# syntax

Document Link

# 1. No need to use semicolon for ending statement like other language.

```
# define main function to print out something
def main():
    i = 1
    max = 10
    while (i < max):
        print(i)
        i = i + 1

# call function main
main()
```

**So Python officially doesn't support ++ or --**

# 2. Divide line use backslash

Python uses a newline character to separate statements. It places each statement on one line.
However, a long statement can span multiple lines by using the backslash (\) character.
The following example illustrates how to use the backslash (\) character to continue a statement in the second line:

```
if (a == True) and (b == False) and \
   (c == True):
    print("Continuation of statements")
```

# 3. Identifier

Python identifiers are case-sensitive. For example, the **counter** and **Counter** are different identifiers.

# 4. **Keywords**

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

# 5. String literals

Python uses single quotes ('), double quotes ("), triple single quotes (''') and triple-double quotes (""") to denote a string literal.

The string literal needs to be surrounded with the same type of quotes. For example, if you use a single quote to start a string literal, you need to use the same single quote to end it.
The following shows some examples of string literals:

```
s = 'This is a string'
print(s)
s = "Another string using double quotes"
print(s)
s = ''' string can span
      multiple line '''
print(s)
s = “”” string can span use triple double code
      multiple line “””
print(s)
```

# Summary

A Python statement ends with a newline character.

Python uses spaces and indentation to organize its code structure.

Identifiers are names that identify variables, functions, modules, classes, etc..

Comments describe why the code works. They are ignored by the Python interpreter.

Use the single quote, double-quotes, triple-quotes, or triple double-quotes to denote a string literal.

operator

**Arithmetic Operators :**

| Operator | Description | a | b | Example | Result |
|---|---|---|---|---|---|
| + | Addition | 5 | 2 | a + b | 7 |
| - | Subtraction | 5 | 2 | a - b | 3 |
| * | Multiplication | 5 | 2 | a * b | 10 |
| / | Division (Floating) | 5 | 2 | a / b | 2.5 |
| // | Floor Division | 5 | 2 | a // b | 2 |
| % | Modulus (Remainder) | 5 | 2 | a % b | 1 |
| ** | Exponentiation | 5 | 2 | a ** b | 25 |

Assignment operator :

| Operator | Description | a | b | Example | Equivalent To | Result |
|---|---|---|---|---|---|---|
| += | Add and Assign | 5 | 2 | a += b | a = a + b | 7 |
| -= | Subtract and Assign | 5 | 2 | a -= b | a = a - b | 3 |
| *= | Multiply and Assign | 5 | 2 | a *= b | a = a * b | 10 |
| /= | Divide and Assign | 5 | 2 | a /= b | a = a / b | 2.5 |
| //= | Floor Divide and Assign | 5 | 2 | a //= b | a = a // b | 2 |
| %= | Modulus and Assign | 5 | 2 | a %= b | a = a % b | 1 |
| **= | Exponentiate and Assign | 5 | 2 | a **= b | a = a ** b | 25 |

## Comparison operators :

Python has six comparison operators, which are as follows:

- Less than ( `<` )
- Less than or equal to (`<=`)
- Greater than (`>`)
- Greater than or equal to (`>=`)
- Equal to ( `==` )

- Not equal to ( `!=` )

**Logical Operator :**

Python has three logical operators:

- `and`
- `or`
- `not`

**a and b**

| a | b | a and b |
|---|---|---|
| True | True | True |
| True | False | False |
| False | False | False |
| False | True | False |

**a or b**

| a | b | a or b |
|---|---|---|

| | | |
|---|---|---|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

**not a**

| **a** | **not a** |
|---|---|
| True | False |
| False | True |

# Basic

Python

Youtube link

1. For getting input we use
2. input(‘write input value’’) method. [ python ], scanf() [ for c] cin() for [c++] and new Scaner () for java
3. For comment we use = # for single line. For multiple line we use = … comment message … or “”” cmt message “””
4. Data type = int,float/double,char,string,bool,