6.00.1x Week 1 FAQ

Commonly Used Acronyms

PSETs Problem Sets

Fexes Finger Exercises

Community TAs/TAs Community Teaching Assistants

Course Logistics

- **Can I try the course before upgrading to a Verified Certificate?**
  Of course! You are welcome to try the course and all its materials before upgrading to a verified certificate but if you wish to receive a certificate you must upgrade before the date indicated in the "Home" tab.

- **Are finger exercises graded?**
  Yes, completing finger exercises counts towards your final grade but you have unlimited attempts to complete them. Finger exercises are here to help you solidify your knowledge and learn from your mistakes.

- **How many attempts am I allowed for finger exercises?**
  As many as you need. Finger exercises are here to help you solidify your knowledge, detect areas where you need to work on and learn the topics presented.

- **Can I spend more time than the estimated time on an exercise?**
  Absolutely! It's okay to spend more than the estimated time on a finger exercise, just make sure that you understand the topic and what the exercise required you to do so you are able to solve it more quickly the next time you are presented with a similar challenge. If you have any questions on the exercises please ask them in the discussion forums. Your classmates and Community TAs will be there to help you.

- **Are there deadlines I need to meet to complete lectures and exercises?**
  No, in this self-paced version of the course there are no deadlines for lectures and exercises but you must complete all assignments before the course ends. You can check this date on the "Home" tab.

- **How to submit code to the grader for exercises that ask me to assume that variables are already declared or to not ask for user input?**
  Don't include the variable declaration (or the line which gets the user input) when you paste your code to the grader, remove that line from your code as in the example 1 and 2:

Listing 1: Code you run on your IDE to test your solution

```
# your own test value
num = 5
# what you are asked to code
if num != 6:
print(num)
```

Listing 2: Code submitted to the grader

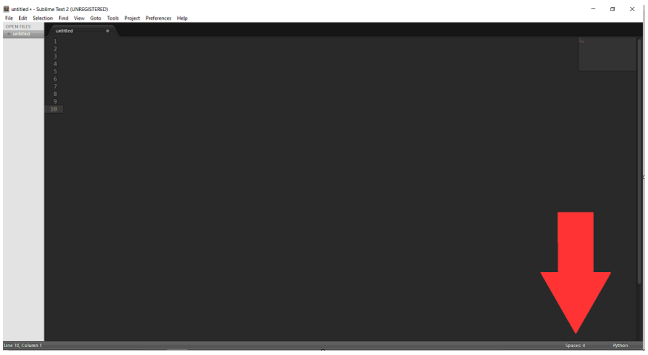
```
                    # what you are asked to code
                    if num != 6:
                    print(num)
```

- **How can I keep indentation consistent when I paste my code to the grader?**
  It usually helps to use tabs instead of spaces to create indentation. If you are using Sublime Text as your Text Editor, highlight your code to check if you see these dots to the left of your indented code (see Figure 1).

Figure 1: Indentation using spaces in Sublime Text

If you do, this means that you are using spaces instead of tabs for indentation. To change this, select your code and click on the option highlighted with the red arrow on the image below, select the option located at the bottom right corner of the screen that says “spaces” (see Figure 2).

Figure 2: The *spaces* option



Then click on “Convert Indentation to Tabs” (see Figure 3)

This will convert the spaces to tabs for the code you selected (see Figure 4). After this you can copy and paste your code to the grader to keep your indentation. Good luck and if you have any questions we are here to help you in the discussions forums!

Figure 3: Convert Indentation to Tabs

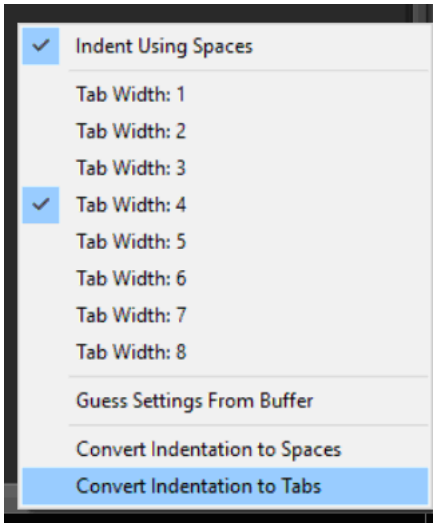


Figure 4: Indentation using tabs in Sublime Text

Course Content

- **What is Static Semantic vs. Semantic?**
  Static semantics checks which syntactically correct expressions have meaning (e.g 3×5 is syntactically correct and multiplying two numbers has meaning) while Semantics interprets the meaning of the expression. That meaning can't be ambiguous.

- **Why is `True` different from true and `False` different from false in Python?**
  Python is case sensitive; therefore the correct way to write Booleans is `True` and `False`. Using true and false will return an error because they are not recognized as Booleans.

- **What is Bool?**
  Bool is a way to refer to Booleans, the truth values `True` and `False`.

- **Difference between / and // in Python**
  The / operator returns a `float`, while // returns an `int`.

- **What does `!=` mean?**
  This is a comparison operator that means "not equal". It returns `True` if the values compared are NOT equal and returns `False` if the values compared are equal.

- **What is the syntax for advanced string slicing?**
  `"string"[start: end: step]`

- **Does `input()` always return a string?**
  Yes. The Built-in function `input()` casts ("converts") the input into a string and returns it. You can find more information on this function at this link.

- **What does the error "`'str' object is not callable`" mean?**
  This error occurs when you name one of your variables `str`. If you use this name, you will "replace" the built-in function `str()` and you will not be able to use it anymore as a built-in function (More information on `str` at this link).

- **Difference between `if` and `elif`**
  `if` is the first condition checked in an if/elif/else statement. If it is `True` the "if" code block will be executed. Else, if it is `False` the subsequent "elif" conditions will be checked. The code block of the first `elif` condition that evaluates to `True` will be executed. If none of them evaluates to `True`, the `else` code block will be automatically executed.

- **What is # and what is it used for?**
  In Python, # is used to comment out a line of code or to add a comment to your code. These lines will not be executed as code, they are simply a way to help you explain your thought process to yourself and to other developers. For comments you do not need to follow the syntax of the programming language, you can write them as if you were talking to yourself or explaining what your code does. It is a very helpful way to communicate with developers that will look at your code in the future to try to understand how it works.

- **What is the difference between an Error and an Infinite Loop?**
  An error is a warning thrown by your IDE or Shell, it is descriptive and gives you some information on what caused it. An infinite loop occurs when a while loop keeps executing indefinitely because there is no condition to stop its execution or if there is, that condition

is never met. Infinite loops may not throw errors at all and the code will keep executing indefinitely until you manually execute a command to stop its execution.

- **How do I stop an infinite loop?**
  By pressing `CTRL + C` on Windows.

- **Difference between While and For loop**
  A While loop is used for situations where we want to keep executing a code block while a condition is not met, meaning that we don't know ahead of time how many times the loop has to be executed to meet that condition. In contrast, a For Loop is used when we do know how many times the loop has to be executed.

- **What is `str`?**
  `str` is the built-in string class. You will learn about classes in the next few weeks of the course but right now you need to know that you can check if a variable is a string by comparing it directly to `str` (To learn more about str)

- **Difference between `for i in` and `for i in range`?**
  `for i in` is used to iterate over all the elements of a sequence (characters in a string, and elements of other data structures that you will learn in the next weeks of the course such as tuples and list). `for i in range` is a way to execute the loop a specific number of times that you can customize. Both update their variable (`i` in this case) on every loop iteration.

- **How does `while True` work?**
  `while True` will execute a While loop indefinitely until it reaches a break statement to stop its execution. This is useful when you need to perform the same task an indefinite number of times until a condition is met.