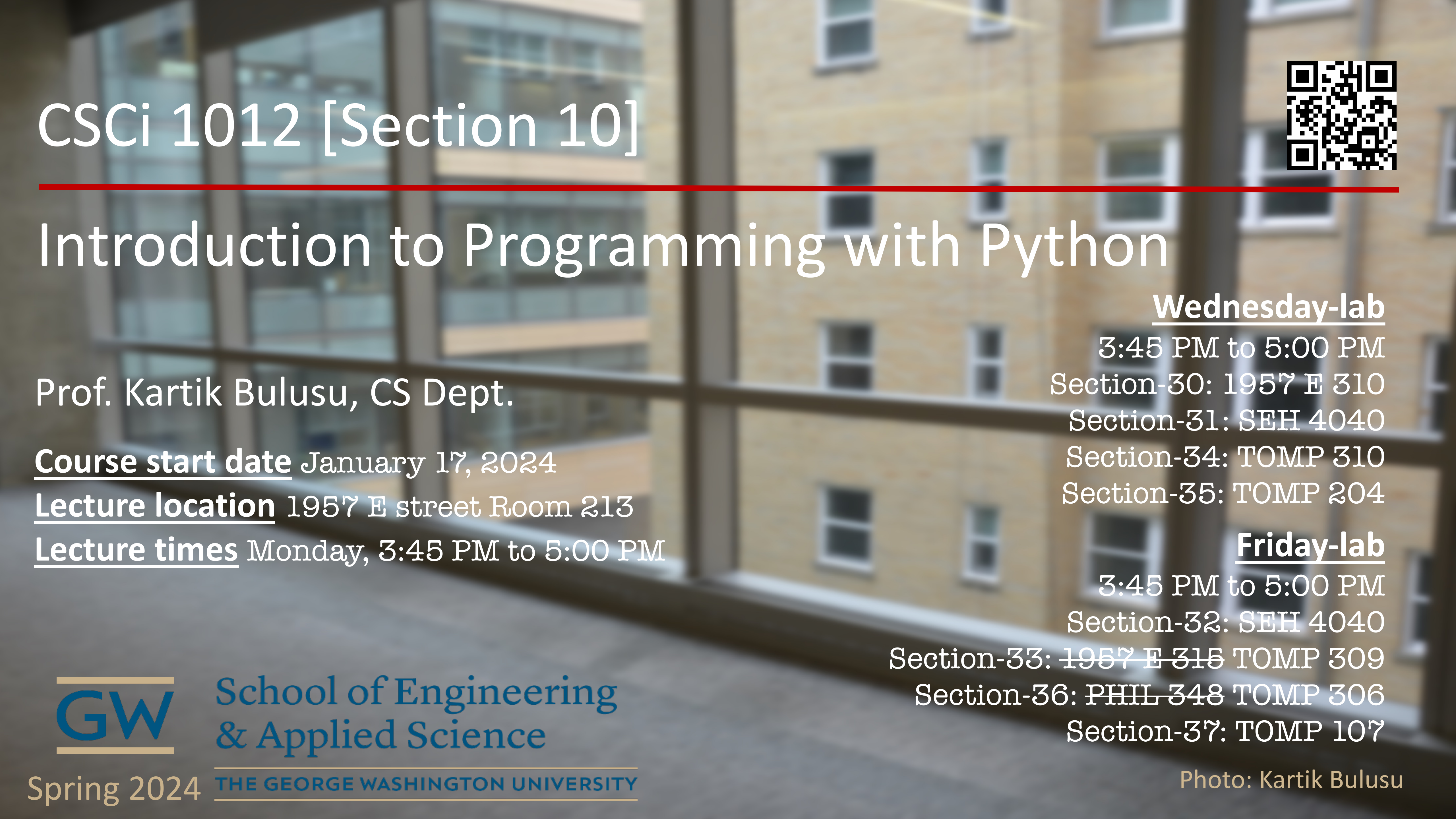


CSCi 1012 [Section 10]

Introduction to Programming with Python

Prof. Kartik Bulusu, CS Dept.

Course start date January 17, 2024
Lecture location 1957 E street Room 213
Lecture times Monday, 3:45 PM to 5:00 PM

Wednesday-lab
3:45 PM to 5:00 PM
Section-30: 1957 E 310
Section-31: SEH 4040
Section-34: TOMP 310
Section-35: TOMP 204

Friday-lab
3:45 PM to 5:00 PM
Section-32: SEH 4040
Section-33: ~~1957 E 315~~ TOMP 309
Section-36: ~~PHIL 348~~ TOMP 306
Section-37: TOMP 107

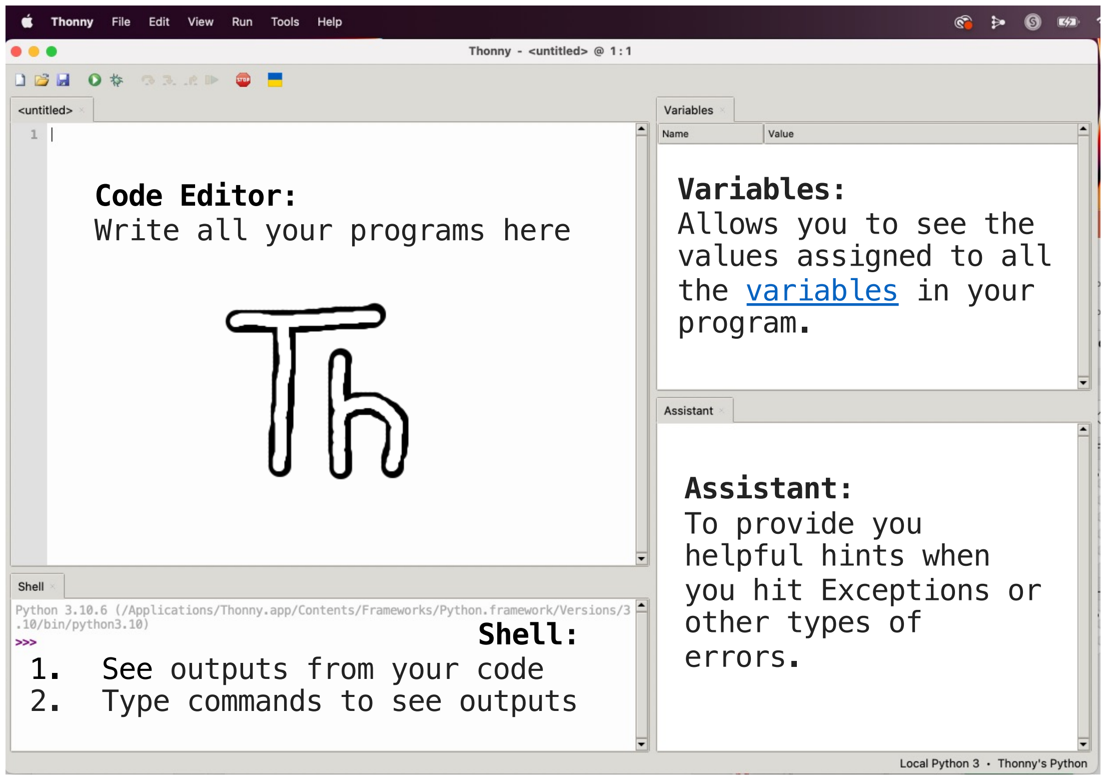
Spring 2024

School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Photo: Kartik Bulusu

Quick Recap: Thonny integrated development environment (IDE)

Sources:
Thonny (IDE): https://en.wikipedia.org/wiki/Thonny
Thonny: The Beginner-Friendly Python Editor: https://realpython.com/python-thonny/



Thonny File Edit View Run Tools Help
Thonny - <untitled> @ 1:1
<untitled>
Variables
Name
Value
Assistant
Shell
Local Python 3 • Thonny's Python

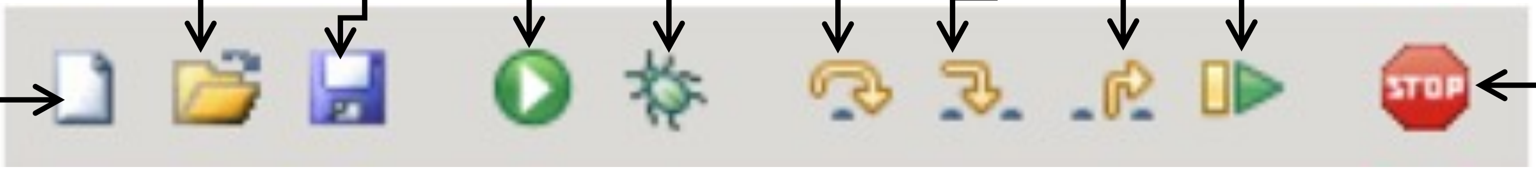
```
Python 3.10.6 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin/python3.10)
>>>
```

Code Editor:
Write all your programs here

Variables:
Allows you to see the values assigned to all the variables in your program.

Assistant:
To provide you helpful hints when you hit Exceptions or other types of errors.

Shell:
1. See outputs from your code
2. Type commands to see outputs



Allows you to run the code, i.e., "Do what I told you do do!".

Allows you to open a file that already exists on your computer

Create a new file

Save your code. Press this early and often.

Allows you to debug your code. A bug is another name for a problem.

The arrow icons allow you to run your programs step by step.
These icons are used after you press the bug icon

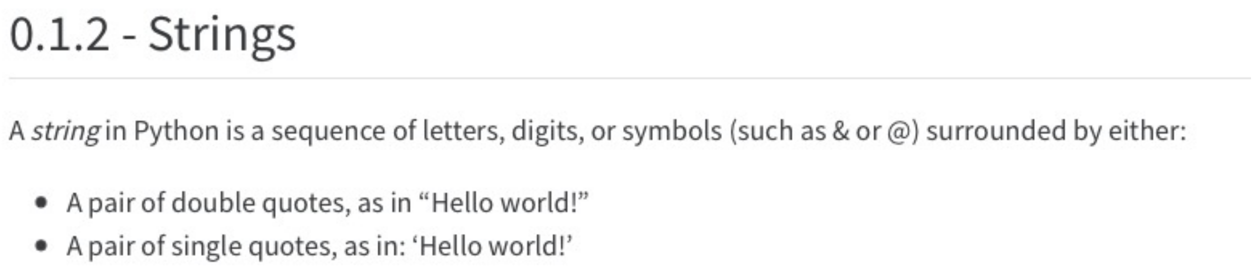
The resume icon allows you to return to play mode from debug mode.

The stop icon allows you to stop running your code

Built-in function `print()`

- By default, Python‘s `print()` function ends with a newline
- Commas between each entry outputs a space between each entry
- The entry of arguments of `print()` can be strings

More ways to use the `print()` function



0.1.2 - Strings

A *string* in Python is a sequence of letters, digits, or symbols (such as & or @) surrounded by either:

- A pair of double quotes, as in “Hello world!”
- A pair of single quotes, as in: ‘Hello world!’

An escape character is a backslash \ followed by the character you want to insert.

`\'`	Single Quote
`\\`	Backslash
`\n`	New Line
`\r`	Carriage Return
`\t`	Tab
`\b`	Backspace
`\f`	Form Feed
`\ooo`	Octal value
`\xhh`	Hex value

sep

```
print(“Hello”, “World!” ,”I”, “love”, “Python”, sep=“,”)
```

end

```
print(“Hello”, “World!” ,”I love Python”, end=‘ ‘)
```

Syntax and Skeleton of a user-defined `function`

```
def name(parameters):

        statement
        statement
        . . .

        return value
```

Functions are blocks of resuable pieces of code

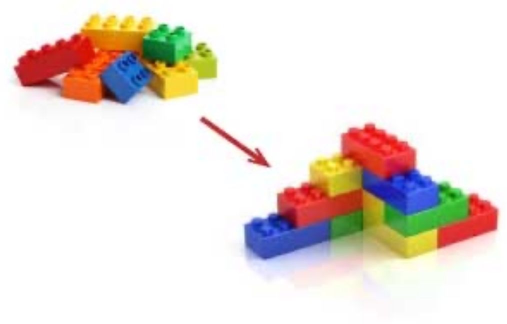


image source:
https://www.thescoopformommies.com/teaching-colors-to-preschoolers/lego-blocks-clip-art-ulsdyc-clipart/
https://www.123rf.com/clipart-vector/lego.html?sti=mt1rz1yzzh7428ua74|

Function name: Identifier by which it is called in the program

(Optional) **Arguments:** values passed to the function

Function Declaration: Starts with “def” that is not indented

```
def func_name(parameters):
    statement
    statement
    . . .
    return value
```

Colon; Don’t miss it!

Indentation: Tab or 4 spaces for each statement

Body: Statements executed each time a function is called

(Optional) **return value:** Can end function call and send data back to the main program

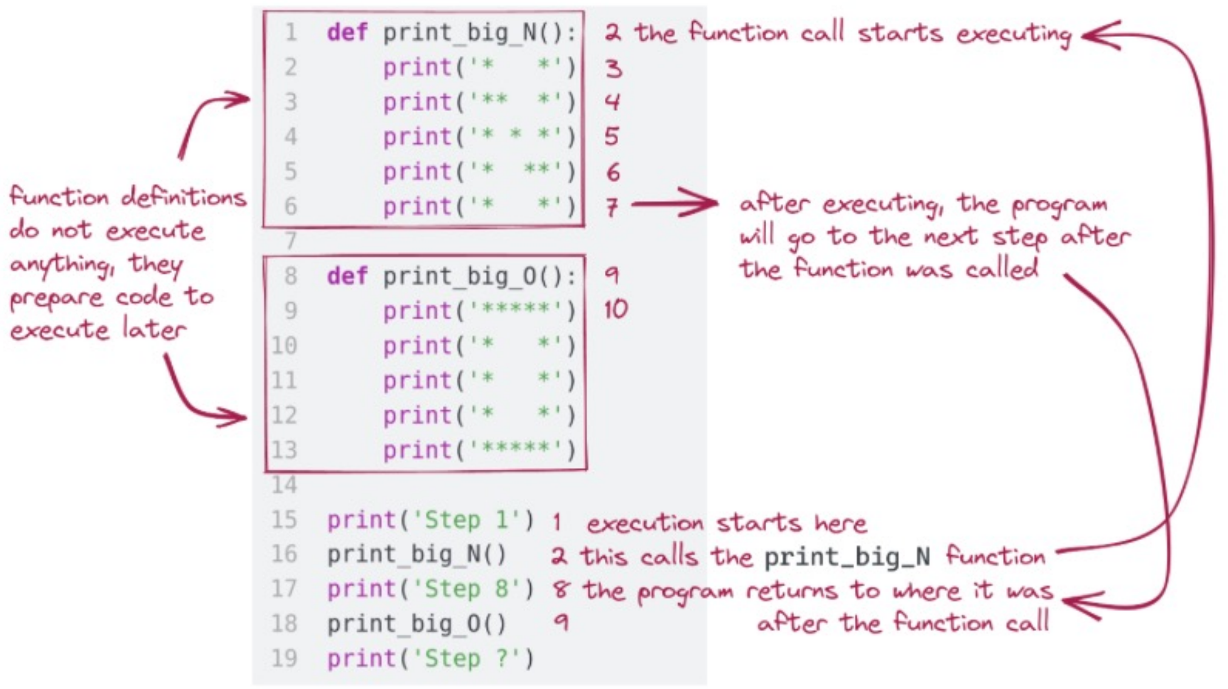
Function definition

`func_name()`

Function call

Tracing a `function` in a Python program

Source: https://www2.seas.gwu.edu/~cs4all/1012/unit0/module0.2.html



```
def print_big_N():
    print('*    *')
    print('**   *')
    print('* *  *')
    print('*   **')
    print('*    *')

def print_big_O():
    print('*****')
    print('*    *')
    print('*    *')
    print('*    *')
    print('*****')

print('Step 1')
print_big_N()
print('Step 8')
print_big_O()
print('Step ?')
```

function definitions do not execute anything, they prepare code to execute later

2 the function call starts executing

3

4

5

6

7

after executing, the program will go to the next step after the function was called

9

10

1 execution starts here

2 this calls the print_big_N function

8 the program returns to where it was after the function call

9

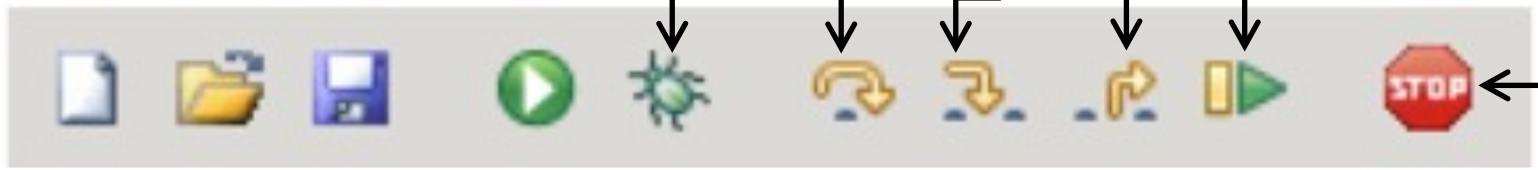
Demos

Allows you to debug your code.
A bug is another name for a problem.

The arrow icons allow you to run your programs step by step.
These icons are used after you press the bug icon

The resume icon allows you to return to play mode from debug mode.

The stop icon allows you to stop running your code



Built-in function `range()`

Python's `range()` function

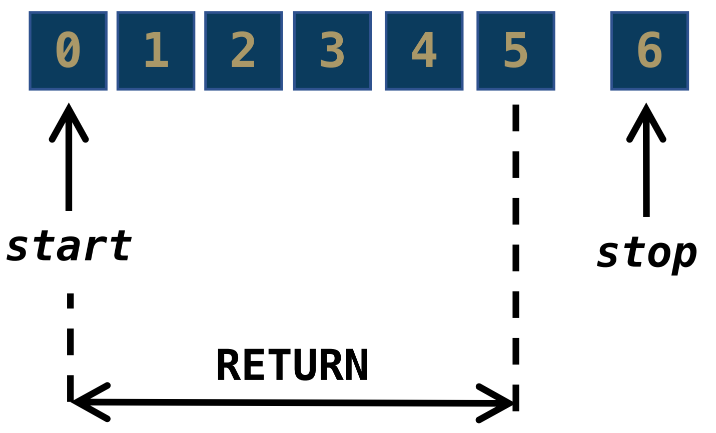
returns a sequence of numbers

works only with integers

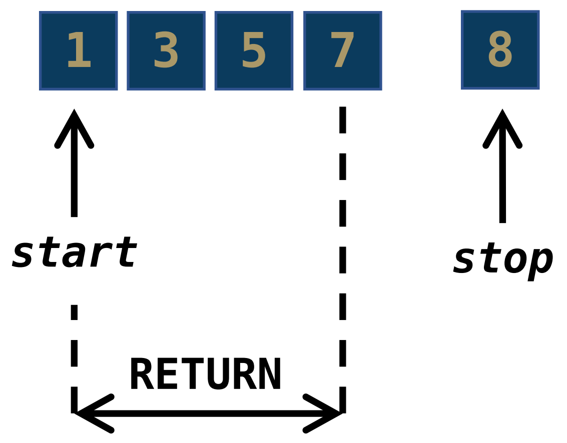
```
start:   at the value (default = 0)
step:    up or down at the increment value (default = 1)
stop:    at the value but not including it
```

`range(stop)` `range(start, stop)` `range(start, stop, step)`

```
>>> range(6)
```



```
>>> range(1,8,2)
```

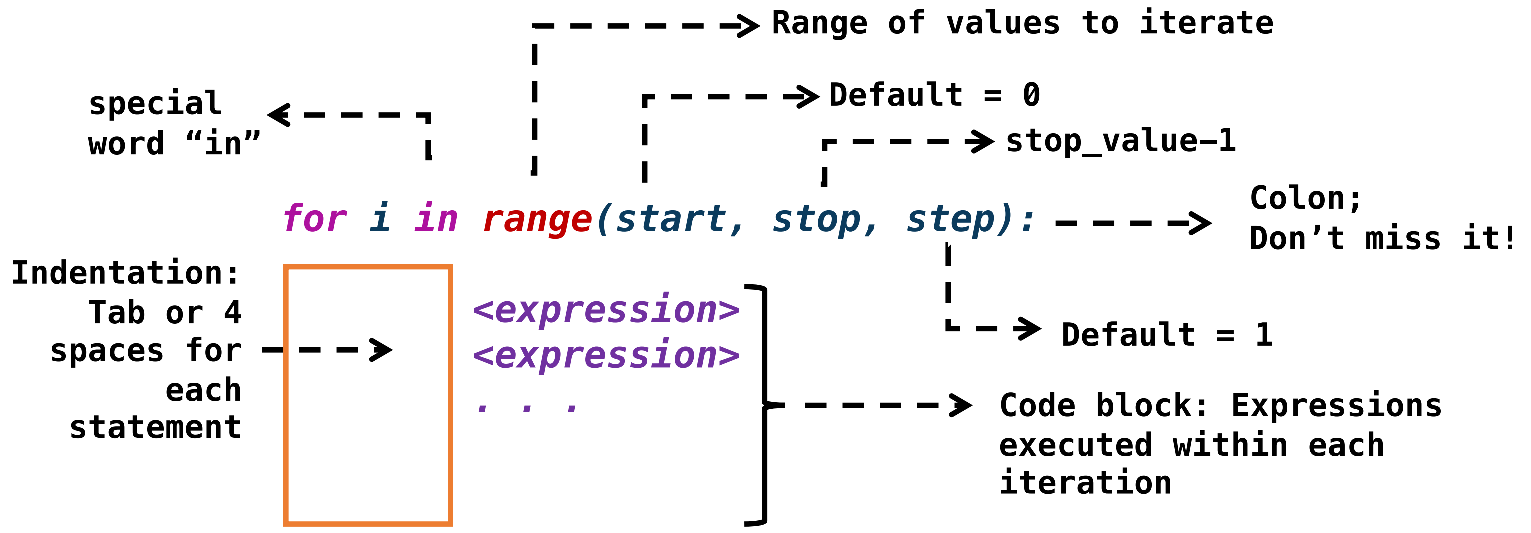


`range()`
- can be utilized in (for) loops
- to specify a range to iterate or do repetitions

Demos

Skeleton of the for-loop

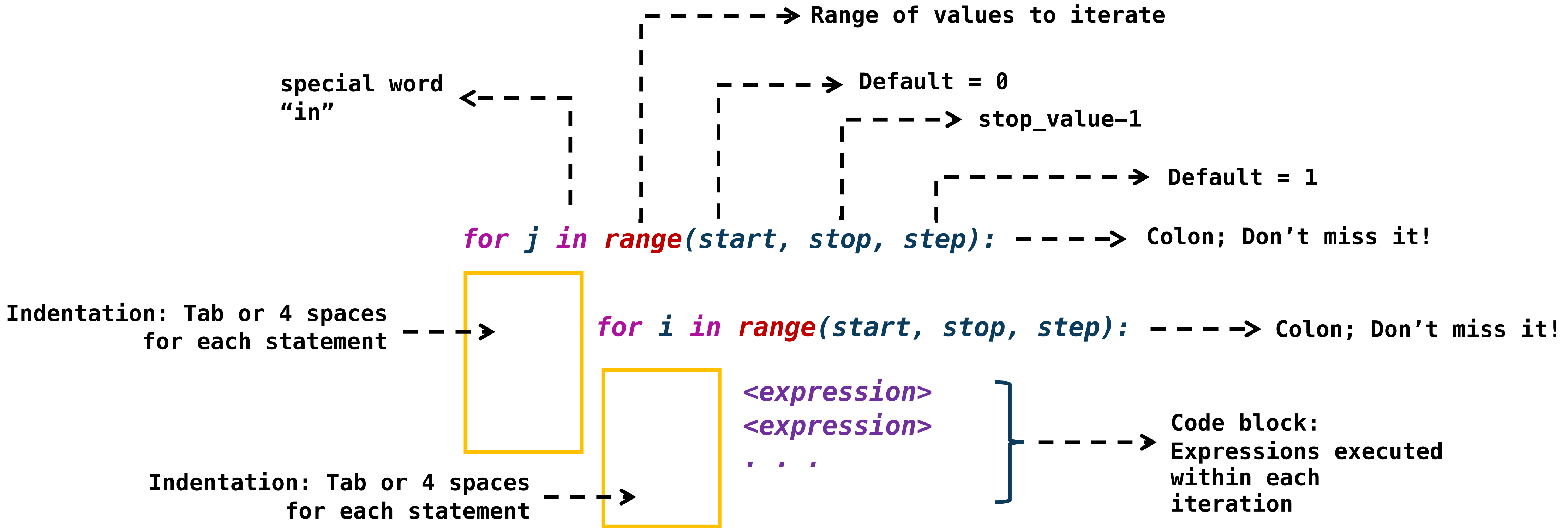
```
for i in range(start, stop, step):

        <expression>
        <expression>
        . . .
```


Range of values to iterate
Default = 0
stop_value-1
special word "in"
for i in range(start, stop, step):
Colon; Don't miss it!
Indentation: Tab or 4 spaces for each statement
<expression>
<expression>
. . .
Default = 1
Code block: Expressions executed within each iteration

Skeleton of the nested for-Loop

Demos

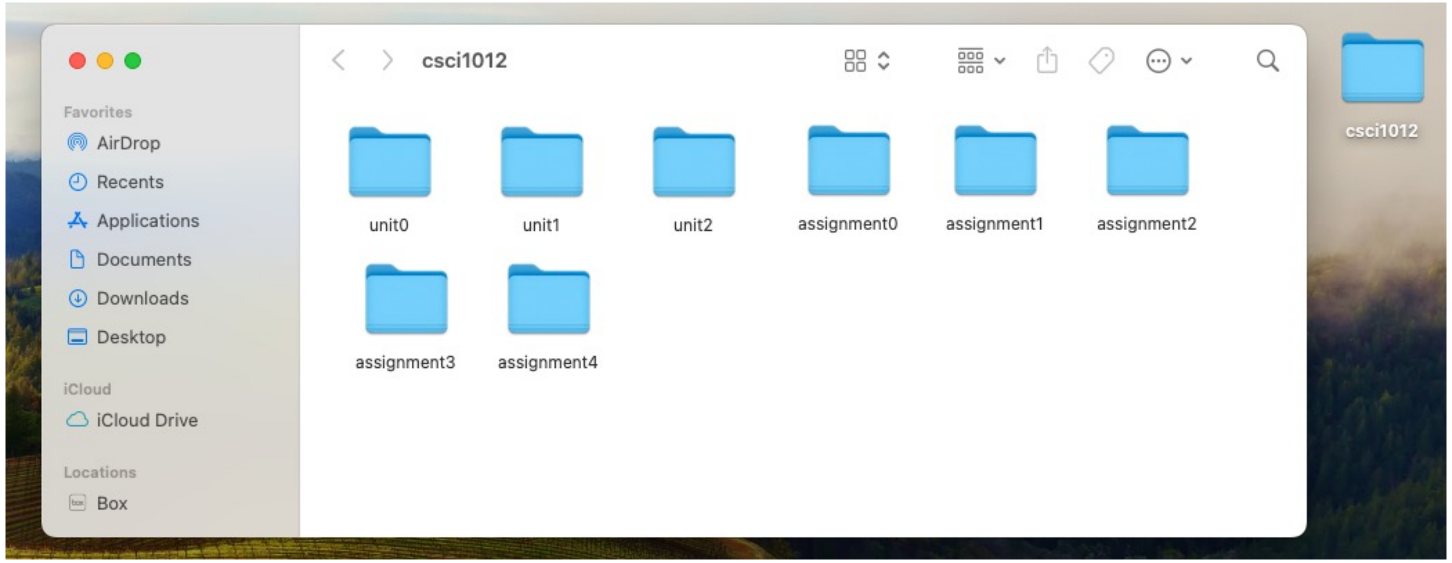
```
for j in range(start, stop, step):
        for i in range(start, stop, step):
                <expression>
                <expression>
                . . .
```

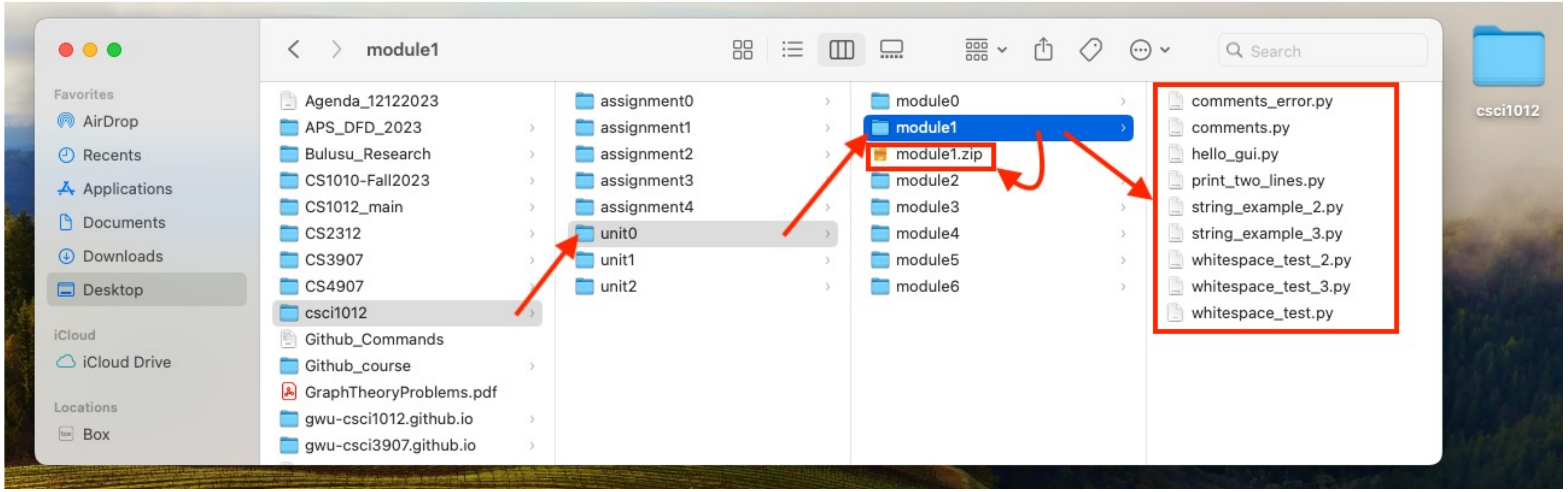


```
for j in range(start, stop, step):
        for i in range(start, stop, step):
                <expression>
                <expression>
                . . .
```

Range of values to iterate
special word "in"
Default = 0
stop_value–1
Default = 1
Colon; Don't miss it!
Indentation: Tab or 4 spaces for each statement
Colon; Don't miss it!
Code block: Expressions executed within each iteration
Indentation: Tab or 4 spaces for each statement

File-folder-structure



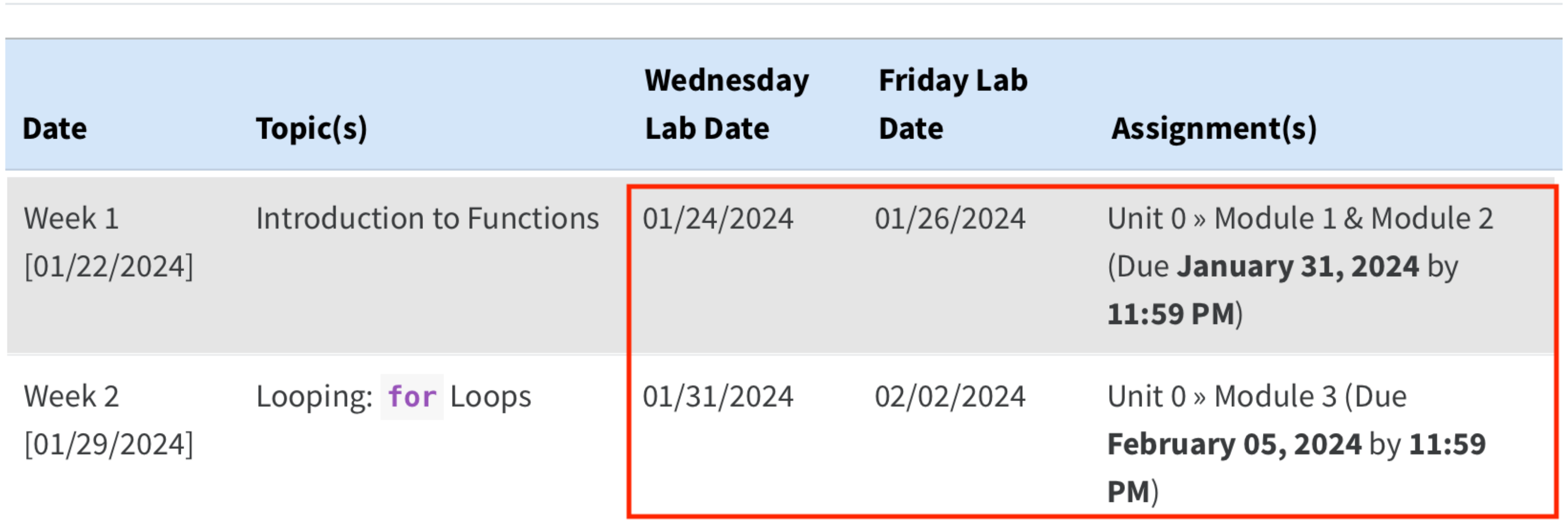


School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

GW

HWs

- Due dates
- Late work
- Extensions



Date	Topic(s)	Wednesday Lab Date	Friday Lab Date	Assignment(s)
Week 1 [01/22/2024]	Introduction to Functions	01/24/2024	01/26/2024	Unit 0 » Module 1 & Module 2 (Due **January 31, 2024** by **11:59 PM**)
Week 2 [01/29/2024]	Looping: `for` Loops	01/31/2024	02/02/2024	Unit 0 » Module 3 (Due **February 05, 2024** by **11:59 PM**)

- **Office hours location change:** Friday 10:00 AM to 2:00 PM is SEH B1280
- **CSCI 1012.36 (CRN: 94171)** - Moved to TOMP 306
- **CSCI 1012.33 (CRN: 94168)** - Moved to TOMP 309
- **IMPORTANT:** Please attend the ONLY lab that you registered into.

Late Work

- **Late work is not accepted, with the following exceptions:**
  - Every student many turn in **as many as four (in total, not each) assignments or modules 48 hours after the deadline with no penalty**. Requesting an extension is not necessary.
- **Extensions** will be **granted should there arise circumstances beyond your control** that impede your ability to complete coursework.
  - Notify your professor as soon as feasible in these cases.
    - Examples of such circumstances include (but are not limited to) illness, death in the family, and loss of housing. To ensure fairness toward all students, we will request documentation of such circumstances.