

Unit 1: Getting Started with Python

Skill Builder 2: Editing, Variables, Expressions

In this lesson, you will work in the Python Editor, investigate the menus, use some mathematics operations, experience color highlighting, note special keypad changes, and work with different data types.

**Objectives:**

- Explore the menus
- Use some operators
- Learn about several of the basic types of variables
- Use the assignment statement (=)
- Experience the keyboard differences in the Editor

1. Either use the TI-Nspire document you created in Skill Builder 1 or start a new document and add a Python Editor. `print('Hello, World')` is left over from the previous lesson.

In the Editor, write the following statements:

```
x = 4 + 5
y = 3*x + 7
z = 5*x - 2*y
print(x, y, z)
```

The comma (,) is to the left of the letter 'O' on the keypad.

Notice that none of these statement produces any results...yet.



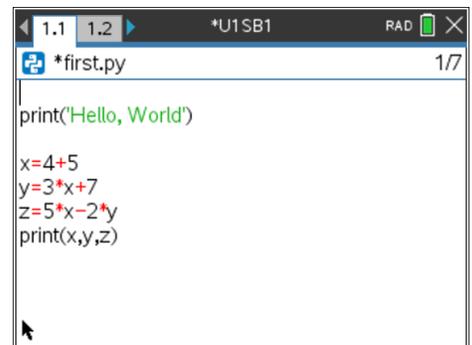
**Teacher Tip:** Note the **red** color highlighting of the operators. Also, you must use the multiplication sign. Python does not support implied multiplication like TI-Nspire does:  $5x$  is bad (syntax error).  $5*x$  is good.  $x5$  is the name of a variable in both worlds. '^' is a Python operator. But  $5^2 = 7$ ,  $6^2 = 9$ , and  $5^1 = 4$ . It stands for bitwise XOR. Avoid using ^ but be prepared!

2. *Variables* are letters or words that hold values. `=` is the 'assignment operator'. It stores the result of the expression to its right in the variable name on the left:

$$x = 4 + 5$$

stores the value 9 in the variable named x. The variable x can then be used in other expressions, like `z=5*x - 2*y` and the value of x is used.

Note: Python is case-sensitive. X and x are two different variables!

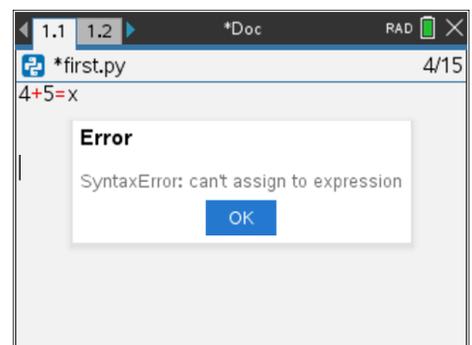


3. What happens if you try:

$$4 + 5 = x$$

Press **ctrl+B**

Not allowed! The expression must be on the right side.



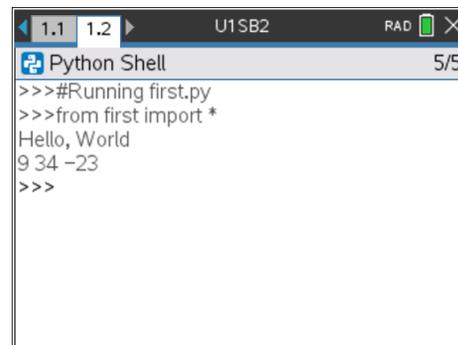
**Teacher Tip: Identifiers:** Python is case-sensitive. **X** is a different variable than **x**. Python programmers are free to use either UPPERCASE or lowercase variables or even a combination of the two, such as **HoursWorked** to make the program more readable. Also, the underscore ( `_` ) character is allowed in a **Variable\_Name** and you see it used a lot in the provided tools.

4. Press **ctrl+R** to run this code. If you used the exact same expressions, you will see:

```
Hello, World
9 34 -23
>>>
```

That is: x is 9, y is 34, and z is -23. Do the math!

Press **ctrl+left arrow** to go back to the Editor and try other expressions.



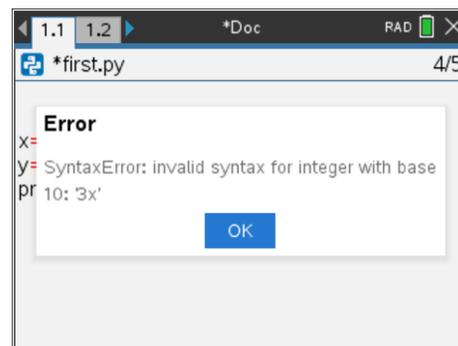
**Teacher Tip:** Programming errors come in three flavors. Interpreter errors are usually reported as a 'syntax error' and runtime errors are detected by the computer. The third kind of error is in the head of the programmer, e.g., entering an incorrect expression (like misuse of order of operations) or logical structure (like using 'and' where 'or' is required). Example:  $5+3/7+1$  vs.  $(5+3)/(7+1)$  (order of operations). Both are evaluated properly but give different results. What did the programmer intend?

5. Try the following:

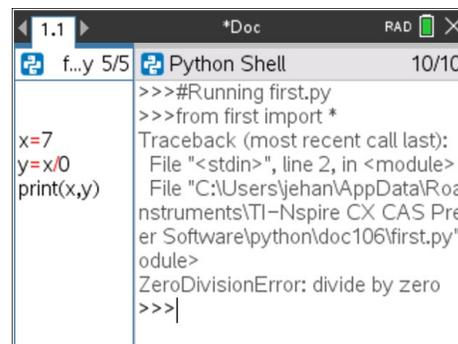
```
x = 7
y = 3x
print(x, y)
```

What do you get when you run this code?

Welcome to the world of computer programming! A 'Syntax Error' is an error in the text of the code. The English teacher calls it a grammatical error. '3x' is not allowed in Python even though TI-Nspire allows it. It must be  $3*x$ .



6. Programming errors come in three flavors. Interpreter errors are usually reported as a 'syntax error' and runtime errors are detected by the computer. The third kind of error is in the head of the programmer, e.g., entering an incorrect expression (like misuse of order of operations) or logical structure (like using 'and' where 'or' is required). Example:  $5+3/7+1$  vs.  $(5+3)/(7+1)$  (order of operations). Both are evaluated properly but give different results. What did the programmer intend?





# 10 Minutes of Code - Python

TI-NSPIRE™ CX II TECHNOLOGY

## UNIT 1: SKILL BUILDER 2

### TEACHER NOTES

- Return your code to the three expressions and the `print()` statement from step 1. To improve the appearance of the output, add a 'message' (such as `x=`) to each of the values:

```
print("x=",x," y=",y," z=",z)
```

There are some spaces in front of `y=` and `z=` inside the quotes.

Spaces are ignored by the interpreter but if they are inside quotes they will be printed as written.

Be *very careful* when entering this statement: the positions of the commas and the quotes is crucial. If any symbol is in the wrong position, you will see a 'Syntax Error' message when you press **ctrl+R**. You will have to find and correct the error, which is usually near or right above the cursor.

- When you run this program, you should now see:

```
x= 9   y= 34   z= -23
```

```
*U1SB1
RAD
*first.py 1/7
print("Hello, World")
x=4+5
y=3*x+7
z=5*x-2*y
print(x,y,z)
```

```
*U1SB2 v...prs
RAD
Python Shell 6/6
>>>#Running first.py
>>>from first import *
Hello, World
9 34 -23
x= 9   y= 34   z= -23
>>>|
```

**Teacher Tip:** There are other ways to format the output in Python, including the `.format()` feature which is not covered in this course. String concatenation is another method of improving output.