Cmpe 150 Lab 2: Functions

Last Week

- We learned about how computers work and how we can do basic operations like getting input, doing arithmetic, and giving some output.

- We can do several things with these tools, yet we ought to be careful.

Spaghetti Code

- We may need hundreds or even thousands of instructions/lines of code for complex tasks.
- It's super hard to understand and maintain such code. Comments won't be enough to solve the issue.

When you try to debug the client developer's spaghetti code...
quickmeme.com

Solution: More Structured Code with Functions

- A function is a unit of program carrying out a specific task.
- To define a function, we use a syntax like the following. (With indentation)
- def function_name():

```
    your_code_line_1

    your_code_line_2

    .

    .
```

How to Use/Call a Function

- After we define it as described in the previous slide, we just say function_name() in our code.

- When we do this, our computer jumps to the part where we define what we want our function to do. Executes instructions there, and then it returns to where it jumped from.

How to Use/Call a Function

```
def my_first_function():
    print("Hello this is my first function.")
    print("I am printing some output by using it.")

my_first_function()
```

"Why is my function not outputting anything"?
"Oh I never called the function"
ProgrammerHumor.io

Main Function

- As a convention, we define a function called main, corresponding to the whole program.

- If we write everything inside the main, are we done?

Main Function Convention

```
def main():
        line1
        line2

if __name__ == '__main__':
    main()
```

Inside a function, we can call another function.

```
def main():
    function1()
    function1()

def function1():
    print('I am running function 1')

main()
```

We Can Even Call Multiple Functions

```
def main():
    function1()
    function2()
def function1():
    print('I am running function 1')
def function2():
    print('I am running function 2')
main()
```

Idea

- The main function will correspond to the whole program. We will call different functions inside it, each corresponding to a subtask. We'll define what each of these functions will do as well.

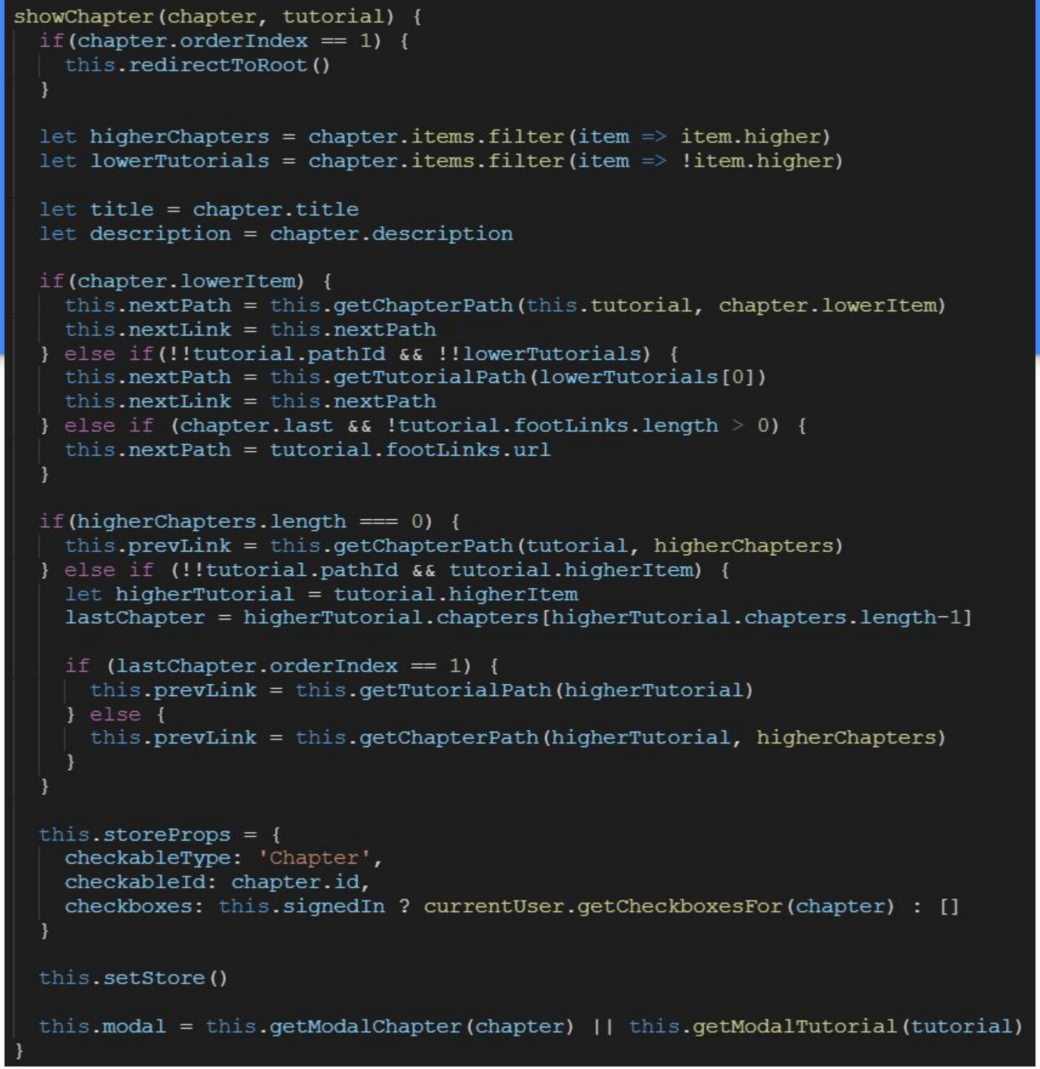
- Is it sufficient? Can we deal with 1000s of lines of code?

Idea (Cont.)

- Further, divide each new function into smaller components again until you reach a point where dividing a function does not make sense. For instance, calculating 2x+5 can be a single function—no need to divide.

- Also, it is better to stay within 5-6 lines of code in a function at most. Not as a restriction imposed by Python but as a good practice. So that everyone can understand your code more easily.

Bad Example



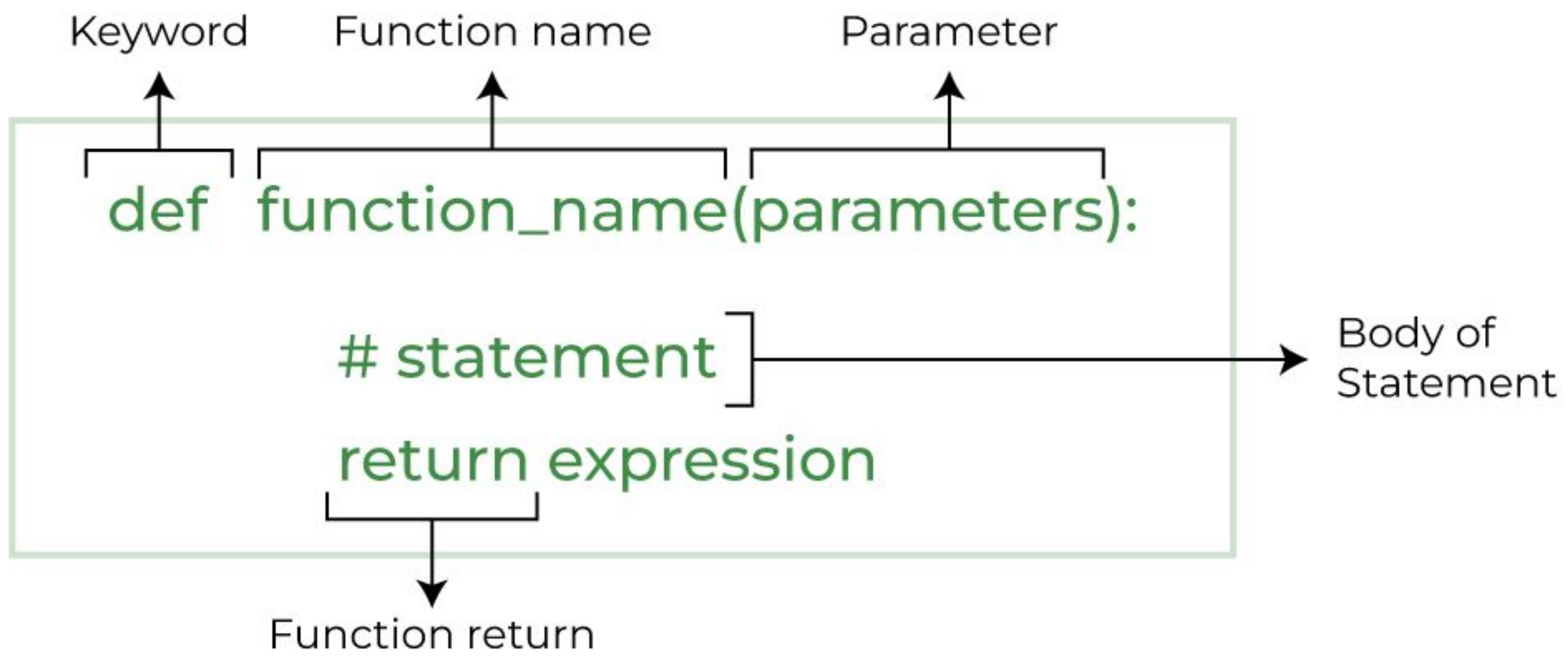
```
showChapter(chapter, tutorial) {
  if(chapter.orderIndex == 1) {
    this.redirectToRoot()
  }

  let higherChapters = chapter.items.filter(item => item.higher)
  let lowerTutorials = chapter.items.filter(item => !item.higher)

  let title = chapter.title
  let description = chapter.description

  if(chapter.lowerItem) {
    this.nextPath = this.getChapterPath(this.tutorial, chapter.lowerItem)
    this.nextLink = this.nextPath
  } else if(!!tutorial.pathId && !!lowerTutorials) {
    this.nextPath = this.getTutorialPath(lowerTutorials[0])
    this.nextLink = this.nextPath
  } else if (chapter.last && !tutorial.footLinks.length > 0) {
    this.nextPath = tutorial.footLinks.url
  }

  if(higherChapters.length === 0) {
    this.prevLink = this.getChapterPath(tutorial, higherChapters)
  } else if (!!tutorial.pathId && tutorial.higherItem) {
    let higherTutorial = tutorial.higherItem
    lastChapter = higherTutorial.chapters[higherTutorial.chapters.length-1]

    if (lastChapter.orderIndex == 1) {
      this.prevLink = this.getTutorialPath(higherTutorial)
    } else {
      this.prevLink = this.getChapterPath(higherTutorial, higherChapters)
    }
  }

  this.storeProps = {
    checkableType: 'Chapter',
    checkableId: chapter.id,
    checkboxes: this.signedIn ? currentUser.getCheckboxesFor(chapter) : []
  }

  this.setStore()

  this.modal = this.getModalChapter(chapter) || this.getModalTutorial(tutorial)
}
```

Parameters and Return

- A function had better be able to get some input and give output as well; otherwise, each function would be independent. Similar to functions in Math.

- Remember to give each parameter's value(argument) when calling a function.

General Syntax of Functions



Example

```
def add_numbers(num1, num2):
    sum = num1 + num2
    print("Sum: ", sum)
    return sum

result = add_numbers(5, 4)
```

Multiple Return Values

- Like being able to have multiple parameters, we can return multiple values.
  return x, y, z

- Remember to make the call accordingly.

- x, y, z = a_function_returning_three_values()

Benefits of Using Functions

- Structured code
- Abstraction
- Reusability of the same code again and again (Just change the arguments if needed)

Benefits of Using Functions (Cont.)

- Others can use your code as well.
- Or vice versa: Python libraries
- Easier to make changes and fix bugs

User Defined Functions vs Built-in Functions

- What we covered was user-defined functions.
- Python already provides some built-in functions like input() and print()

Information Regarding the Quiz

- Quizzes are done individually, so there is no information sharing between students. If you encounter difficulty while writing the code, raise your hand so that our student assistants will come and help you as much as possible.

- Everyone will use the lab computers. Using your PC is strictly FORBIDDEN.

Information Regarding the Quiz (Cont.)

- You will log into the Teaching Codes system using your username and password. Please be sure that you know your username and password. If you do not, please learn by checking your email as soon as possible.

- Try to log in using the lab computer to avoid a problem when you try it at the beginning of the quiz.

Thanks

Any questions?

References

1. http://www.quickmeme.com/p/3w1bz5/page/3
2. https://programmerhumor.io/wp-content/uploads/2023/10/programmerhumor-io-programming-memes-e315529c6afdef1.jpg
3. https://www.geeksforgeeks.org/python-functions/

Regarding the Quiz

- Everyone will use the lab computer. Also, no phone, PC, book, notebook, etc. on the desk.
- No communication with other people
- If you want to ask something, just raise your hand so that student assistants will come.

Regarding the Quiz (Cont.)

- If you are already logged in, please log out.
- Open up a new project using File -> New Project and be sure that you can run some arbitrary code like print('Deneme')
- Now login, please. It may take time since everyone tries it at the same time. Be patient.

Regarding the Quiz (Cont.)

- You should be able to see a text saying Quiz1…
- Click “Open” and notice that a new folder came into your directory. It can be seen among the files on the left.
- On the right, there’s a button saying description. Click to see the description.

Regarding the Quiz (Cont.)

- First, check some general warnings.
- Now let's read the question description. Notice that we already provided some code, so you won't write everything from scratch.
- Now let's look at the starter code on the Main.py file and pay attention to comments saying DO NOT EDIT. (2 of them)

Regarding the Quiz (Cont.)

- Let’s say that we wrote our code. We can run it as classical to test if it works.

- Also, we can use Tester.py by right-clicking and choosing “Run”. We’ll try your code with other test cases as well, so passing the ones we provided is not sufficient.

- If you think that you wrote it correctly and you passed the tests. You can click Submit. Make sure that it says the submission is successful. You can make multiple submissions.

Regarding the Quiz (Cont.)

- Please try to write your code and raise your hands if you have any questions.