Assignment 2

Lexical Conventions and Grammars of Languages: C vs. Python vs. Javascript

SAI HARSHA KOTTAPALLI

CS17BTECH11036

C

There are six classes of tokens which are:

1. identifiers
2. keywords
3. constants
4. string literals
5. operators
6. Seperators

Identifiers

This token is a sequence of letters and digits where in this case, ‘letter’ includes ‘_’, is used to represent an entity in C.

Identifier always starts with letter instead of number.

Upper case and lower case letters are different.

Keywords

Few special identifiers are used as keywords which restricts their used as variable names, etc.
Example: auto, double. Int, struct , break, etc

Constants

a. Integer – constant

It is a sequence of digits which can be represented as decimal, as octal(using 0 as prefix) and hexadecimal(using 0x as prefix).
Suffix ‘U’ or ‘u’, denotes unsigned integer, while, Suffic ‘L’ and ‘l’ denotes long integer. These suffix can both be appended.
‘-’ is used a prefix for negative numbers.

b. Character – constant

It Is a sequence of one ot more characters enclosed in single quotes, which is translated to some numeric value based on machine’s character set at execution time.
As for multi-characters, it Is implementation defined.
Escape characters are denoted by the use of ‘\’ followed by letters or numbers(in decimal/octal/hexadecimal).
In some implementations, there is an extended set of characters that cannot be represented in the char type, so the set is represented using ‘L’ before the enclosing single quotes used to represent the character..

c. Floating – constant

This constant is a combination of integer part, a decimal part, a fraction part, an e or E, an optionally signed integer exponent and an optional type suffix ‘f’, ‘F’(float), ‘l’, or ‘L’(Long double).
Either fractional part ot integer part can be missing but not both together.

‘-’ is used a prefix for negative numbers.
Similarly is the case for ‘E’/’e’ and ‘.’

d. enumeration – constant
Set of values of type int represented as named constants.

String Literals

Also known as string constant, is a sequence of characters enclosed in double quotes.
This is an array of characters and of static storage class.
The string constant ends with a null character i.e. ‘\0’ to represent the ending of the string.
Using L as prefix with the double quotes allows us to used extended character as string literals.

Operators

It is used to perform specific mathematical or logical functions.
Supported operators:
a. Logical Operators (! , ||, etc)
b. Bitwise Operators (|, <<, etc)
c. Relational Operators (!=, ==, etc)
d. Arithmetic Operators (+, - , etc)
e. Assignment Operators (=, += , etc)
f. Misc Operators (sizeof(), ?:, etc)

Seperators

Separators are used to separate one token from other. Such as separating keyword from keyword, keyword from identifier, identifier from other identifier, etc.

Comments

The characters /* and */ are used for commenting, they are used as a pair.

These comments do not nest, and they are not treated so if they occur within a string or character literals.

Python

Line Structure

Each line in the program ends with a ‘\n’ or a newline representing a statement.

In case a long line needs to be written, it is often preferred to break up into two lines using ‘\’ as the line continuation character

Although the line continuation character is not needed when using parentheses (...), brackets [...], braces {…} or triple quotes as these are used to signify the beginning and ending of the definition.

Indentation

The general body of the python program consists of blocks of code with uniform indentation to represent the bodies of functions, loops, etc.

The amount of indentation is not fixed and left to user to decide but it is required to have the same amount of indentation for one block of code.

If the body of a function, conditional, loop, or class is short and contains only few statements(preferably only one statement), we can place the next statement directly after it.

‘pass’ is used to represent empty block.

Spaces are preferred over tabs as suggested by the python programming community.

Running python with flag ‘t’ throws warnings and ‘tt’ throws error with the usage of tabs.

For multiple statements in a line, ‘;’ is used as the delimiter, although this style is not preferred.

Identifiers

The name used to represent functions, variables, etc.

It is a sequence of letters and digits where in this case, ‘letter’ includes ‘_’, is used to represent an entity.
Identifier always starts with letter instead of number.

Upper case and lower case letters are different.

Identifiers starting with single ‘_’ is not imported when used as a module.

Identifier starting with ‘__’ are used for private class members.

Few identifers of the form, __init__ are used for some special methods.

Keywords

Few special identifiers are used as keywords which restricts their used as variable names, etc.
Example: and, elif, global, etc

Literals

a. Booleans
The identifiers True and False are interpreted as Boolean values with the integer values of 0 and 1, respectively

b. Integers
It is a sequence of digits which can be represented as decimal, as octal(using 0 as prefix) and hexadecimal(using 0x as prefix).
Suffix ‘L’ or ‘l’ is used to denote Long integers.
‘-’ is used a prefix for negative numbers.

c. Long Integers
Suffix ‘L’ or ‘l’ is used to denote Long integers.
‘-’ is used a prefix for negative numbers.
Example: 1234L

d. Floating point numbers
This constant is a combination of integer part, a decimal part, a fraction part, an e or E, an optionally signed integer exponent and an ‘-’ is used a prefix for negative numbers. optional type suffix ‘f’, ‘F’(float), ‘l’, or ‘L’(Long double).

e. Complex numbers
An integer or floating-point number with suffix ‘j’ or ‘J’ represents a complex number.

Two types of string literals are supported -
a. 8-bit character data (ASCII)
By default, 8-bit string literals are defined by enclosing text in single (‘), double (“), or triple (‘’’ or “””) quotes.

b. Unicode (16-bit-wide character data)
It represent multibyte international character sets and allow for 65,536 unique characters.
Unicode string literals are defined by preceding an ordinary string literal with a u or U, denoted as u”hello”.

The Backslash character is used for escape characters.
Unrecognized escape sequences are left in the string unmodified and include the leading backslash

Raw strings are denoted by r’<string literal>’, although r”\” is not allowed.
All the backslash characters are left intact in raw strings.

Adjacent Strings are automatically concatenated to form single string literal, with any mix of ordinary, raw, and Unicode strings(Unicode taking precedence)
If Python is run with the -U command-line option, all string literals are interpreted as Unicode.
Values enclosed in square brackets [...], parentheses (...), and braces {…} denote lists, tuples, and dictionaries.

Operators

It is used to perform specific mathematical or logical functions.
Supported operators:
a. Logical Operators (! , ||, etc)
b. Bitwise Operators (|, <<, etc)
c. Relational Operators (!=, ==, etc)
d. Arithmetic Operators (+, - , etc)
e. Assignment Operators (=, += , etc)
f. Misc Operators (sizeof(), ?:, etc)

Delimiters

( ) [ ] { } , : . ` = ;

Special Symbols

‘ “ # \ @

Documentation Strings

Serves for the purpose of documentation.

The first statement of a module, class or function, should be a string.

Decorators

Decorators are denoted with the @ symbol and must be placed on a separate line immediately before the corresponding function or method.

purpose of it is to modify the behavior

Source Code encoding

When the special coding: comment is supplied, Unicode string literals may be specified directly in the specified encoding(at the top of program)
Example:

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
```

Comments

The '#' character denotes the start of comments where everything to the right of it is commented out, except the case where it occurs in string literals.

Javascript

Whitespace and comments

Used to seperate different tokens.

Two types of comment styles -

a. block comments formed with /* */ (do not nest)

b. line-ending comments starting with //.

Second type of commenting style is preferred and suggested.

Names

The name used to represent statements,, variables, etc.

It is a sequence of letters and digits where in this case, ‘letter’ includes ‘_’, is used to represent an entity.
Identifier always starts with letter instead of number.

Upper case and lower case letters are different.

Few special identifiers are used as keywords which restricts their used as variable names, etc.

Numbers

JavaScript has a single number type. Internally, it is represented as 64-bit floating point.
With this, short integers issues are solved and there is no need to worry about overflow and many other errors are avoided.
‘-’ is used a prefix for negative numbers.
NaN is not equal to any value, including itself. It is used to represent ‘Not a Number’.

Infinity is a value greater than 1.79769313486231570e+308.

Strings

A string literal can be wrapped in single quotes or double quotes.
The <string literal> can be empty or contain any number of characters.
The '\' backslash serves as escape character.
JavaScript does not have a character type.
All characters are Unicode-16.
The \u convention allows for specifying character code points numerically.
Strings are immutable, although,new strings can be genrated with concatenation using '+'.
Two strings containing exactly the same characters in the same order are considered to be the same string.

Statements

Javascript has a common global namespace for different scripts.
When used inside of a function, the var statement defines the function's private variables, otherwise global variables.
A label can be assigned for switch, while, for, and do.
Statements tend to be executed in order from top to bottom.
A block is a set of statements wrapped in curly braces.
False is equivalent to:
a. false
b. ''
c. 0
d. NaN
All other values are taken as true.

Operators

WRT precedence

. [] (

delete new typeof + - !

* / %

+ -

>= <= > <

=== !==

&&

| |

? :

Expression

It is of the form -

A literal value or variable or (true, false, null, undefined ,NaN, Infinity), an invocation expression with ‘new’, refinement expression with delete, an expression wrapped in parathesis or preceded by prefix operator followed by -

a. Infix operator and expression

b. ternary operator ?:

c. Invocation (results in execution of a function value)

d. Refinement (specify a property or element of an object or array.)

Functions

Defines a function value, which can have an optional name with an ability to make calls recursively.

Object Literals

Properties are specified as names or strings.

The names of properties must be known aat compile time.