntax and semantics of the assignment:

syntax an assignment consists of a variable, followed by $=$, followed by an expression.

semantics the assignment stores the value of the expression at the right of $=$ into the object referred to by the variable at the left.

Programming languages have a *context-free grammar*.

languages, strings, lists, and tuples

1. Formal Languages
   - syntax and semantics
   - Backus-Naur Form
2. Strings, Lists, and Tuples
   - composite data types
   - shared references

Backus-Naur Form

named after John Backus and Peter Naur

The Backus-Naur Form (or BNF) is a formal notation to define the syntax of programming languages.
The BNF of a language consists of

an alphabet is a finite set of symbols, containing terminal and nonterminal symbols.
Keywords are special terminal symbols.

rules of type $\mathbb{A} \ ::= \alpha$ where $\mathbb{A}$ is nonterminal, $::=$ is a reserved symbol (in BNF) and α is a string of terminal and nonterminal symbols.

the axiom is the initial symbol.

Sentences are derived by starting with the axiom.
Then we apply the rules, replacing the axiom by strings until the final string only consists of terminal symbols.

BNF of a Sum

Let us define the syntax of a sum of natural numbers.

Denote the axiom by `S` (initial or start symbol).

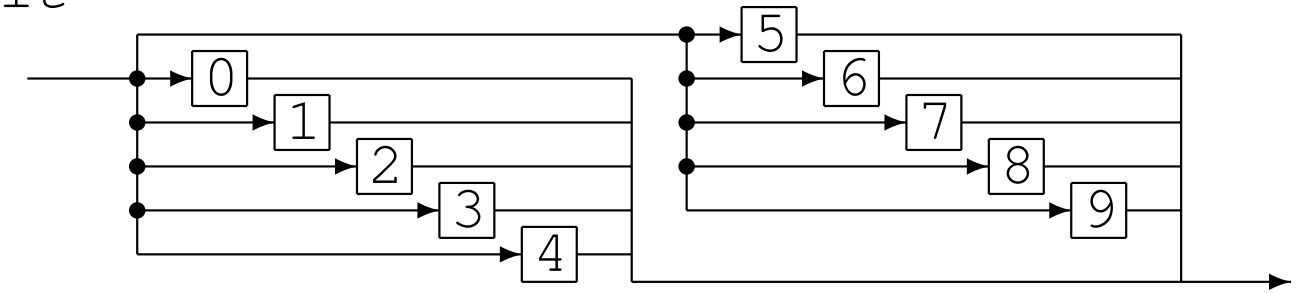
Nonterminal symbols are enclosed by `<` and `>`.

Vertical bars `|` indicate choice, they mean *or*.

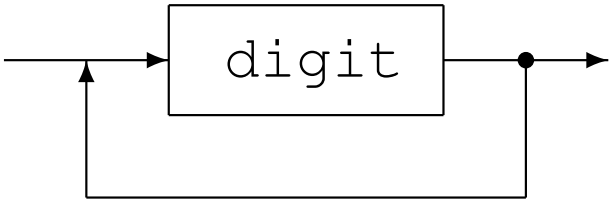
```
S ::= < number >|< number > + < sum >

< number > ::= < digit >|< digit >< number >

< digit > ::= 0|1|2|3|4|5|6|7|8|9

< sum > ::= < number >|< number > + < sum >
```

Syntax Diagrams

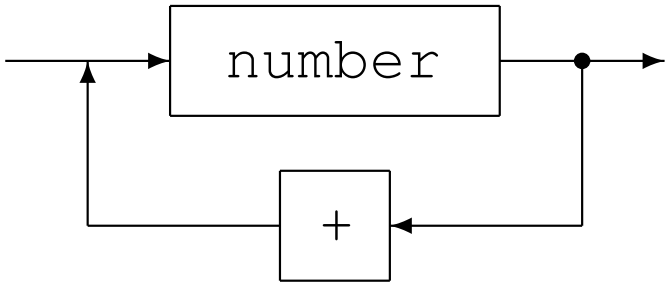
digit



number



sum



Syntax Errors

Interpreters and compilers always first *parse* the statements and check for syntactical correctness.

A *syntax error* means that the statement does not belong to the language.

In the second phase, for valid statements, the interpreter or compiler checks the semantics.

```
>>> float(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
```

Syntax error?

```
>>> flot(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'flot' is not defined
```

languages, strings, lists, and tuples

1. Formal Languages
   - syntax and semantics
   - Backus-Naur Form

2. Strings, Lists, and Tuples
   - composite data types
   - shared references

strings, lists, and tuples

String constants are sequences of characters, enclosed between right quotes (or double quotes).

```
>>> s = 'hello'
>>> L = [x for x in s]
>>> L
['h', 'e', 'l', 'l', 'o']
>>> T = tuple(L)
>>> T
('h', 'e', 'l', 'l', 'o')
```

Lists are a versatile compound data type:

- elements in a list can be of different type
- the length of a list is variable.

Tuples are lists of fixed length. Observe the brackets: `(`, `)` versus `[`, `]`.

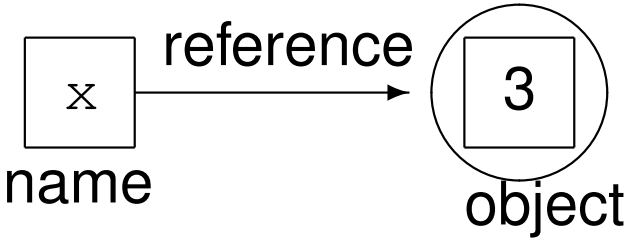
languages, strings, lists, and tuples

1. Formal Languages
   - syntax and semantics
   - Backus-Naur Form

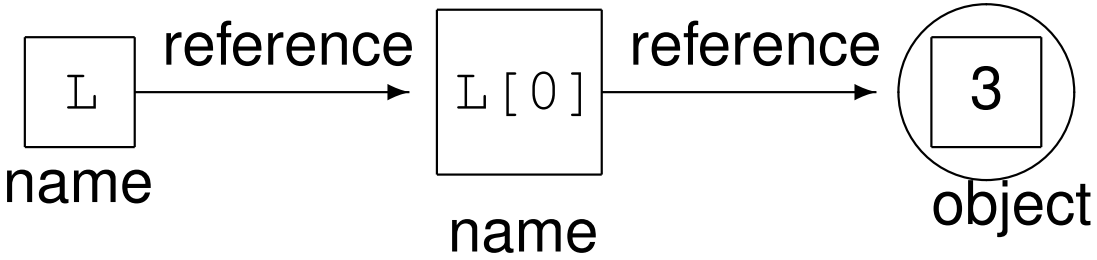
2. Strings, Lists, and Tuples
   - composite data types
   - shared references

lists and references

```
>>> x = 3
```

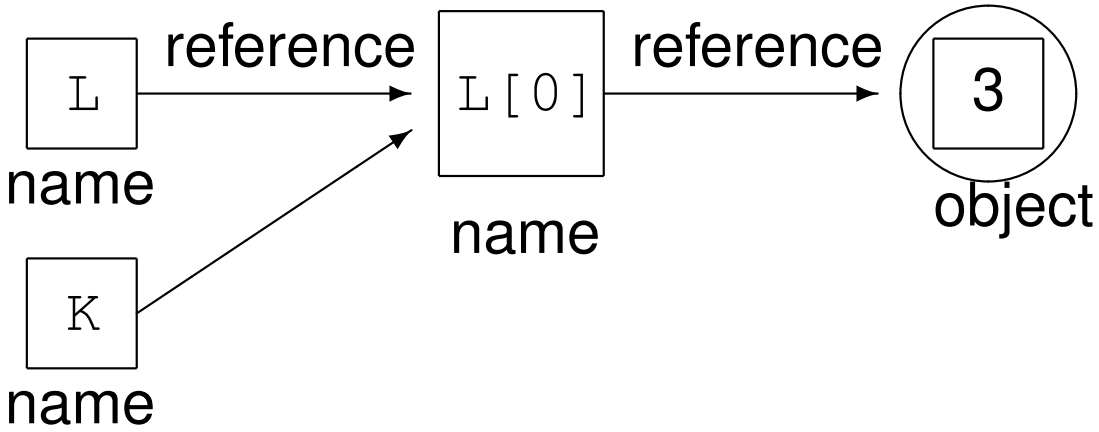



```
>>> L = [3]
```



sharing references

```
>>> L = [3]
>>> K = L
```




```
>>> K[0] = 4
>>> L
[4]
```

We changed `L` through `K`.

Exercises

1. Give the BNF of a floating-point number.
2. Extend BNF for a sum to that of a polynomial in one variable, in fully expanded form.
3. Find the BNF of the Python language.
4. Compare 2**3**4 with (2**3)**4 and explain.
5. Give the Python commands to convert `"06/16/2023"` into `"16-06-2023"`, *without* retyping `16`, `06`, or `2023`.