# Week 8 - Functions

Over the past weeks, the program we build has grown to almost or over one hundred lines of code. Let's move the detailed works into functions and make the main flow clearer.

Here are the functions we'll extract:

* initialize: read the file 'records.txt' or prompt for initial amount of money.
* add: prompt for some records and add them into the record list (or other data structure you're using).
* view: print the records.
* delete: prompt for a record to delete and delete it from the record list.
* save: write the records to the file 'records.txt'.

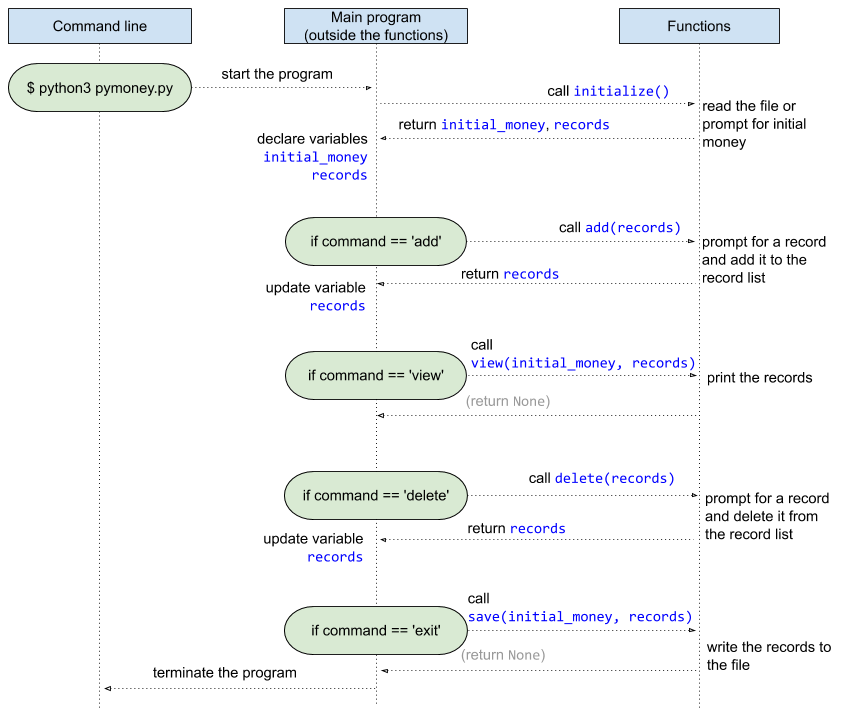
After defining the functions, the remaining part (not in functions) of your code should be like this:

| import sys  # The 5 function definitions here  initial\_money, records = initialize()  while True:  command = input('\nWhat do you want to do (add / view / delete / exit)? ')  if command == 'add':  records = add(records)  elif command == 'view':  view(initial\_money, records)  elif command == 'delete':  records = delete(records)  elif command == 'exit':  save(initial\_money, records)  break  else:  sys.stderr.write('Invalid command. Try again.\n') |
| --- |

Here we want to make the functions independent of other variables defined in the global scope. That is, none of the functions should access **initial\_money** or **records** directly. Instead, needed variables should be passed into the functions as parameters and reassigned as the returned value from the functions.

| **O** | **X** |
| --- | --- |
| def foo(L):  s = input()  L.append(s)  return L  L = ['abc', 'def']  L = foo(L) | def foo():  s = input()  L.append(s)  L = ['abc', 'def']  foo() |
| The function foo does not know there is a variable L outside. It only sees the passed-in parameter L and simply returns it after modification. | Yes, this works the same as the code at the left. But now foo is dependent on the outside variable L, which is not what we want here. |

This diagram illustrates the data flow between the main program and the functions:



## Required Steps

1. Identify the variables that are used and maintained throughout the main program. Generally you should find only **initial\_money** and **records** (maybe under other names). You may have other ones but make sure they are necessary. Eliminate unnecessary ones and keep those kind of variables as few as possible.
2. Define the 5 functions: **initialize**, **add**, **view**, **delete**, and **save**.
   1. Define the functions with proper formal parameters and move the code originally in **if-elif** into respective functions.
   2. Adjust the code inside the functions so that they're only using parameters and not accessing outside variables.
   3. Return values for the caller (i.e. main program) to update related variables.
3. Rearrange the remaining code as the example in purple border above.
   1. Call the functions and pass needed variables as parameters.
   2. Declare or update related variables after a function returns.

## Notes

* Please follow the example code and diagram as much as you can, especially the function signature (i.e. what parameters and what return value does a function have) and the data flow (i.e. when and where is a variable passed or returned).

## Related Knowledge

* Functions, parameters, and return values
* Scope of identifiers