IDX G9 Computer Science S Study Guide Issue #1

By Leona Cai, Edited by Doris

Contents:

1.1 Computer Category and Programming Language

The History of Computers

- 3000 B.C.: The **abacus**, used for addition, subtraction, division and multiplication, it can also extract square roots and cubic roots
- 1973: Xerox introduced Ethernet (wired internet), allowing computers to connect
- Various computer companies were established, including *Microsoft* (Bill Gates and Paul Allen, 1975) with the first home kit-built computer, and *Apple* (Steve Jobs and Steve Wozniak, 1976) with the first computer with a single circuit board
- 1999: Wi-Fi was invented, replacing wired Ethernet connections

Categories of Computers

- **Special-Purpose Computers**: Designed for a particular function, executing the same stored set of instructions (EX. Microwave, washing machine etc.)
- **General-Purpose Computers**: Used for solving many different types of problems, available in a variety of sizes and capabilities.
    - **Microcomputers**: AKA personal computer (PC), can be placed on a desktop or carried from room to room
        - Size varies, from the smallest laptop computers/notebook computers to the largest type of microcomputer known as a workstation
    - **Mainframe Computers**: Powerhouse with massive memory and extremely rapid processing power, used for large businesses, scientific/military applications where computer needs to handle massive amounts of data or complicated processes
    - **Supercomputers:** Used for tasks requiring extremely rapid and complex calculations with hundreds of thousands of variable factors
        - Used for scientific research, weather prediction, aircraft design, nuclear weapons etc.

Languages

- **Programming Languages**: Agreed upon format of symbols that allow a programmer to instruct a computer to perform certain predefined tasks
- **Machine Languages**: Natural language of a computer, and is the only language that a computer can directly use. Its instruction is a binary string of 0s and 1s.
- **Assembly Language**: Consists of English-like abbreviations, making them easier to understand. It uses language translators called **assemblers** to convert them to machine code.
- **High-level Languages**: Machine-independent, does not require programmers to know anything about the internal structure of the computer

1.2 Programming Intro

Hardware and Software

- **Hardware**: Only understands the binary system
- **Software**: set of steps for instructions for computer hardware operations

Programming Problems

- Define the problem -> Plan the solution -> Write the code -> Testing/Debugging the program
- **Interactive mode:** Start with >>>, runs one line at a time
- **Script mode:** Uses ".py" file, runs all lines you wrote at once
- **Indentation**: Indicates a block of code

Math Operators

Operator	Operation	Example	Output
+	Addition	2+2	4
-	Subtraction	5-3	2
*	Multiplication	2*3	6
/	Division	10/2	5
**	Exponents	3**2	9

- Python follows the **PEMDAS** rule, so uses parentheses when needed

print() function

- A value that is passed to a function call is an **argument**
- EX. print("Hello World"), output: Hello World
- If it is a value, don't use quotation marks. Use quotation marks for letters

1.3 Variables and Data Types

Operator "+" is used to **concatenate** two strings as the operation

- **input() function**
  - It will return a string type, and the user can input
    - EX. input("What is your name"); output: What is your name; Then the user can input a name
- **Variables**
  - Containers for storing data values can store numerical or textual values
  - Rules for naming variables:
    - Can contain only letters, numbers, and underscores
    - Cannot start with a number
    - Spaces and **special characters** (-, !, @, #, %, ^, &, *) are not allowed

        - Variables are **case sensitive**, meaning age, AGE, and Age are different
        - Avoid using **reserved keywords**: EX."True", "False","or", "not", "and", "if"
    - **Camel Case**: makes compound names easier, EX. myList, listOfNumbers
        - Can contain only letters, numbers, and underscores
    - When assigning values, use "=" operator
        - Multiple assignment: multiple variables in a single statement, EX. x=y=z=50, or a,b,c=5,10,15

Data Types

- **Integer:** int(), converts a number/string into an integer
- **Float**: float(), approximations to real numbers, they are decimals
- **String**: str(), converts a number to a string

1.4 Expressions

Consists of values and operators so they always evaluate down to a single value

Importing Modules: from the standard library, contains related group of functions that can be embedded in your programs

- Math Modules
    - math.pi: on its own, it returns the first 15 digits of pi
    - math.ceil(x): returns an integer ≥ x (EX math.ceil(2.4); output: 3)
    - math.floor(x): returns an integer ≤ x (EX math.floor(2.4); output: 2)
    - math.sqrt(x): returns the square root of x (EX math.sqrt(9); output: 3)

1.5 List

A collection of items in a particular order, indicated by square brackets []

- Access an element in a list by its index: listName[index]. (EX. myList[3];output: the fourth element in the list)
- len() function: returns the number of elements in a list
    - Index of a list starts at 0. (1st element's index is 0, 2nd element's index is 1, etc.)
    - len(myList), (assuming myList has 5 elements); output: 5
- lst.index(value) function: finds the index of an element

Action	Methods/Functions
Modifying an element	**listName[index of element you want to change]=(new value/string)** EX. myList[1]=4; myList now equals [1,4,3,4]
Adding an element	**append():** adds the new element to the end of a list EX. myList.append(5); myList now equals [1,4,3,4,5]
	insert(,): insert (index,value) EX. myList.insert(2,6); myList now equals [1,4,6,4,5]
Removing an element	**del listName[**index**]:** removes the element of that index EX. del myList[1]; myList now equals [1,6,4,5]
	remove(): removes the specific value, deletes the first occurrence of the value you specify EX. myList.remove(1); myList now equals [6,4,5]

1.6 For-Loop

- split() function: splits a string to a list
    - str.split(x) will remove every x from str and return a list of the leftover
- for loop: repeatedly perform the same task with each element in a list
    - for variable in range OR for variable in myList; enter the task you want to perform
    - iterate a block of statements several times
- sum() function: sum of the entire list

1.7 For-Range

- range() function/syntax: generates a sequence of numbers in that range
    - for i in range(5); print(i); output: 0, 1, 2, 3, 4 (starts from 0, ends at the integer < 5)
    - for I in range (2,6,2): (start, stop, step)
        - output: 2, 4
        - if there isn't an end (2,6,-3), then it will return nothing
- random numbers: use module **random**. (import **random**)
    - **randint(a,b)**: returns in integer in the range [a,b]
    - EX. random.randint(1,5); output: 4

1.8 If Statement

- Boolean values: True or False
- Relational Operators

Operator	**Description**	**Example**	**Output**
<	Less than	2<3	True
>	Greater than	3>4	False
<=	Less than or equal to	3<=9	True
>=	Greater than or equal to	8>=7	True
==	equal	2==3	False
!=	Not equal to	2!=2	False

- **If Statement**: **if** condition: -> statements
  - The condition has to be a Boolean expression that evaluates to True or False
  - **If else:** if condition: statement 1; else: statement2
  - **If elif else**: if condition: statement 1; elif: statement 2; else: statement 3

1.9 Boolean Operators

- Boolean operators: evaluate the expressions down to a Boolean value
  - () > not > and > or: meaning the computer evaluates Boolean operators in this order
  - **and** operator truth table:
    - True and True: True
    - True and False: False / False and True: False
    - False and False: False
  - **or** operator truth table:
    - True or True: True
    - True or False: True / False or True: True
    - False and False: False
  - **not** operator truth table:
    - not True: False, not False: true
- Remainder(%): gives the remained of division (EX: 22%5=2)
- Integer division: floor value of a quotient produced by a division (EX: 1234//100=12)
- Precedence: <u>**(Exponents)</u> > <u>(*, /, %, //)</u> > <u>(+ -)</u> > <u>(<=, <, >, >=, !=, ==)</u> > <u>(not, and, or)</u>