Semantic Rules and Sample Problem (identifier, literal and function call)

This reading contains the current semantic rules for lesson 2.07. It also contains a sample problem that applies semantic analysis to a statement using these semantic rules.

New Rules: function call, identifier, literal

Expressions etc.

*** new * function call - fc**

1. **+** evaluate the identifier to obtain a function object
2. **^** if there is an expression, evaluate it to obtain an argument object
3. if there is an argument object, pass it into the function
4. evaluate the function code to obtain a result object

*** new * identifier - id**

if the identifier is in the namespace, dereference it to get the result object, otherwise report an error

*** new * literal - li**

create a new object for the literal

- A plus (**+**) denotes a step that uses another semantic rule.
- A caret (**^**) denotes a step that may use another semantic rule.
- A step without a caret (**^**) or plus (**+**) does not use another semantic rule.

Sample semantic analysis problem:

You will be asked to perform semantic analysis on a semantically valid Python statement, using the current semantic rules. Each rule is labeled with a two letter abbreviation you must use in your answers. For example, **id** denotes the identifier rule.

Some semantic rules do not use other semantic rules. If such a rule is applied, a left angle bracket indicates that a rule has started, its two letter abbreviation indicates which rule, and a right angle bracket indicates that the rule has also completed. For example, the identifier rule does not use other rules, so its application is denoted by:

<id>

Some semantic rules use other semantic rules. If such a rule is applied, a left angle bracket indicates that a rule has started, and its two letter abbreviation indicates which rule. However, since this rule uses other rules, no right angle bracket is used yet to indicate that the rule has completed. Other rules must be started and completed before this rule is completed. For example the function call rule, uses at least one other semantic rule, so its start is denoted by:

<fc

When a semantic rule that uses other semantic rules is completed, its two letter abbreviation indicates which rule, and a right angle bracket indicates that it has completed. For example the function call rule, uses at least one other semantic rule, so its completion is denoted by:

fc>

Here is a sample semantic analysis problem and its solution. The problem consists of 4 questions.

Question 1: This Python statement has valid semantics:

```
print('hello')
```

What is the first semantic rule that is started?

We use annotations to help solve this problem. The annotations are removed to create the final answer. You should also use annotations to help you obtain your answers and then remove them when you enter your answers.

Since **print('hello')** is a function call, the function call semantic rule is used. Since this rule uses other rules it only starts. This is denoted:

```
<fc starting print('hello')
```

We add what the interpreter is trying to match as an annotation at the end of this line, to help keep track of what is going on.

The answer does not include the annotation: starting print('hello')

The answer is:

<fc

Question 2: This Python statement has valid semantics:

```
print('hello')
```

What is the next semantic rule that is started, applied or completed?

In step 1 of the function call semantic rule, the identifier **print** is evaluated using the identifier rule. Since the identifier rule does not use any other rules, this semantic rule both starts and completes. This is denoted:

```
<fc starting print('hello')

  <id> evaluated print
```

The answer does not include the annotation: evaluated print

The answer is:

<id>

Question 3: This Python statement has valid semantics:

```
print('hello')
```

What is the next semantic rule that is started, applied or completed?

In step 2 of the function call semantic rule, the argument **'hello'** is evaluated using the literal rule. Since the literal rule does not use any other rules, this semantic rule both starts and completes. This is denoted:

```
<fc starting print('hello')

  <id> evaluated print

  <li> evaluated 'hello'
```

The answer does not include the annotation: evaluated 'hello'

The answer is:

<li>

Question 4: This Python statement has valid semantics:

```
print('hello')
```

What is the next semantic rule that is started, applied or completed?

In the remaining steps of the function call semantic rule, applied to **print('hello')**, no other evaluations occur. The last step includes "evaluate the function code" for the built-in **print** function. However, since the **print** function code is not part of your code, you cannot apply semantic analysis to it. Therefore, do not include its evaluation in your answer. The function call semantic rule is complete. This is denoted:

```
<fc starting print('hello')

  <id> evaluated print

  <li> evaluated 'hello'
```

```
fc> evaluated print('hello')
```

The answer does not include the annotation: evaluated print('hello')

The answer is:

fc>

Note that there are three left angle brackets, one for each time a semantic rule starts, and three right angle brackets, one for each time a semantic rule successfully completes. The number of left angle brackets must equal the number of right angle brackets, since each semantic rule that is used must start and complete.