



CS 1112: Introduction To Programming

Exam 1 Review

Dr. Nada Basit

basit[at]Virginia[dot]edu

Friendly Reminders

- Your **safety** and **comfort** is important!
 - If you choose to wear a mask you are welcome to do so
 - *We will interpret wearing a mask as being considerate and caring of others in the classroom (not that you are sick), and realize that some may choose to mask to remain distanced*
- Remember to always be **kind, respectful, supportive, compassionate and mindful of others!** 😊
- Be an **active** participant in your learning!
You're welcome and **encouraged** to ask questions during class!
- If you feel **unwell**, or think you are, **please stay home**
 - *Contact us! We will work with you!*
 - Get some rest 😊
 - View the recorded lectures – *please allow 24-48 hours to post*



Announcements

- **Quiz 1, 2, 3, 4** :: View scores on Canvas, view details/comments (+ score) on Sherlock
- **PA03 (Functions)** graded (see Gradescope and Canvas)
- **Exam 1** is on **February 26, 2025 (Wednesday)**
 - If you have **SDAC** *time and/or distraction-free accommodations*, please **book** a time slot with SDAC to take the exam at their facility (*any time* on the same day, not another day, please, unless you've spoken with me ahead of time!)
 - **Everyone else**: please **come to class as normal!** 😊 (*Please show up on-time!*)

Please bring your UVA ID card with you to the exam!
Check-in with TA as you leave!

Exam 1: Policies

- Closed-book, closed-notes, closed Internet, ... closed everything ☺
- Closed PyCharm (IDE)
- No collaboration at all - *must represent your individual effort*
- Location: Taken in-class (in-person)
- Exam taken on **Sherlock** (the **only** site you're allowed to have open!)
 - Please do bring your own laptop (already charged, if possible!)
- Duration: class time - 1 hours and 15 minutes

Please bring your UVA ID card with you to the exam!
Check-in with TA as you leave!

If you have laptop issues leading up to the exam...

Reminder: CS Laptop Loaner Program

- This course requires students to have a **laptop**
- I realize that not everybody might have one (nor necessarily need one for their desired major / path...)
- If you do not have a laptop for any reason... *not to worry!*
- The CS department's Systems staff has a notebook / laptop loaner program and will be able to loan you a notebook / laptop computer for the duration of the semester if you don't have one or if you cannot afford one.
 - Also available if your laptop is broken and under repair, we can arrange for you to receive a loaner laptop for a week or two until your own laptop is fixed

Interested? Link: https://www.cs.virginia.edu/wiki/doku.php?id=cs_laptop_loaner

I am happy to be your sponsor. Please let me know.

Exam 1: Resources to Study From

- Quizzes 1, 2, 3, and 4
- Programming Assignments PA00 – PA03
 - Don't worry about PA01 (Turtles)
- Course Materials on Canvas (all: incl. Style & Debugging)
 - Except slides relating to Turtles
- Python Scripts on Canvas (all)
 - Except scripts relating to Turtles
- In-class activities (all)
 - ... you guessed it...
except activities relating to Turtles

No Turtles on Exam 1



Don't forget about me! ☹️

General

- Emphasis on content and *reading / understanding code*
 - May only be asked to write a very small amount of code
 - Read code and answer questions about it
 - Given input, trace code, what is the output / what does the code do?
 - Given code, what might be wrong with it?
 - Provide an example scenario for a given programming concept
-
- Will *all* of these topics be on the exam? *No. Impossible to cover everything! (Many will be)*
 - Is everything within these slides representative of Exam 1 material? *No. This is a summary!*


Summary of Topics

Basics

- `print()` function, concatenation (+)
- Print with end and sep
- Comments
 - `'''comment'''`
 - `"""comment"""`
 - `# comment`
- Escape characters: `\n` `\t`

Basics

We have seen examples that uses **lists** as part of the code, like in a for-loop, etc. so in this context we may see lists.



- **Variables**

- Different value types: int, float, Boolean, string, list (not: tuple, dictionary)
- `type(x)`: gives the data type

- **Arithmetic operators:** + - * / // ** %

- `//`: integer division (contrast with "regular division" where result is always a float!)
- `**`: power (`a**b` means "a to the power of b")
- `%`: modulus

- **Assignment statement**

- Assignment vs asking about Equality (`=` vs `==`)

- **Boolean Operations:** and or not

Basics

- Type Casting

- E.g.,
`f = 7.11`
`int_f = int(f)`
- Beware of invalid type casting
- E.g.,
`text = "42abc"`
`num = int(text)`

- Pseudocode

- Writing and reading/understanding pseudocode

The building blocks of programs [Day 3, slide 12-13]

- Sequence
- Repetition (loops)
- Conditions / decisions
- Named actions (functions)

Algorithms

- What makes a **good algorithm**? [Day 3, slide 20]
 - Unambiguous
 - Executable
 - Terminating
 - Deterministic
 - Correct
- What are **bad algorithms**?
 - Ambiguous
 - Not executable
 - Non-terminating
 - Incorrect

input() function

- Obtaining information from the user
- Input obtained from the keyboard
- Data transferred and stored as a **string**
 - Type cast afterwards if needed

• E.g., `num_of_sales = input("How many items did you sell?")`

Type cast: `num_of_sales = int(input("How many items did you sell?"))`

Booleans and Conditionals

- **Booleans**

- Boolean data type
- Values (True and False)
- Operators (not, and, or)
- Comparison operators
(`<` `>` `<=` `>=` `==` `!=`)
- Boolean expression / relational expression

- **Conditionals**

- Conditional decision statement
- Examples:
 - if
 - if-else
 - if-elif
 - if-elif-else
 - match-case

Mutability

- Strings are **immutable**!
 - Cannot change the string (cannot modify it)
 - You can swap out a string for another (a new string can include pieces of old strings)
- Various types and their mutability

<u>Type</u>	<u>Stores</u>	<u>Syntax</u>	<u>Mutable?</u>
Range	ints	<code>range(3, 7)</code>	no
String	characters	<code>"Hello", "abc 123"</code>	no
List	anything	<code>[1, 2, 3, 6, "hello"]</code>	yes
Tuple	anything	<code>(1, 2, 3, 6, "hello")</code>	no

*Tuples will not
be on Exam 1*

FUNCTIONS (1 of 3)

- Reasons to use functions [Day 12, slide 6]
- Function header / creating functions / parts of a function [Day 12, slides 7-9]
- Indentation (part of the syntax)
- return statement
- Calling functions - function call
- Parameters vs arguments
- Side effects of functions
- Use informative function names

FUNCTIONS (2 of 3)

- **Comment** your code (in-line), additionally use: **multi-line docstrings**
- You can save useful functions in a file for reuse in the future
- Write code clearly and concisely
 - Increase readability
 - Increase ease of debugging / troubleshooting
 - Increase ease of writing the code in the first place
- **Functions calling other functions**
- **Scope** [Day 13, slides 11-22]
 - **Global vs local variables** ("car with tinted windows")
 - Recognize and identify local and global variables in given code

FUNCTIONS (3 of 3)

- Function arguments [Day 14 (most of these slides)]

- **Arguments** are

- Positional
- Named `my_func(a=10)`

- **Parameters** are

- Required, like "a" Optional, like "b", default value provided

`def another_func(a, b=17)`



- Mixing positional and keyword (named) arguments
 - Positional arguments (all) should come before any keyword arguments

Loops

- **For-loops** [Day 15, slides 26-41]
 - `for <variable> in <collection>`
 - Strings
 - Lists
 - Integers (**range**)
 - Remember indentation
 - `for-else`

- **While-loops** [Day 15, slides 7-21]
 - Guard condition (boolean expression)
 - Some aspect of the guard condition must change in the body of the loop
 - Remember indentation
 - `While-else`

-
- Accumulator pattern [Day 15, slide 42]
 - When to use each kind?
[Day 15, slides 47-48]
 - **Break** use **Continue** [Day 16]

Testing/Debugging & Style [Day 17]

- 3 broad categories of errors: **syntax, runtime, and logical errors**
- Debugging Techniques:
- `print()` statements
- **Calling functions**: expectation vs. actual
- **Trace code**
- **Commenting** should help in this endeavor
- Helpful to **name variables and functions appropriately** to increase readability (which in turn aids in code maintenance and debugging)
- **Style** - helps in readability and debugging, too! (Understand this in general)

Q&A

I'm happy to address any questions you have about the exam!



Good Luck on the Exam!