From the Python Language Reference:

```
if_stmt ::= "if" expression ":" suite
          ( "elif" expression ":" suite )*
          ["else" ":" suite ]
```

- This so-called RULE is a “template” for Python if statements.
- Quoted symbols like “if”, “:”, “elif” and “else” stand for themselves...called TERMINALS.
- ( ... )* means take 0, 1, 2, ... of what's inside.
- [ ... ] means take 0 or 1 of what's inside.
- `expression & suite` are defined by other RULES on the Python Web Site.
The formal answer ...

if X:
  statement1
  statement2
elif Y:
  statement3
  statement4
else:
  statement5
  statement6

What is this? One if statement? If you were writing a Python interpreter, you'd ask: where would it end?

- It begins with if
- X is an expression. Then the :
- Two inner statements form a suite

Next, there's one (1) elif clause--”elif”, expression Y, the colon :, and another two statement suite.

Finally, there's one else clause--”else”, (NO expression), the colon :, and another suite.
Defining ( ... )* by simpler rules

- 0_or_more_elifs was expressed by
  ( "elif" expression ":" suite )*

- Here is another way to express ( ... )* idea:
  0_or_more_elifs ::= Λ (means empty string)
  | "elif" expression ":" suite 0_or_more_elifs

- The vertical bar | is read “or”

- The huge bunch of rules “0, or 1, or 2, etc.”
  is replaced by ONLY two rules. Computer
  Scientists love that kind of thing.
Definition
A Formal Grammar is:

- A set of symbols called TERMINALS. (They will be the symbols in the actual language's sentences. Python: if, else, :, elif, = and more.)

- A set of symbols called NON-TERMINALS. (They represent grammatical categories. Python: if_stmt, expression, suite and more.)

- A set of rewriting RULES, each of the form:
  left-hand-side ::= right-hand-side
  - There might be many rules with the same left-hand-side. The bar | notation for “or” is a shorthand.
The rules for using RULES

- When you want to derive an example of a category, write its non-terminal symbol.
- A rule is applicable when its left-hand-side appears as a contiguous substring on your pad.
- Rewrite the whole string on your pad except REPLACE the substring equaling the left-hand-side of an applicable rule by that rule's right-hand-side.
- Continue until only TERMINALS appear.
• This idea of defining a “language” as all terminal strings obtainable this way from a formal grammar was invented by Noam Chomsky. He invented the modern theory of linguistics.

• Main ideas: (1) All natural (i.e., human) languages have essentially the same internal structure. (2) That structure and particular languages can be described by more or less formal rules.
The Chomsky Hierarchy defined by restrictions on the rules.

- Type 0: (General grammar) Any rule is OK.
- Type 1: (Context sensitive): the left-hand-side length must be $\leq$ the right hand side length.
- Type 2: (Context free): the left-hand-side must only be a single non-terminal symbol.
- Type 3: (Regular): Be Type 2 AND the right-hand side must either be either a terminal or be 
  $<\text{one non-terminal}> ::= <\text{one terminal}> <\text{one-non-terminal}>$
  or $<\text{one non-terminal}> ::= \lambda$ (empty string)
| for “or” written out in rules...

0_or_more_elifs ::= \(\Lambda\) (means empty string)  
| "elif" expression ":" suite 0_or_more_elifs

The vertical bar | is read “or”

This merely abbreviates TWO RULES:

1. 0_or_more_elifs ::= \(\Lambda\)

2. 0_or_more_elifs ::=  
   "elif" expression ":" suite 0_or_more_elifs
Our way of expressing zero or more elif's is a context-free rule: (ZME abbr. “0-or-more-elifs”)

- **ZME ::= Λ** (means empty string)
  
  | "elif" expression ":" suite  ZME

  (The vertical bar | is read “or”)

- a ZME could be Λ or it could be, from “elif” expression ":" suite  ZME, replace ZME with Λ and get “elif” expression ":" suite

- We can ALSO get “elif” expression ":" suite “elif” expression ":" suite
Another ZME:

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite

“elif” expression ":" suite
In Python, ZME occur only in if_stmt's

“if” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite
“elif” expression ":" suite

(an if ... : elif ...: ... elif ...: ... doesn't
HAVE TO end with an else: .... )