

Unit 2: Input, Output and Functions

Skill Builder 3: The Midpoint Formula

In this lesson, you will write and use a function that is used more than once

Objectives:

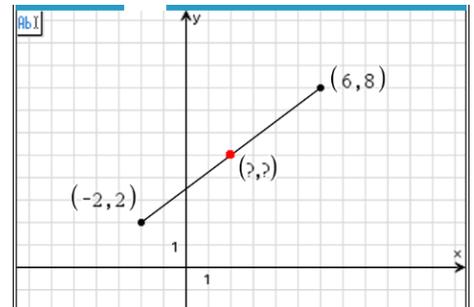
- Review the midpoint formula
- Use uppercase in a variable
- Store the result of a function in a variable
- Return two values from a function

In coordinate geometry there is a method for determining the coordinates of the **midpoint of a line segment** given the coordinates of the endpoints. Write a program that determines the midpoint's coordinates.

There are two midpoint formulas that are really the same formula (the arith**METIC mean** of two numbers) using different variables:

$$\text{midX} = (x_1 + x_2) / 2$$

$$\text{midY} = (y_1 + y_2) / 2$$



1. In a new Python file (our file is named MIDPOINT), select **<Fns...>** and use the **def function()**: template.

Name the function **midpt()**.

```

EDITOR: MIDPOINT
PROGRAM LINE 0002

def midpt():
**
    
```

Teacher Tip: The filename and the Python function name are separate entities apart from any and all identifiers within the Python code, such as variables and function names, so it is permissible to name the Python file “MIDPOINT” and a Python function within the file midpt(). However, this could lead to confusion in some (human) readers.

2. This time write the main program first, then come back to code the midpt() function last. Place your cursor at the *beginning* of a blank line below the function name. Write the four **input** statements to enter the four coordinates of the endpoints using x1, y1, x2, y2 as the four variables. Try it before moving to the next step.

Remember to take advantage of copy/paste. You can copy once, then paste three times.

```

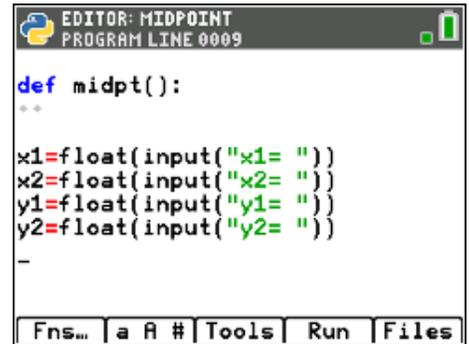
EDITOR: MIDPOINT
PROGRAM LINE 0005

def midpt():
**
-
    
```

3. The image on the right shows all four input statements. The first is:

x1 = float(input("x1 = "))

Copy and paste comes in handy here. With your cursor on the **x1=** line, select **<Tools> Copy Line** and then **select <Tools> Paste Line Below** three times. Then be sure to edit the pasted code to suit.



```

EDITOR: MIDPOINT
PROGRAM LINE 0009

def midpt():
..
x1=float(input("x1= "))
x2=float(input("x2= "))
y1=float(input("y1= "))
y2=float(input("y2= "))
-
Fns... a A # Tools Run Files
    
```

4. Write **two** assignment statements to store the result of the **midpt()** function into **two different** variables:

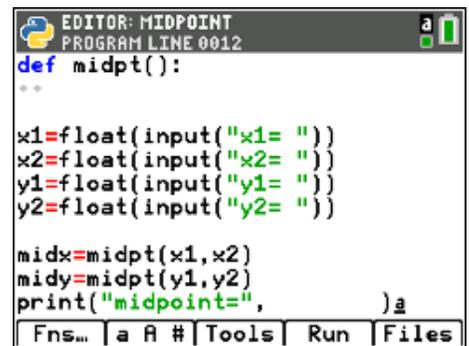
midX = midpt(x1, x2)
midY = midpt(y1, y2)

(We are using the *same function* for two *different* sets of variables.)

Write the **print** statement to display the complete midpoint:

print('midpoint = ',)

Can you complete this statement so that the output appears as an ordered pair? For example: midpoint = (13, 5)



```

EDITOR: MIDPOINT
PROGRAM LINE 0012

def midpt():
..
x1=float(input("x1= "))
x2=float(input("x2= "))
y1=float(input("y1= "))
y2=float(input("y2= "))

midx=midpt(x1,x2)
midy=midpt(y1,y2)
print("midpoint=", )a
Fns... a A # Tools Run Files
    
```

Teacher Tip: The coding of the print statement is very detailed regarding the use of string literals, commas and variables. An example is provided at the end of this lesson.

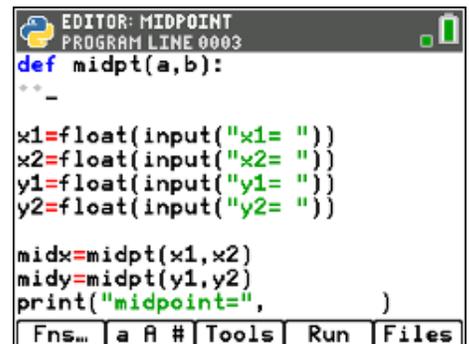
5. Now go back to complete the **midpt()** function.

Use two arguments. It is better to use two new variables as arguments:

midpt(a, b):



Try completing the function now. Then compare your work to the next step.



```

EDITOR: MIDPOINT
PROGRAM LINE 0003

def midpt(a,b):
..
-
x1=float(input("x1= "))
x2=float(input("x2= "))
y1=float(input("y1= "))
y2=float(input("y2= "))

midx=midpt(x1,x2)
midy=midpt(y1,y2)
print("midpoint=", )
Fns... a A # Tools Run Files
    
```

Teacher Tip: Using formal parameters that are different from the actual parameters helps to clarify the difference between the two, especially when a function is being called with different arguments as in this lesson.

6. The body of the function consists of one calculation:

```
◆ ◆ avg = (a + b) / 2
```

and the **return** statement:

```
◆ ◆ return avg
```

We use the variable **avg** because the expression is calculating the average or arithmetic mean of the two arguments.

Make sure that these two statements are indented the same amount.

Recall that **return** is found on <Fns...>.

Teacher Tip: Yes, you can also just write

```
◆ ◆ return (a + b) / 2
```

7. If you have completed the print statement, then run the program and enter four values at the prompts. Does your output resemble ours?

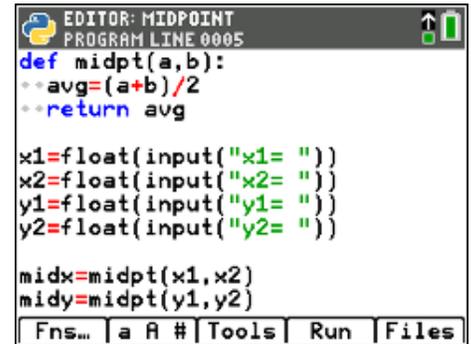
Summary: Python functions can be used many times during a program.

Displaying output in a pleasing format can be tricky. Our **print** statement is:

```
print( "midpoint = ( ", midX, ", ", midY, " )
```

with extra spacing to see where the quotes and commas belong.

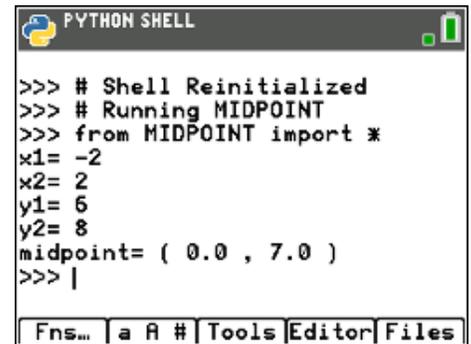
Why are there decimals in the output? Because we used **float()** in the input statements.



```
EDITOR: MIDPOINT
PROGRAM LINE 0005
def midpt(a,b):
    avg=(a+b)/2
    return avg

x1=float(input("x1= "))
x2=float(input("x2= "))
y1=float(input("y1= "))
y2=float(input("y2= "))

midx=midpt(x1,x2)
midy=midpt(y1,y2)
```



```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running MIDPOINT
>>> from MIDPOINT import *
x1= -2
x2= 2
y1= 6
y2= 8
midpoint= ( 0.0 , 7.0 )
>>> |
```

Teacher Tip: The output (2.0, 5.0) contains parentheses and a comma; this is how an ordered pair is usually depicted.

Another approach to printing the ordered pair is to use a tuple (not included in this course):

```
print('midpoint = ', (midX, midY))
```

which, if you are new to Python, actually does work because the structure (1,2) is a *tuple*, a valid data structure *similar* to a list, which is indicated using square braces, like [1,2]. But we're not ready to go there just yet!