

Unit 1: Getting Started with Python

Skill Builder 3: Introducing the Python Function

In this lesson, you will define a function and use the function to evaluate expressions. You will experience the purpose of indentation in Python and the assistance of inline prompts in the Editor.

**Objectives:**

- Define a function
- Use the function in evaluating expressions
- Use the input() function

Functions play a big role in Python programming as in mathematics. Functions can be used to generate many different kinds of values and functions can serve as Python subroutines which help to break a complex process into smaller, manageable parts.

A program is a step-by-step recipe for solving a problem: an **algorithm**. All programs are algorithms underneath.

1. Write a program that lets the user enter a number for  $x$  and the program will use that value to evaluate the function  $f(x)=x^2 + 3x - 1$ .

First, define a function using **menu > Built-ins > Function > def function()**. The function template appears and the **inline prompts** *function*, *argument*, and *block* are provided and must be replaced with your own code.



```

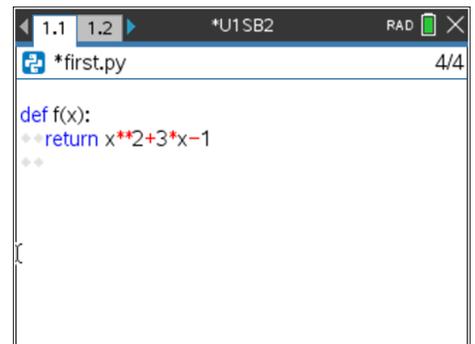
def function(argument):
    ++block
    {
    }
    
```

2. Notice that *function* is currently selected. Replace *function* with **f** by typing the letter **f**, press **tab** to replace *argument* with **x** and **tab** again to change block to:

**return x\*\*2 + 3\*x - 1**

**return** is found on **menu > Built-ins > Function**.

**x\*\*2** is Python notation for  $x^2$ , or **x\*x**  
and you *must* write **3\*x**, not **3x**.



```

def f(x):
    ++return x**2+3*x-1
    ++
    {
    }
    
```

Notice that block is indented two spaces. The light gray diamonds are 'placeholders' for you to keep track of the amount of indentation in more complex programs. Indenting is very important in Python.

We have finished defining the function. When you run this program, this function definition is not executed right away but the Shell 'knows' that it exists.

- Use an **input()** statement to allow the user to enter a number for x. Backspace (press the **del** key) to the beginning of a new, blank line and write:

**x=input("Enter a value for x: ")**

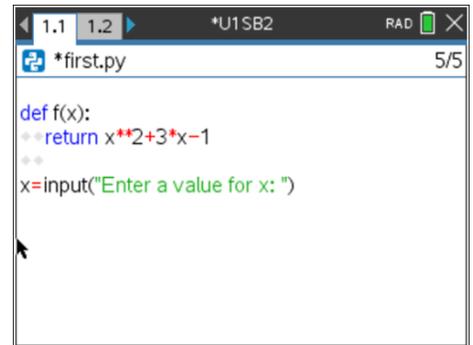
Use the = key on the keypad.

For **input()** press **menu > Built-ins > I/O** (or just type it in).

You can use either single (punctuation key) or double (**ctrl+x** (multiplication symbol)) quotes.

The green text in quotes is entered character by character from the keypad.

The colon (:) character is on the punctuation key. It is *not* required but makes the prompt look nice.



```

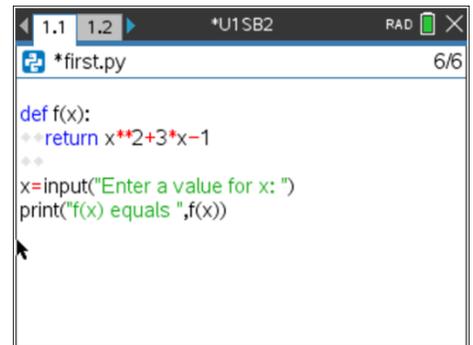
1.1 1.2 *U1SB2 RAD
*first.py 5/5
def f(x):
    return x**2+3*x-1
x=input("Enter a value for x: ")
    
```

- On the next line write the **print()** statement:

**print("f(x) equals " , f(x))**

Recall that **print()** is on **menu > Built-ins > I/O**.

The **f(x)** in quotes will be printed as **f(x)** but the **f(x)** after the comma is a *function call* and will be replaced with the value that is returned by the function.



```

1.1 1.2 *U1SB2 RAD
*first.py 6/6
def f(x):
    return x**2+3*x-1
x=input("Enter a value for x: ")
print("f(x) equals ",f(x))
    
```

- Press **ctrl+R** to run the program. At the prompt, enter a number for x and press **enter**.



```

1.1 1.2 *U1SB2 RAD
Python Shell 3/3
>>>#Running first.py
>>>from first import *
Enter a value for x: |
    
```

6. Welcome to (possibly) your first 'runtime' error. The message is lengthy but the important part is in the last two lines. Note the last line number and the TypeError: "can't convert..."

The error happens because the **input()** function returns a string, not a numeric value. The programmer (that's you) must convert the string to a numeric value.

There are five simple types of data in Python: **int** (integer), **str** (string), **float** (numbers with decimals), **complex** (complex numbers) and **bool** (Booleans are either True or False).

7. Fixing the error: There are several convert functions built into Python. They are found on **menu > Built-ins > Type**.

They are **int()**, **float()**, **str()** and **complex()**.

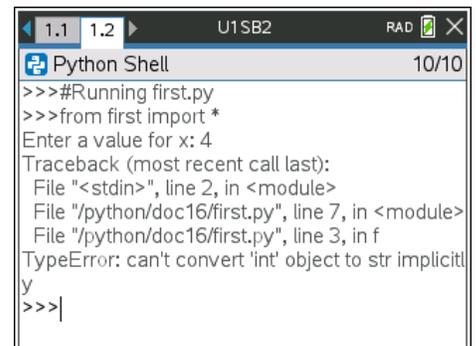
8. To convert the string *x* to a number, use

$x = \text{float}(x)$   
or  
 $x = \text{int}(x)$

just before the print statement. Try both and see the difference.

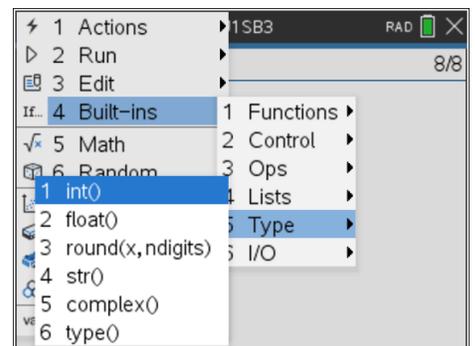
We chose **float()** in this example. Why?

9. Run the program again and it will work as intended.



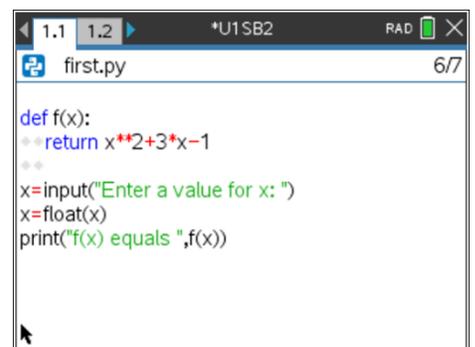
```

>>>#Running first.py
>>>from first import *
Enter a value for x: 4
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "/python/doc16/first.py", line 7, in <module>
  File "/python/doc16/first.py", line 3, in f
TypeError: can't convert 'int' object to str implicitly
>>>|
    
```



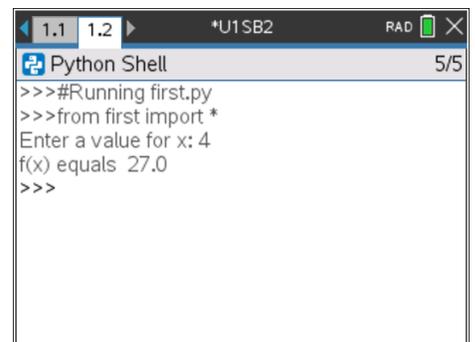
```

1 Actions
2 Run
3 Edit
4 Built-ins
5 Math
6 Random
1 int()
2 float()
3 round(x, ndigits)
4 str()
5 complex()
6 type()
    
```



```

def f(x):
    return x**2+3*x-1
x=input("Enter a value for x: ")
x=float(x)
print("f(x) equals ",f(x))
    
```

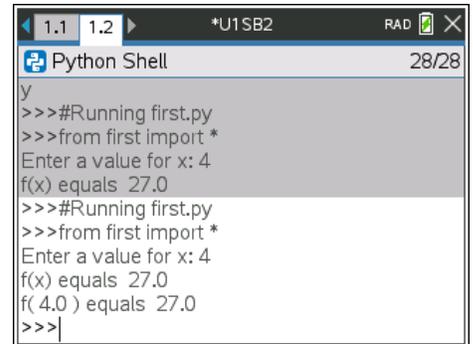


```

>>>#Running first.py
>>>from first import *
Enter a value for x: 4
f(x) equals 27.0
>>>
    
```

10. Challenge: How would you change the code to get the value of x to appear inside the function's parentheses (instead of the letter x) like this:

**f(4.0) equals 27.0**



```
Python Shell 28/28
y
>>>#Running first.py
>>>from first import *
Enter a value for x: 4
f(x) equals 27.0
>>>#Running first.py
>>>from first import *
Enter a value for x: 4
f(x) equals 27.0
f( 4.0 ) equals 27.0
>>>|
```