

**1. Input:** `>>> print 'Hello, World!'`

**Output:** File "<stdin>", line 1 print 'Hello, world!' ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Syntax Error:  
Missing parentheses in call to 'print'. Did you mean print(...)?

### **Explanation**

The input produces a Syntax Error due to a lack of parentheses (). Therefore, the correct print statement should be `>>> print ('Hello, World!')`

**2. Input:** `>>> 1/2`

**Output:** 0.5

### **Explanation**

The input statement yields 0.5, which indicates that the division of the two integers '1' and '2' was performed, where the integer '1' was the dividend and the integer '2' was the divisor. Dividing 2 integers in Python 3 results in a floating-point number, of which 0.5 belongs.

**3. Input:** `>>> type(1/2)`

**Output:** <class 'float'>

### **Explanation**

The type ( ) function is used in Python 3 to determine the type of the value 1/2. Because dividing two integers usually yield a floating-point number, the type of the value is <class 'float'>, as shown in the output.

**4. Input:** `>>> print (01)`

**Output:** File "<stdin>", line 1 print (01) ^Syntax Error: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers

### **Explanation**

The input produces a Syntax Error since the leading zero in the number '01' is often detected as an octal number in Python. Nevertheless, python 3 always requires the octal literals to begin with '0o'. The syntax error, therefore, can be corrected by excluding the leading zero, hence the input statement should have been `>>> print (1)` to get an output as 1.

**5. Input:** `>>> 1/(2/3)`

**Output:** '1.5'

### **Explanation**

Python 3 often perform  $1/(2/3)$  as a nested division by beginning with '2/3' to get '0.6666666666666666', whose type belongs to a floating-point number. The next division involves dividing '1' by '0.6666666666666666' to get '1.5'.

**Question:** What are the similarities between formal and natural languages?