Here is an example program using lists.

```
"""
Simple task list.
"""

def new(tasklist, task):
    """Add new task"""
    tasklist.append(task)

def remove_by_num(tasklist, tasknum):
    """Remove by number"""
    if tasknum > 0 and tasknum <= len(tasklist):
        tasklist.pop(tasknum - 1)

def remove_by_name(tasklist, taskname):
    """Remove by name"""
    if taskname in tasklist:
        tasklist.remove(taskname)

def printlist(tasklist):
    """Print task list"""
    print("=======================")
    num = 1
    for task in tasklist:
        print(num, task)
        num += 1
    print("=======================")

def run():
    """Manipulate task list"""
    tasks = []
    new(tasks, 'Teach Class')
    printlist(tasks)

    new(tasks, 'Buy some ties')
    new(tasks, 'Learn Python')
    printlist(tasks)

    new(tasks, 'Build new task list')
    printlist(tasks)

    remove_by_num(tasks, 1)
    printlist(tasks)
    remove_by_num(tasks, 2)
    printlist(tasks)

    remove_by_name(tasks, 'Buy some ties')
    printlist(tasks)

run()
```

This is a simplistic task list program to keep track of your tasks. The task list itself is simply a list of strings. There are functions to manipulate the list, including adding and removing tasks and printing the list. Let's break down its components.

First, there is a function to add a new task to the task list:

```
def new(tasklist, task):
    """Add new task"""
    tasklist.append(task)
```

This uses the list **`append`** method. This adds the new task to the end of the task list. This is the easiest way to add a new item to a list. You could also add it in any location (if you wanted to prioritize or sort the task list, for example), by using the **`insert`** method.

Next, there are two different functions to remove a task from the task list:

```
1  def remove_by_num(tasklist, tasknum):
2      """Remove by number"""
3      if tasknum > 0 and tasknum <= len(tasklist):
4          tasklist.pop(tasknum - 1)
5
6  def remove_by_name(tasklist, taskname):
7      """Remove by name"""
8      if taskname in tasklist:
9          tasklist.remove(taskname)
```

The first function, **remove_by_num**, takes a task number (starting from 1, as we are assuming the users are not computer scientists!) and removes that task from the task list. Notice that the function first makes sure that the given **tasknum** is in the bounds of the list. Given that we assume the items are numbered starting at 1, instead of 0, we need to make sure that **tasknum - 1** will be a valid index into the list so that the **pop** method will not cause an error. When given a number within the bounds of the list, **pop** removes the item at the given index in the list and shifts all later items down to fill the gap.

The second function, **remove_by_name**, takes the name of the task and removes. Again, notice that the function first makes sure that the task is actually in the list so that the **remove** method will not cause an error. If it is in the list, then it uses the **remove** method on the task list to remove it. This will find a matching item, remove it from the list, and shift all later items down to fill the gap.

Finally, there is a function to print out the task list:

```
1  def printlist(tasklist):
2      """Print task list"""
3      print("=======================")
4      num = 1
5      for task in tasklist:
6          print(num, task)
7          num += 1
8      print("=======================")
```

This function iterates over the task list and prints out each item in turn. It also keeps a running counter (starting at 1), called **num**, so that it can print out the task number in front of the task name. This function demonstrates the use of a for loop to iterate over a list. The body of the loop, lines 6 and 7 are executed once for each item in **tasklist**. During each the execution of the loop body, the variable **task** will take on the value of next item in the task list.

Finally, the **run** function uses the above functions to manipulate a task list. We encourage you to run this program, modify it, and make sure that you understand how it is operating on lists.