## ****CS1101 – Unit 1 Discussion Forum: Python Basics****

**Hello everyone,**

I hope you all are doing well!

For this discussion, I have tested the provided Python statements and here are my observations, explanations, and reflections based on the version of Python installed on my laptop.

### ****Python Version Check:****

import sys

print(sys.version)

**Output:**

3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (AMD64)]

This shows that I am using **Python 3.x**, which affects how some statements behave compared to Python 2.x.

### ****1st Code:****

print 'Hello, World!'

**Output:**

SyntaxError: Missing parentheses in call to 'print'. Did you mean print('Hello, World!')?

**Explanation:**  
In **Python 3**, print is a function and requires parentheses. The syntax shown is from **Python 2**, where parentheses were optional. In Python 3, it should be:

print('Hello, World!')

This reflects the change in print statement usage between Python 2 and 3, as mentioned in **Chapter 1 - The Way of the Program** in the book Think Python: How to Think Like a Computer Scientist.

### ****2nd Code:****

1/2

**Output:**

0.5

**Explanation:**  
In Python 3, the division operator / always performs **true division** and returns a **float** result. In Python 2, this would result in 0 due to **integer division** (unless one of the operands is a float). This behavior highlights an important improvement in Python 3 to avoid unintentional integer division errors.

### ****3rd Code:****

type(1/2)

**Output:**

<class 'float'>

**Explanation:**  
Because 1/2 results in 0.5, Python correctly identifies this as a **float**. The type() function returns the data type of the value provided.

### ****4th Code:****

print(01)

**Output:**

SyntaxError: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers

**Explanation:**  
Python 3 does not allow leading zeros in decimal integer literals because they can cause confusion with octal numbers. To specify octal numbers in Python 3, the prefix 0o must be used, like this:

print(0o1)

**Output:**

1

### ****5th Code:****

1/(2/3)

**Output:**

1.5

**Explanation:**  
According to **operator precedence (PEMDAS/BODMAS)**, the expression inside the parentheses 2/3 is evaluated first, which results in approximately 0.666.... Then 1 / 0.666... = 1.5. This demonstrates Python's correct handling of **floating-point division**, ensuring reliable arithmetic operations.

### ****Connection to Course Material:****

This task directly relates to the concepts discussed in **Unit 1: Python Basics**, such as syntax rules, data types, operators, and version differences. These topics are clearly presented in **"Think Python: How to Think Like a Computer Scientist" (Chapter 1 – The Way of the Program)**. For example, the errors shown in the first and fourth codes are a result of changes made from Python 2 to Python 3 to ensure clearer, more consistent syntax rules.

### ****Discussion Question:****

What would happen if we used the floor division operator // instead of the true division operator / in the expression 1/2? How would this affect the result, and why is it important to understand the difference between these two operators when performing arithmetic operations in Python?

Reference:

Downey, A. (2015). Think Python: How to think like a computer scientist (2nd ed.). O’Reilly Media. Retrieved from <https://greenteapress.com/wp/think-python-2e/>