**[Skip Navigation](https://next.tech/projects/78066632-1c01-4dbc-a01d-6c72ce26c1a7?access_token=1BA03245BBDB8059276B51A507046B1F&step_id=ac999920-fa58-42af-9965-2bdedf2859cf" \l "left-sidebar)**

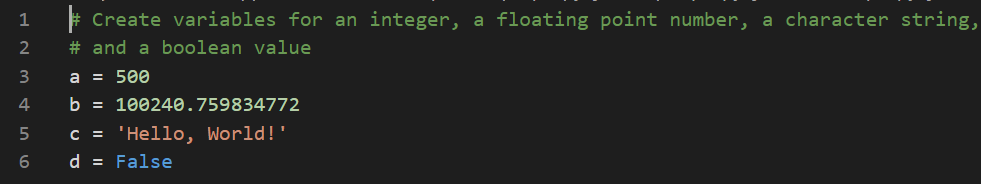
**Solution: Formatting Simple Data Types**

**Task 1**

1. Create variables for an integer, floating point number, character string and boolean value.

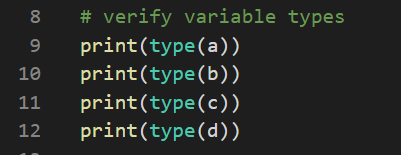
Lines 3 to 6

* Integer: a = 500,
* Floating point number: b = 100240.759834772
* Character string: c = 'Hello, World!'
* Boolean value: d = False

[](https://cdn.filestackcontent.com/LTL58EpkTFaDnvQeIGpA)

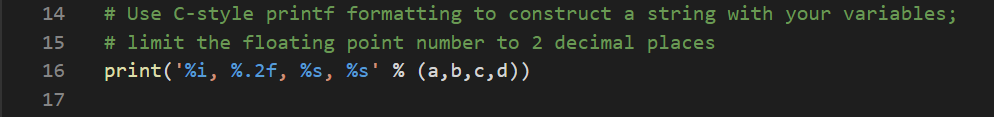
1. Verify the variable types done on lines 9 to 12. Print thetype of each variable.

The type function is a built-in method in Python which prints thetype of a variable.

[](https://cdn.filestackcontent.com/ZCrIBVnLREWPMJCkp6AM)

**Task 2**

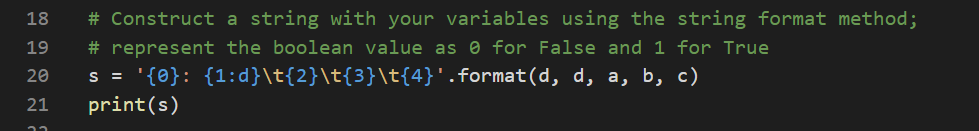
1. Use C-style printf formatting to construct astring with your variables. Make the floating point numberto two decimal places.
2. On line 16, print the string that has been constructed. Print the string with%i, %.2f. The %.2f is the formatting for floating point number using the C-style formatting that will limit the floating point number to two decimal places. %s is for the string c, and %s for the boolean value.
3. Then after the string, I put in the modulo operator (%). This is the string formatting or interpolation operator, which directs Python to use the C-style printfformatting. Then I have in parentheses the variables to be put into the string, so a, b, c, and d.

[](https://cdn.filestackcontent.com/GfQaKbR8SVyYPKwHRX8D)

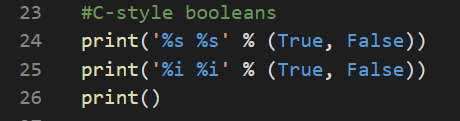
**Task 3**

The next step is to construct a string with the variables using the string format method. Represent theboolean value as 0 for false and 1 for true.

1. On line 20, Create a stringsequal to the string with the formatting options {0}. The variable at 0 in the format list, will be put into this placeholder. Then there is a : and then the open{1 so the next index of our string replacement: and then thetype.
2. Define a formatting option d, a numeric value for an unsigned numeric value. Then a \t for a tab character. In braces, insert 2, \t,3 and braces \t and 4 in braces. Enumerate our variables from 0 to 4, there are five variables, but the boolean value needs to be in twice. And then afterdot format I have d, d, a, b, c.
3. The first d will be the string value the false. And then the second d will be the boolean value represented as 0 or 1. And then the remaining variables a, b, and c, and then I print s on line 21.

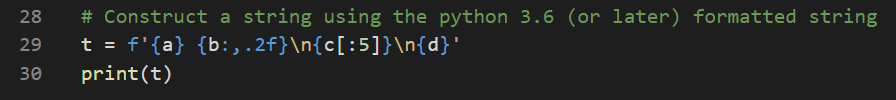
[](https://cdn.filestackcontent.com/nADgP34tR4qtlHlMZlGS)

1. If our boolean is represented 0 and 1 using the C-style booleans, use %s to print the strings false or true. And we use %i or %d, which would also be valid to print true or false as 1 or 0.
2. There is a string inside of the print statement, then %s space %s. And then the interpolation operator, the modulo symbol, and then thetuple with True and False. Print out True and False and then the second print with %iand %i. The modulo operator and True and False, it'll print 1 and 0. And then put an empty print statement on line 26 to put some space and empty line between this step and the next step.

[](https://cdn.filestackcontent.com/7CuRgeYQpitZovZdMtyf)

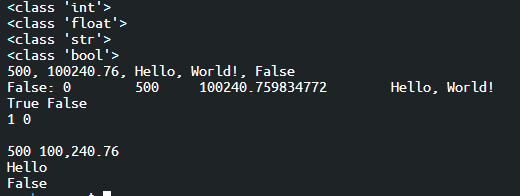
**Task 4**

1. The final step is to Construct a stringusing the python 3.6 or later formatted string. This method was introduced in Python 3.6. So on line 29, I have t = f. The characterffollowed by astring. This string is like using the dot format on a string like we did on line 20. But we don't have to use that method and we can put our variables right inside the string.
2. There is f, a single quote, or it could be a double quoted string. And then I have the braces in closing the variable I'd like to substitute, in this case a. Then a space, and then the braces in closing b, followed by a : and then the formatting. We have2f which will give a 1,000 separated number, separated with commas so at every thousand separator. And then everything after the decimal place will be truncated or rounded to two decimal places with the decimal 2f. And then we have a\ n for new line character. Inside the next format inside of the braces, we have cwith square brackets :5. So this will take the first five characters from the stringc, and then\n and our boolean value d, and thenprint the variable to finish the exercise.

[](https://cdn.filestackcontent.com/A7J5WlNcRWqW3pUoqG43)

**Terminal**

1. Now run the code, and we'll get the results in the console. Scroll up to the top of the output, there you'll see class int, class float, class str for string and class bool for boolean. This corresponds to the first step in the exercise to create variables of those particular types. And then we used C-style printf formatting to construct a string. Print 500 the integer, the floating point number to two decimal places. Instead of 759834772, we get it shortened to 0.76 for the decimal and then there is the string Hello, World!, and the boolean False printed.
2. The next step was to construct using the string format methodand print the results. So we have the variable False, and it's represented as an integer, a 0 for False or 1 for True, in this case, False corresponds to 0, followed by a tab character, the integer of 500. So the variable a, variable b, so the 100,240 and all of its decimal places, followed by another tab character and the string Hello, World!, to finish it off. And we saw the example with the C-style booleans with True and False using the string formatting and 1 and 0 using the percent %iformatting for the integers. We saw the example with the C-style booleans with True and False using the string formatting and 1 and 0 using the percent%i formatting for the integers.
3. Finally, constructing the string with the Python formatted string. We have the 500, the 100,240.76 in this case because we're going to two decimal places and we have a comma at the thousand separator as I specified in the formatting. And then the first five characters of the string is just the word Hello, and finally, theboolean False. And that concludes this exercise on formatting simple data types.

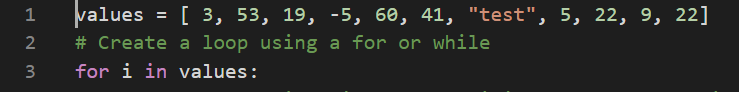
[](https://cdn.filestackcontent.com/ONl3R7VR96mI4Ei0fpDQ)

## Solution: Flow Controls and Conditionals

## Task 1

##### For Loop Solution

1. Start out by declaring the list of values that you are going to use inside of your loop.on line 1, values = [ 3, 53, 19, -5, 60, 41, "test", 5, 22, 9, 22] So the first step of the exercise is to create a loop using a for or while loop. use afor loop in this case to loop over the values in the list. On line 3, we have for i in values:and then the body of the loop is indented from there.

[](https://cdn.filestackcontent.com/tgHaLRThSWTbfmQxQTOA)

##### While Loop Solution

1. For the sake of completeness, we should include a while loop example in this exercise as well. on line 15, declare a variable total = 0 loop = 0. the while loop goes while total is less < 200.

[Graphical user interface, text, application, chat or text message

Description automatically generated](https://cdn.filestackcontent.com/aELIwNkLRGuxyj3Rjv0J)

## Task 2, 3, 4

### For Loop Solution

1. The next step is to use one or more if-elif-else conditional statements within the loop. on line 5 we have if i = 9: , we break, so if it sees a 9 in the list, it will break out of the loop. elif isinstance(i, str):

[Graphical user interface, text, application

Description automatically generated](https://cdn.filestackcontent.com/zU5oPCt1STmsl4hBIvWD)

1. What this function does is check the type of i, if it's an instance of a string it executes to continue. we've incorporated abreak and it continues the statement as required by the exercise, and then the else print(i).

[Text

Description automatically generated](https://cdn.filestackcontent.com/70S1SixoS0mp1uaTSabO)

### While Loop Solution

1. We have to be careful in these instances because if the total never reaches 200, we can end up with an infinite loop. So somewhere in this while loop, we should have a break statement just in case.
2. On line 18 in the body of the while loop we have anif isinstance(values[loop], str):. we're looking at the values. look at the type, if it's a string type, increment the loop. This counter we have that increments inside of this while loop, and then a continue statement.

[Graphical user interface, text

Description automatically generated](https://cdn.filestackcontent.com/z9XsaTjZTviUr5atSqvS)

1. If the Elif values at index loop are > 0, increment the total. If the value is less than 0, the current value at index loop is less than 0 it won't add it to the sum. Because in that case it would decrease it, and we don't want that to happen. On line 22, we havetotal += values[loop],

else:  
print(f'skipping {values[loop]}')

In this case, it'll skip any value that's zero or less, and we'll get something printed here.

[Text

Description automatically generated](https://cdn.filestackcontent.com/qvokZIhERvuHA3ihja1i)

1. Then on line 26, we have a new conditional if loop if loop == len(values)-1:. What this means is if the loop reaches the last index in the values list, it breaks out of the loop. Otherwise, it increments loop by 1, and continues looping until the total reaches 200. And then outside of the loop, weprint the sum, whatever the total is, and the loop value, the loop counter variable loop.

[Text

Description automatically generated](https://cdn.filestackcontent.com/S3Bnin7URZuSTU5fiHzb)

## Solution: Copy Containers

## Task 1

1. For this exercise, we import copy from the Python standard library. This gives us access to the copy and deep copy methods that we can use to copy our list of lists and create a list of lists to illustrate the copy and deep copy.
2. On line 4, we create our list of lists a = [[1,1,1,1], [2,2,2,2], [3,3,3,3]]

[Text

Description automatically generated](https://cdn.filestackcontent.com/waMCXVMR7eyiDeSBC6Aw)

1. We have a list of lists with three internal lists.

## Task 2

1. On line 7, there is a variable called shallow\_a. Shallow\_a = copy.copy(a) using the Python standard library. This will copy the list to the variable shallow\_a, but it's only a shallow copy, which means the elements in the lists will be the same as in shallow\_a.

[](https://cdn.filestackcontent.com/sNe3ZWWTb2YTciY8wh9x)

## Task 3

1. Create a deep copy of the list on line 9 with the variable we called deep\_a = copy.deepcopy(a) will copy the internal lists and make new copies. the idea being that if we modify an element of the shallow copy, it will be modifying the list internal to a and shallow\_a at the same time. But since the deep copy copies the internal elements as well, if we make a modification, it doesn't affect a or shallow\_a

[](https://cdn.filestackcontent.com/DotlV4YJSuOPkPkC1wyV)

## Task 4

1. On line 12, we have shallow\_a at list 1 at index 0. that's the internal list of 1. we're enumerating from 0, so this is the list of 2's and index 0 inside of the 2. It's the first element of the second list. And we set that value to 0. we're setting it in shallow\_a and since shallow\_a is a shallow copy(a), it should modify both.

[Graphical user interface

Description automatically generated](https://cdn.filestackcontent.com/h98u7RISsWXZyCmQFEB7)

## Task 5

1. If I modify the same element inside ofdeep\_a, as I do on line 15, so thatdeep\_a[1][0] = 99. It's modifying that first element of the second internal list.

[A picture containing graphical user interface

Description automatically generated](https://cdn.filestackcontent.com/4NW8eiTQ4CNAp9CMBaws)

## Task 6

1. then the final step is to print the contents of the resulting lists. on line 18, print(f'original: {a}'). This is going to print the list a. Then on line 19, print(f'shallow: {shallow\_a}'), and then print(f'deep: {deep\_a}'). we'll see the contents of them and see how these modifications affected these three lists. Then there is an empty print statement on line 21 to put some space between these results, the contents and printing the IDs

[Text

Description automatically generated](https://cdn.filestackcontent.com/oSOVubQETkKepKcIUDzg)

1. This is another way of highlighting what's happening internally in the list of lists. the idea being the shallow copy at a[1] index one, the idea of it should be the same between a and shallow\_a. But since the deep copy copies the internal elements as well, the idea of deep\_a at index 1 should be different.

[Text

Description automatically generated](https://cdn.filestackcontent.com/swKWvrQaSraJ4r7BIzqi)

## Terminal

1. Now run the code, and we can see the results in the debug console. the original result has our list of lists with the 1s untouched. But in the second list, the list of 2s the first element was set to 0. But it was set to 0 in the shallow copy. But that also reflected in the a because the internal list is referencing the same list in a shallow copy. And then the list of 3s remain untouched.

[](https://cdn.filestackcontent.com/cTRnfshpSPiazJ0RzfiT)

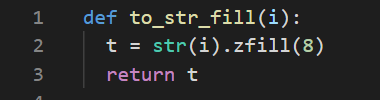
1. Now in the deep copy, set the first element of the second list to 99, and since this is the deep copy, it copies the internal elements so this does not affect the shallow copy nor the original.
2. And then the IDs, the difference is that the ID of the internal list of our deep copy is different from the internal lists of the shallow copy and the original a. the shallow copy keeps the same internal values, the same internal objects. it's referencing the same lists, but the deep copy creates a new one, so it gets a new ID and a new location in memory. And that concludes this exercise on copying containers.

[Text

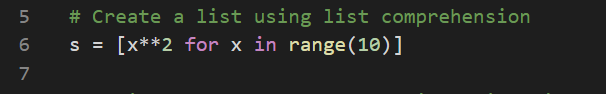
Description automatically generated](https://cdn.filestackcontent.com/ix632eVTTdeayaeNHjtF)

## Task 1

To begin this exercise, we define a function that we need later on in our list comprehension. On line 1, we have def to\_str\_fill(i):, and it takes a single parameter i. Set this to string fill or to \_str\_fill, it's going to take an integer, an int, convert it to a string and zero fill it out to a certain length. So in line 2, t = str(i).zfill(8). If the number is less than 8 characters long, it will add zeros to the left side of the number and then returns the result t. So, return t.

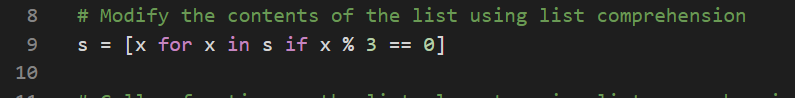
[](https://cdn.filestackcontent.com/gin3vueZTVewMj1FRjBK)

1. Now the first step of the exercise was to create a list using list comprehension. This for loop will take 10 integers. the range 10 means from zero to nine inclusive. And then that value the x for each of those, it will do xto the power of 2 and then returns it as a list because it's surrounded in square brackets. that's our basic list comprehension and we're returning a result which will be a new list. swill contain those values.

[](https://cdn.filestackcontent.com/e3VVu8A2RweRhmBhgp0U)

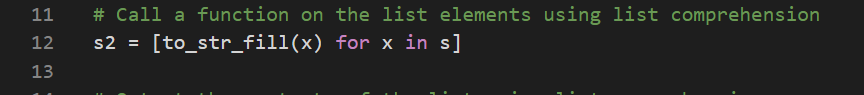
## Task 2

1. If we want to modify the values, we can't modify them in place using list comprehension. What we have to do is create a new list from an existing one and overwrite the existing one to modify it.
2. On line 9, I haves = then I start my list comprehension with the square bracket. And then I havex for x in s and I want to demonstrate something else we can do is have conditionals. It's going to return anx to be just returned as is if it's divisible by 3. Take all the values divisible by 3. ifx % 3 == 0, the remainder of x divided by 3 is equal to 0, is divisible by 3 and we return that value.s will remove all of the values that are not divisible by 3.

[](https://cdn.filestackcontent.com/HODd2cQbTnOoaxOZVUQy)

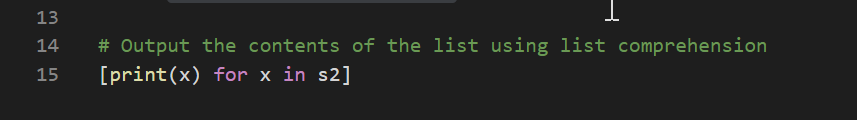
## Task 3

1. Then the next step is to call a function on the list elements using list comprehension, which is why the function is defined at the beginning the to \_str\_fillor to string fill.
2. On line 12, we have s2, create a copy of the list, this will be a new list, equals in square brackets. the list comprehension with to\_str\_fill(x) for x in s, so each of these that are divisible by 3, we're going to left fill them, we're going to string fill them with zeros. s2will be a list of strings.

[](https://cdn.filestackcontent.com/MLUltHRSLHjX0d5vA4pg)

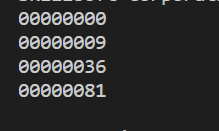
## Task 4

1. Now we're going to output the contents of the list using list comprehension. the resulting list is s2.
2. On line 15, just start the statement with a square bracket. we're surrounding in square brackets, and we're going to apply the print function on x for x in s2. this will print the values in the resulting s2list.

[](https://cdn.filestackcontent.com/WL1zomRTSeRa6aqPxL9e)

## Terminal

1.Now run the code. And in the results, we get four values that are divisible by three, the 0, the 9, 36, and 81. So for 0, it pads out all the zeros, for 9 it pads out seven 0s to the left, for 36 there's six 0s on the left, and the same for 81. everything lines up for each number to have eight characters. And that's what we expected based on my exercise code. And that concludes this exercise on list comprehension.

[](https://cdn.filestackcontent.com/UrNt0LHwTwmFD1kqFzcU)

## Solution: List Comprehension

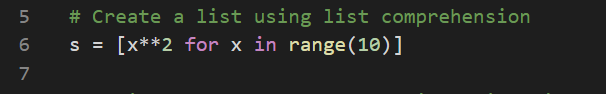
## Task 1

To begin this exercise, we define a function that we need later on in our list comprehension. On line 1, we have def to\_str\_fill(i):, and it takes a single parameter i. Set this to string fill or to \_str\_fill, it's going to take an integer, an int, convert it to a string and zero fill it out to a certain length. So in line 2, t = str(i).zfill(8). If the number is less than 8 characters long, it will add zeros to the left side of the number and then returns the result t. So, return t.

[Text

Description automatically generated](https://cdn.filestackcontent.com/gin3vueZTVewMj1FRjBK)

1. Now the first step of the exercise was to create a list using list comprehension. This for loop will take 10 integers. the range 10 means from zero to nine inclusive. And then that value the x for each of those, it will do xto the power of 2 and then returns it as a list because it's surrounded in square brackets. that's our basic list comprehension and we're returning a result which will be a new list. swill contain those values.

[](https://cdn.filestackcontent.com/e3VVu8A2RweRhmBhgp0U)

## Task 2

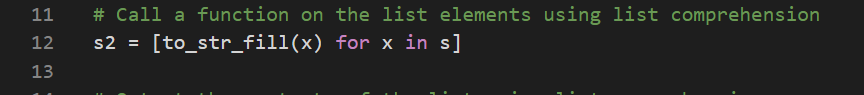
1. If we want to modify the values, we can't modify them in place using list comprehension. What we have to do is create a new list from an existing one and overwrite the existing one to modify it.
2. On line 9, I haves = then I start my list comprehension with the square bracket. And then I havex for x in s and I want to demonstrate something else we can do is have conditionals. It's going to return anx to be just returned as is if it's divisible by 3. Take all the values divisible by 3. ifx % 3 == 0, the remainder of x divided by 3 is equal to 0, is divisible by 3 and we return that value.s will remove all of the values that are not divisible by 3.

[Graphical user interface, application

Description automatically generated](https://cdn.filestackcontent.com/HODd2cQbTnOoaxOZVUQy)

## Task 3

1. Then the next step is to call a function on the list elements using list comprehension, which is why the function is defined at the beginning the to \_str\_fillor to string fill.
2. On line 12, we have s2, create a copy of the list, this will be a new list, equals in square brackets. the list comprehension with to\_str\_fill(x) for x in s, so each of these that are divisible by 3, we're going to left fill them, we're going to string fill them with zeros. s2will be a list of strings.

[](https://cdn.filestackcontent.com/MLUltHRSLHjX0d5vA4pg)

## Task 4

1. Now we're going to output the contents of the list using list comprehension. the resulting list is s2.
2. On line 15, just start the statement with a square bracket. we're surrounding in square brackets, and we're going to apply the print function on x for x in s2. this will print the values in the resulting s2list.

[Text

Description automatically generated](https://cdn.filestackcontent.com/WL1zomRTSeRa6aqPxL9e)

## Terminal

1.Now run the code. And in the results, we get four values that are divisible by three, the 0, the 9, 36, and 81. So for 0, it pads out all the zeros, for 9 it pads out seven 0s to the left, for 36 there's six 0s on the left, and the same for 81. everything lines up for each number to have eight characters. And that's what we expected based on my exercise code. And that concludes this exercise on list comprehension.

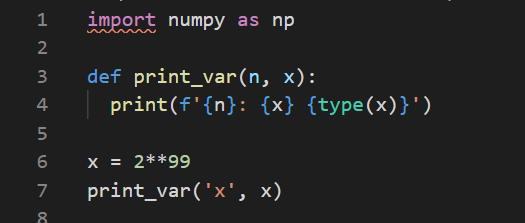
[Graphical user interface, text, application, chat or text message

Description automatically generated](https://cdn.filestackcontent.com/UrNt0LHwTwmFD1kqFzcU)

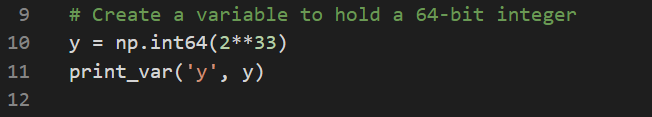
## Solution: Type Conversion

## Task 1

1. for this exercise, we need to use Numpy so that we can specify the number of bits we like to use for our numbers. On line 1, import numpy as np.
2. In Python, numeric values aren't restricted to the particular bit alignments, 32-bit or 64-bit integers. Integers expand as much as necessary to fit in the available memory. we can't be explicit about the size of our floating point numbers or our integers.
3. And then on line 3, define a function called print\_var that is two parameters and an x. n is the name of the variable and x is the value. print the f string, the formatted string with the name of the variable followed by a colon, the contents of the variable, and then the type of variable. this will make it easier to output the variables and their contents, and to see what we're working with for each step in the exercise.
4. On line 6, create a Python integer. we have x equals 2 to the power of 99. And then when you check the type using print var, printvar of the string xso we can output the variable name comma x, so we can also put the contents of the variable. we can see that the integer x can have a value greater than 64-bits, and Python handles it just fine.

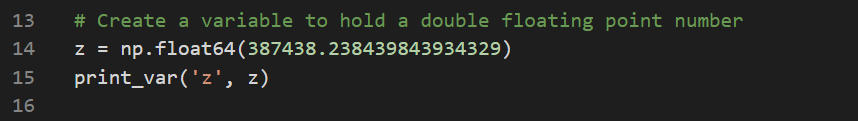
[](https://cdn.filestackcontent.com/DSJaWxDVQum6NID7zscA)

1. Now the actual first step in the exercise is to create a variable to hold a 64-bit integer. And we can define that using NumPy. So in line 10,y = np.int64, then give it a value, so 2 to the power of 33. this is a numeric value that requires an integer to store it that's greater than 32-bits. And then on line 11, print var of y.

[](https://cdn.filestackcontent.com/IDk44JyeTaOvRTE0lMVA)

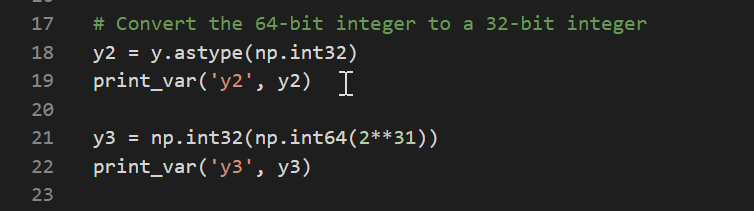
## Task 2

1. Create a variable to hold a double floating point number Or a float 64. On line 14, we have z = np.float64 of this number 387,438.2384 and along decimal number that's 15 decimal places long. And then print var of the variable z.

[](https://cdn.filestackcontent.com/QFTsLa4FSquBO9CSFjaB)

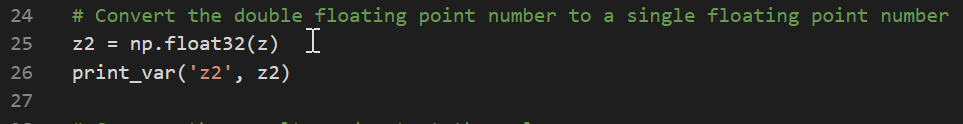
## Task 3

1. Now convert the 64-bit integer, the 2 to the power of 33, the integer yto a 32-bit integer. call this oney2, and set that equal to y.astype(np.int32). astypeis one way of converting from one type to another using NumPy. And then print var of y2.
   1. Then on on line 21, use a different method. take y3 = np.int32, and then as an argument give it np.int64 of 2 to the power of 31. we'll see what happens when we take a 64-bit integer. That's a large number, 2 to the power of 31, and we're converting it into a signed 32-bit integer. the size of the value is important, because it does fit into a 32-bit integer. Whereas the variable y, that's 2 to the power of 33 does not fit. we'll see how it behaves and we'll see the contents ofy3 when we do a print var of y3.

[](https://cdn.filestackcontent.com/UuqQLOpQqyhjgk7YnWAp)

## Task 4

1. Convert the double floating point number to a single precision floating point number. So we have z2 = np.float32(z), and then Iprint var of z2

[](https://cdn.filestackcontent.com/H9RVWmZ7QD6FokTIfCAz)

## Task 5

1. And the last step was to compare the results and output the values. To do this, run the code, and have a look at the results in the debug console. In the debug console, we have the first variable x, that's a Python end with this very large number which was 2 to the power of 99, as we did on line 6. And it gives class "int" as it's type. And this y is a number that's greater than a billion, and it's a class of "numpy, int64". z is the number we defined as 387,000 with the long decimal.

## Solution: Functions and Variable Scope

## Task 1

1. The first step in this exercise is to create a global variable. On line 2, we have g\_x = 2.5. We have this g\_ prefix, we know that we're using a global variable. it's just a naming convention to use for this exercise.

[Text

Description automatically generated](https://cdn.filestackcontent.com/45iXkiILQkCafMbLmlZA)

## Task 2

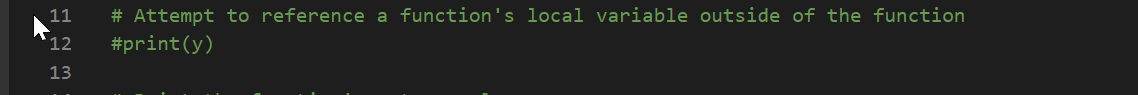
1. Then the next step is to create a function that uses the global variable and a local one. the function is defined in line 5, we have def f1, and it takes a single parameter x.
2. In the body of the function on line 6, we have a global variable, specify that we're using theg\_x the global variable and then y = 6. the local variable is y which is set to 6. And then on line 8, we haveg\_x = we set the global variable to be x \* y, so x times 6, and then we return g\_x.

[Text

Description automatically generated](https://cdn.filestackcontent.com/dGdGK7MSRXKeZcFW3VXL)

## Task 3

1. The next step is to attempt to reference a function's local variable outside of the function. the variabley, if we try to access it, say print it outside of the function, we're going to get an error. the editor here Visual Studio Code picks up that y as a problem.
2. Visual Studio Code tells me thatyis an undefined variable because it's not in the right scope. leave that line commented out because that attempt to reference that local variable failed outside of the function.

[](https://cdn.filestackcontent.com/ANcNbUPwSTi2H7PjQ3M2)

## Task 4

1. The final step is to print the function's return value. call the function twice. First call f1with 5 and this is embedded in a print statement. On line 15, we have print with the formatted string, so the f string, so f, and in the string we have f(5):, so the literal string, followed by{f1}(5) the actual function call, which will display the results.
2. Then do the same with the variable {g\_x}. And then do another call to the function passing g\_x as a parameter. make that function call and output the results and output the final contents ofg\_x.

[Text

Description automatically generated](https://cdn.filestackcontent.com/DGz3tfA5SLeFIuNQhEv8)

## Terminal

1. Now run the code, and we can see the result of the exercise. call f1(5), so that will pass 5 and then it will take y=6, and then 5 times 6 is 30, and then returns 30. the return value of that function is the same as the global variable, so the values should match. when you outputf1(5), it is 30, andg\_xat that point is also 30.
2. And if I do the same thing where I callf1(g\_x), which at this point in the code is the value 30, so 30 times 6 is 180. So it setsg\_x to 180, return set from the function, displays the results, and we verify that the contents of the global variable is also 180. And that's what we get in the output. f1(g\_x) at that point is 180 and g\_x is 180. And that concludes this exercise on functions and variable scope.

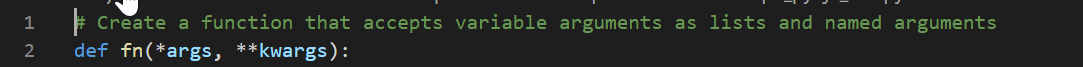
[Graphical user interface, text, application, chat or text message

Description automatically generated](https://cdn.filestackcontent.com/SDDxastT42nmsxHVdm5E)

## Solution: Function Arguments and Return Values

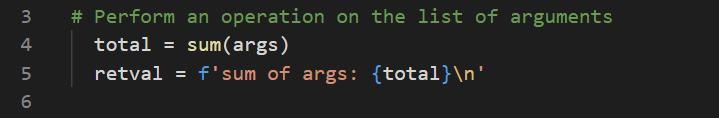
## Task 1

1. The first step in this exercise is to create a function that accepts variable arguments as lists and named arguments. So on line 2, define a function def fn, the function is called fnand it has two parameters, \* args, \*\* kwargs. Args are list of variables passed to the function and kwargs are keyword arguments or dictionary arguments passed to the function.

[](https://cdn.filestackcontent.com/GOQfXZxGQjGy6oy2pzcG)

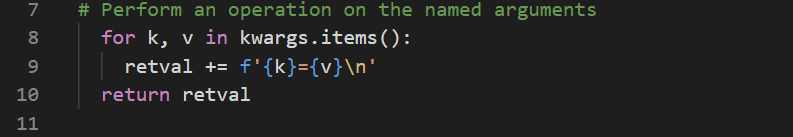
## Task 2

1. The next step is to perform an operation on the list of arguments. perform the operation on args is the sum function. On line 4, we havetotal = sum(args). Now this supposes that the args are numeric. If other types are passed like string types, then this will cause an error, but we'll deal with that shortly.
2. Then on line 5, set retval, which is going to be a string equal to the formatted string or f'sum of args: {total}\n'. So a newline character.

[](https://cdn.filestackcontent.com/HHj3TiW0RGWYuv7rR8mP)

## Task 3

1. Then we need to perform an operation on the named arguments. take the named arguments, loop over all of the named arguments that are passed into this function and append them to our retval.
2. On line 8 for k, v in kwargs.items(): set the retval, so I do += to append thef string of k = v. And then put another \nor another newline character and the end of this function, it returns a retval, which will return different return values based on their arguments. And that was the requirement for the last step in the exercise

[](https://cdn.filestackcontent.com/jYd2EN7jQDi3ZwVRQvST)

## Task 4

1. On line 13, call the function and set the result to the variable s. s = fn(1,2,3, test=5, p='Hello, World!'), we'll get past this keyword arguments are kwargs, and 1, 2 and 3 will get passed in the arg list. print (s) should print the retval string returned by our function fn.
2. Then call the function again, on line 16, t = fn(4, 5, 6, and then the string,'hi'. the first four parameters will get passed asargs. the list of arguments, then greeting = 'Hello' and recipient = 'World' and then print(t). But since we passed the string hi, which gets passes args, which gets passed into sum function, this is going to be a problem here. if I run the code as is, we get a problem, we get,unsupported operand type(s) for +, 'int'and 'str'.
3. What we need to do is just remove, hi, Resave it and run it again. In this case, get the results. get the string, sum of arg: 6, a newline character test =5 and then another new line, and thenp = Hello World. the first three elements get printed in the retval. it's returned in the variable sand printed and the same thing for the variablet. 4+5+6 is 15, so the sum of theargs is 15, greeting = Hello and recipient = World. And that concludes this exercise functional arguments and return values.

[Text

Description automatically generated](https://cdn.filestackcontent.com/rSQNnSLVThiyGjZHN6FO)

## Solution: Function Recursion and Closures

## Task 1

1. The first step in this exercise is to create a Python function. define a function def fn\_next, and it takes a single argumentbegin. And then inside the body of this function, we havecurrent = begin. current is going to be a variable that lives within our inner function, within our closure, and it's going to be a state variable. we'll need to extend its scope so we can access it from the inner function.

[Text

Description automatically generated](https://cdn.filestackcontent.com/jpIOP1ThTdBVGXeNrji9)

## Task 2

1. The next step is to create an inner function that acts as a closure. On line 5, we have def fn\_inc with no parameters. this function is defined within the body of the outer function and it declares a non local current. non locals and important keywords, we can extend current to be accessible from within the inner function.
2. Then we increment current by 1, so current +=1, and we return current. Every time this function is called, it's going to incrementcurrentby 1 and return the value. And then on line 9, return fn\_inc. Notice there are no parentheses because we are returning a reference to this function.

[Text

Description automatically generated](https://cdn.filestackcontent.com/fu3uFWRyTnyI8H7lNFt0)

## Task 3

1. Then on line 11, we need to return the closure function object, do that by calling fn\_next. On line 12, get\_next this is going to be a method that points to the particular instance of fn\_inc, where we have set the value of current. f get\_next = fn\_next starting at 100. it's going to start at 100, and each time you call get\_next, it's going to return the next integer. call the function to return a value that changes each time it is called.

[Text

Description automatically generated](https://cdn.filestackcontent.com/5YFJFfobQUCfF10p5pXM)

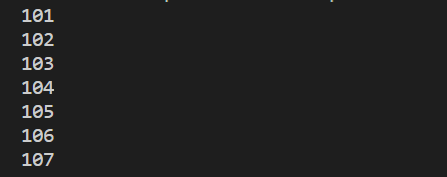
## Task 4

1. On line 15, print(get\_next()). this is called 15, 16, 17, all the way up to line 21. call it seven times. Now, since the starting value is 100, and we increment before we return it, the actual values are going to start at 101.

[Text

Description automatically generated](https://cdn.filestackcontent.com/1Uc3QsV8TdyBfInFxZpb)

1. Now we'll run the code, which will run the programme and output these results to the debug console. And in this case, we get what from 101 to 107 incrementing current as we go. this closure provides a way of data hiding on our current value and we use it as a callback function. it helps us reduce the scope of our variables. current doesn't need to be a global variable, it can be defined as a non local variable and use within the scope of this function. And that concludes this exercise on recursion enclosures.

[](https://cdn.filestackcontent.com/D9D3r2YyTKq649ECmufJ)