some_py_stuff.py

```python
#This is a single line comment. A comment is anything written after a # on a single line.

print("Hi") #Single line comments can be on the same line as code as long as they are the last thing on that line.

'''
This is a multi line comment.
Multi line comments are anything between sets of 3 apostrophe.
Comments are ignored by the python interpreter and therefore do not execute at runtime. They are used for code documentation.
print("This code will not run")
'''

'''
The print function outputs its argument as a string to the console.
You can pass text directly to the function or you can pass it a string stored in a variable.
'''

print("This text was passed directly to the print function. Note the quotation marks.")

var = "This string was stored in a variable named var and passed to the print function. Note the lack of quotation marks in the function argument."
print(var)

'''
The input command prints its argument to the screen and then waits for user input to proceed.
If you don't pass it an arguement it will just wait for input.
Input can easily be stored to a variable.
'''

input("Enter some text: ") #With argument
input() #Without argument
var = input("Enter some text to print: ") #Storing input in var
print(var) #Printing stored input

'''
Lets use this to perform a task. We will create a small program that adds 2 numbers together.
If you don't enter numbers you will get an error. I will cover how to prevent this issue later but for now dont worry about it.
'''

num1 = int(input("Enter number 1: ")) #int() converts its argument to an integer (whole number). int(input()). It will produce an error if passed any value
num2 = int(input("Enter number 2: "))
num3 = num1 + num2
print("The sum is " + str(num3)) #str() works the same as int() except it converts the argument into a string rather than an integer.

'''
Next i will show you how to make your own functions.
To define a function we use def then the function name when in parentheses we place any arguments we wish to use.
It is possible to use an undeclared number of functions but I'll keep it simple for now.
'''

def divide_float(x, y): #Here we use def to define a function, name it divide_float and pass 2 arguments that gets stored as a variables called x and y. Ar
    div = int(x) / int(y) #Here we take the arguments stored as x and y, convert them to integers so we can perform math operations, divide them and store
                          #Note that the code inside the function is indented. This is so the function knows what is inside of it. Once the code reaches th
    return float(div) #Here is where the function returns a value. Functions dont have to return a value. If you do need to return a value you use return t
                      #A float is a floating point number or a number with a decimal value. float() works the same as str() and int().

print(str(divide_float(input("first number: "), input("second number: "))))
'''
The above line calls the function as an arguement for the print function. Functions arent run when they are located in your code rather they run when calle
So we have the print function getting called. As its argument we are passing in the function we made but converted into a string with str().
The arguments being passed to our divide_float function are input functions that will ask for user input when needed. The arguments are separated by a comm
If we run this code python will skip over the function definition and see the print function. It will try to read the argument and see that it is our funct
It will see that that functions arguments are also functions (input) so it will run the two input() functions first. Lets say we put in 10 and 2.
Now we have our arguments it goes back to running the divide_float function. divide_float(10, 2).
Now we are running the indented code inside our function. div = x / y which with agrs is div = 10 / 2.
Now the function returns the variable div but first it converts it into a float (decimal).
Now the code jumps back out to the print function now that it has a value for its argument. It converts the value we returned from our function into a stri
print(str(divide_float(10, 2))) --> print(str(5.0)) --> print("5.0")
'''
```