Summary | Programming Fundamentals

Introduction

> **Note**
>
> Programming Fundamentals is probably the less-organized section at the moment. Let me know how I can improve this.

This module includes 3 sections:

- Programming basics (with python v3.10.9)
- Theories beyond coding
- Hardware

Important points

Confusion about unit prefixes

In computing, the prefix *kilo* —just like other prefixes— has been used to refer either 2^{10} or 10^3 depending on the context.

- 10^3 - Marketing of disk capacities (by disk manufacturers)
- 2^{10} - Memory capacities, and file sizes, disk capacities by operating systems

To avoid this confusion, 2 unit prefixes are used while measuring amounts of data.

- SI prefixes
  Defined by ISO. Based on powers of 10^3. Examples: kilo, mega, giga.
- Binary prefixes
  Defined by IEC. Based on powers of 2^{10}. Examples: kibi, mebi, gibi.

Interning

Interning is re-using objects of equal value on-demand instead of creating new objects. This is done for memory efficiency. Frequently used for numbers and strings in different programming languages.

```
a = 120
b = 120

print(a is b) # True

c = 2000
d = 2000

print(c is d) # False
```

In the above code, `120` is intered by the Python interpreter but not `2000`. Python's integer interning is done only for numbers in the range: `[-5, 256]`

Python interpreter also interns small strings.

```
a = "abcd"
b = "abcd"

print(a is b) # True

# Both text are the same
c = "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequuntur perferendis iste ipsa nat
d = "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequuntur perferendis iste ipsa nat

print(c is d) # False
```

Strings in python can be manually interned using `sys.intern` function.

```
a,b=8,8
c=8
d=8
```

Likewise, in the above code, only 1 integer object is created.

Practice Resources

Programs

The programs are listed in no specific order.

1. **is prime number**: A program that takes in a number n and outputs whether its a prime number or not.
2. **factors**: Take in a number from user. Output all of its factors.
3. **n-th factorial**: A program that takes in a number n and outputs n-th factorial.
4. **is perfect number**: A program that takes in a number n and outputs whether its a perfect number.
5. **fibonacci numbers**: A program that takes in a number n and prints all fibonacci numbers less than or equal to n.
6. **determinant of matrix**: Take in a matrix from user. Output the determinant of the matrix. First try for 2×2. Then go higher-ordered matrices.
7. **pascal's triangle**: Take n from user input. Print pascal's triangle to n rows.
8. **is valid palindrome**: Take a string input from user. Output if the input is palindrome or not. A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers. Try not to use `[::-1]`.
9. **armstrong numbers**: Take n from user input. Print all armstrong numbers (in base 10, of course) between 0 and n (inclusive).
10. **letter analysis**: Take a text input from user. Find how many times each letter is being used in that string. Use a `dictionary` to store the data. Output the final results. Try to read the text from a `.txt` file as well.
11. **word length analysis**: Take a string input from user. Print length of each word separated by a space. Try to include the summary using a `dictionary`.
12. **letter expanding**: A program that converts *b3j8k2* to *bbbjjjjjjjjkk*. The number can be 1 to 99.
13. **binary addition**: Take in 2 numbers in binary (as strings) and output the sum of both numbers. Try not to use `bin` function.
14. **big integer addition**: Given *a very large integer* represented as a list, where each `digits[i]` is the i^{th} digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. Increment the large integer by one and return the resulting array of digits. Don't construct a `int` object.

15. **stack implementation**
16. **queue implementation**

Platforms

- Codewars - https://codewars.com (my most preferred one)
- HackerRank - https://hackerrank.com
- Leetcode - https://leetcode.com (my least preferred one)

> ⚠ **Hard Problems**
>
> If a problem from one of these platforms feels too hard for you, you can just skip and do another problem.

Number Systems

A writing system for expressing numbers. Each number system defines a set of symbols that each represent a specific value.

Base (or radix)

Number of symbols defined by a number system.

Commonly used number systems

- Base 10 - $0 - 9$
- Base 2 - $0, 1$
- Base 8 - $0 - 7$
- Base 16 - $0 - 9, A - F$

> ⚠ **Caution**
>
> These are required for s1:
>
> - Converting integers and floats between number systems
> - Addition, subtraction, multiplication, division in base 2
>
> But I don't know how to include it in a easy-to-understand way. ⬤

One's & Two's Complement

One's complement

The ones' complement of a binary number is the value obtained by flipping all the bits in the binary representation of the number.

- If one's complement of a is b, then one's complement of b is a.
- Binary representation of $a + b$ will include all 1 s.

One's complement system

In which negative numbers are represented by the inverse of the binary representations of their corresponding positive numbers. First bit denotes the sign of the number.

- Positive numbers are the denoted as basic binary numbers with 0 as the MSB.
- Negative values are denoted by the one's complement of their absolute value.

For example, to find the one's complement system representation of -7, one's complement of 7 must be found. $7 = 0111_2$. One's complement of -7 is 1000.

Two's complement

In which negative numbers are represented using the MSB (sign bit).

If MSB is:

- 1 : negative
- 0 : positive

Positive numbers are represented as basic binary numbers with an additional 0 as the sign bit.

For example:

Following equation can be used to convert a number in two’s complement form to decimal.

$$b = -2^{n-1}b_{n-1} + \sum_{k=0}^{n-2} 2^k b_k$$

Steps

1. Starting with the absolute binary representation of the number
2. Add a leading 0 bit being a sign bit
3. Find the one’s complement: flip all bits (which effectively subtracts the value from -1)
4. Add 1, ignoring any overflows

Floating-point Representation

IEEE 754 standard.

2 types:

- single precision
- double precision

Single precision

Uses 32 bits.

- sign bit - 1 bit
- exponent - 8 bit
- mantissa - 23 bit

Sign bit

0 if positive or zero. 1 if negative.

Exponent

Exponent field range - $[0, 255]$. In this range $[1, 254]$ is defined for normal numbers. 0 and 255 are reserved for subnormal, infinite, signed zeros and NaN.

To support negative exponents, we subtract 127 (half of 254) from this range. $[-126, 127]$. This range is the representable range.

Mantissa

In scientific notation, the part that doesn't contain the base and the power.

In binary scientific notation, there will always be exactly one 1 bit before the dot. So we don't include that one.

Example

Take 31.3125.

- In binary: 1111.0101_2
- In binary scientific notation: $1.1110101_2 \times 2^3$
- Add 127 to exponent: 130
- Convert exponent to binary 10000010
- Write the final result: $0\ 10000010\ 00000000000000001110101$

Take 0.125.

- In binary: -0.001_2
- In binary scientific notation: $-1.0_2 \times 2^{-3}$
- Add 127 to exponent: 124
- Convert exponent to binary 01111100
- Write the final result: $1\ 01111100\ 00000000000000000000000$

Double precision

Uses 64 bits.

- sign bit - 1 bit
- exponent - 11 bit
- mantissa - 53 bit

Sign bit

0 if positive or zero. 1 if negative.

Exponent

Exponent field range - $[0, 2047]$. In this range $[1, 2046]$ is defined for normal numbers. 0 and 2047 are reserved for subnormal, infinite, signed zeros and NaN.

To support negative exponents, we subtract 1023 (half of 2046) from this range. $[-1022, 1023]$. This range is the representable range.

Mantissa

In scientific notation, the part that doesn't contain the base and the power.

In binary scientific notation, there will always be exactly one 1 bit before the dot. So we don't include that one.

Example

Take 31.3125.

- In binary: 1111.0101_2
- In binary scientific notation: $1.1110101_2 \times 2^3$
- Add 1023 to exponent: 1026
- Convert exponent to binary: 10000000010
- Write the final result:
  $0\ 10000000010\ 0000000000000000000000000000000000000000000000001110101$

Take 0.125.

- In binary: -0.001_2
- In binary scientific notation: $-1.0_2 \times 2^-3$
- Add 1023 to exponent: 1020
- Convert exponent to binary: 1111111100
- Write the final result:
  $1\ 1111111100\ 0000000000000000000000000000000000000000000000000000$

String Representation

A way of representing non-numerical data.

Commonly used encodings

ASCII

Abbreviation for American Standard Code for Information Interchange. Uses 7 bits for letter representation and a parity bit (MSB). Can represent latin alphabet, digits, punctuations, and control characters.

Major limitation in ASCII is it can’t support multiple languages.

Unicode

Uses 32 bits. Supports multiple languages and emojis. Characters are presented by code points. A code point is a integer (in base 16).

Data Structures & Algorithms

Data structures

Common data types that are useful in many different places.

Abstract Data Type

A data type that has well defined properties and operations but not implementation.

Examples

- Array - fixed-length, one-dimensional
- Set
- Stack - Last in; first out
- Queue - First in; first out
- Binary search tree

> **Note**
>
> Implementations of stacks, queues, and binary search trees are required in s1.

Algorithms

An algorithm is a finite set of instructions, used to solve a problem.

> **Note**
>
> In s1, only sorting algorithms are discussed.

Selection sort

Here is selection sort algorithm that sorts a list of numbers in-place:

```
def selection_sort(arr):
    for current_starting_index in range(len(arr)):
        smallest_index = current_starting_index
        for i in range(current_starting_index + 1, len(arr)):
            if arr[i] < arr[smallest_index]:
                smallest_index = i
```

```
        arr[smallest_index], arr[current_starting_index] = arr[current_starting_index], arr[smallest
```

Bubble sort

Here is bubble sort algorithm that sorts a list of numbers in-place:

```
def bubble_sort(arr: list[int | float]):
    sorted_index_count = 0
    while sorted_index_count < len(arr):
        for i in range(len(arr)-sorted_index_count-1):
            if arr[i] > arr[i+1]:
                arr[i], arr[i+1] = arr[i+1], arr[i]
        sorted_index_count += 1
```

Software Engineering

Software

Refers to all the related things that are required to make a software system work.

Includes:

- programs
- configuration files
- system and user documentation
- user support system
- bug fixes and updates

Software engineering

An engineering discipline that is concerned with all aspects of software production. From the initial stage of writing the requirements to maintaining it while being used.

Software process

Set of activities that are associated with the development of a software product.

Fundamental activities that are common to all types of software development processes:

- Specification - defining the software to be produced and the runtime constraints
- Development - design and development of the software
- Validation - testing phase to check if the software meets the specifications
- Evolution - software is modified to adapt to new specifications

Waterfall

All before-mentioned activities are done sequentially, as clear separate phases. One phase is completed before the next phase is started.

Iterative & incremental

System is developed in iteration. Smaller parts of the system is completed in each iteration, that includes:

- Small amount of requirements specification
- Design and development for the specification
- Validation for the developed parts

Component based

Existing components are combined to implement the system. Main concentration is on the integration of the components.

Quality of software

Can be measured using these aspects:

- Maintainability - how easy it is to making changes
- Dependability - how secure, reliable it is to failures or other unusual activities
- Efficiency - how efficiently hardware resources (such as memory, processor time, disk space) are used
- Usability - how easy it is to use the software from user’s perspective
- Robustness - how resilient it is to invalid inputs

Challenges in software engineering

- Complexity
  - Essential - inherent, difficult to overcome
  - Accidental - not inherent, can be overcome
- Conformity
- Changeability - expected to be changeable to greater extent
- Invisibility - not visualizable
- Can’t guarantee defect free software - no amount of testing can prove absence of defects

Computers

An electronic device for analyzing and storing data, making calculations, etc.

Originates from the word compute which means calculate. Computers can be programmed to process data following some finite set of instructions.

Types of computers

Personal computers

Or PCs in short. Intended to be used by a single person. First introduced by IBM. Built around a family of microprocessors referred to as “8086” (manufactured by Intel). Therefore its structure is referred to as “x86 architecture”.

An alternate system was Macintosh (introduced by Apply). Due to differences in hardware architecture between x86 and Macintosh, software used by these systems were not portable to the other.

Types of personal computers:

- Desktop - designed to sit on or under a desk. too big to be carried around.
- Workstation - desktop computers with high specs. powerful. built for specialized use cases.
- Laptop - portable. categorized under mobile computers.
- Tablet - even lighter than laptops. include touch input support.
- Handheld - can fit in our hands. example: personal digital assistant.
- Smartphone - cellular phones with advanced features
- Wearable - can be worn such as glasses

Network servers

Powerful personal computers with special hardware and software that enable them to function as primary computers in a network. They may be setup in groups called clusters or server farms.

Mainframe computers

Used in large organizations where many people, even thousands, frequently need to access the same data. Traditionally, each user accesses a mainframe computer through a device called a terminal.

Minicomputers

More powerful than personal computers but less powerful compared to mainframes. Aka. midrange computers.

Supercomputers

Most powerful, physically largest computers. Have thousands of processors.

Embedded computers

Computers that reside in a device and not directly visible. Aka. invisible computers. These devices are referred to as *smart* because they can interact with the environment in a more intelligent, adaptive, and efficient way.

Components of a computer system

A computer is a system. A complete computer system consists of:

- Hardware: Tangible elements that can be seen and touched
- Software: Programs that allow users to use the computer system and control its activities. Not tangible.
- Data: Individual facts or pieces of information that is input to the computer for processing or produces as output after processing.
- User(s)

Information

Computers can process various types of data. When a communication link (such as Internet) is provided, the data can be transferred to other users despite of distance. The computer and communication technologies that made this possible are together referred to as information technology or IT (or, sometimes as information and communication technology or ICT). Computers are therefore at the heart of IT.

An information system is a system with well-defined procedures and techniques to collect, store, process, and disseminate information.

Computer Hardware

Major components of a PC

- Motherboard
  - Central Processing Unit (CPU / Microprocessor)
    Communicates with all the other systems and subsystems. Responsible for controlling and managing rest of the system.
  - Memory
  - Graphic / display controller
  - Network / Wi-Fi controllers
  - Audio Interface (Sound card)
  - Universal Serial Bus (USB) controller
- Input / Output sub system
  enables interaction with the users
  - Display Monitor (VDU)
  - Keyboard
  - Mouse
  - Scanner
  - Printer
  - Microphone / Speakers
  - Joystick / Game controller
- Secondary storage devices
  long-term storage of large volumes of data. Examples:
  - Solid State Drive (SSD)
    stores data on solid-state flash memory made with silicon. fast. high capacity.
  - Hard Disk Drive (HDD)
    fast but slower than SSD. high capacity.
  - CD-ROM / DVD drive
  - Floppy disk drive
  - Tape drives (DAT drives)
    stores data on magnetic tapes. high capacity. cost-effective. slow. mainly used for data backups nowadays.

- Chassis
  the platform that holds all other components
    - Power supply unit
    - Fan / cooling system

Traditional computer system

Can be classified into 2 types.

Single-user computer system

Can only be used by one user at a time.

Modern computer system

Motherboard

Large circuit board where all (or majority of the) components of the PC are mounted. Aka. main board.

Computer Memory

2 types:

- Permanent or non-volatile
- Temporary or volatile - will be lost when power is off

Instructions and data for the CPU is sent from the memory. Results are sent back to the memory.

Consists of an array of consecutive memory locations. Each location is identified by a memory address and stores a single piece of data, usually a byte. CPU can either read or write a single memory location at a time.

Memory Bus

A set of electrical connections that connect memory locations with CPU.

3 types:

- Address bus - used to indicate address of a memory location. goes from CPU to memory
- Control bus - used to send control information (read request RD or write request WR) from CPU to memory.
- Data bus - actual data transmission. bidirectional.

Writing data:

1. Address bus is set with the memory address
2. Data bus is set with the data to be written
3. CPU activates WR in control bus

Reading data:

1. Address bus is set with the memory address
2. CPU activates RD in control bus
3. Data is fetched using data bus

Types of Memory

Read Only Memory

Memory is written (hardcoded) when they are fabricated as ICs. Used to store initial start-up programs. Not economical to produce in small quantities.

Example: ICs in the market that have various melodies.

Programmable Read Only Memory

Similar to ROM but, the content can be written (using special equipment) once after the manufacturing process. Cost effective compared to ROMs.

UV Erasable PROM

Similar to PROM but can be written multiple times. The content should be erased using UV light before new content is written. Both erasing and programming process require special equipment.

Electrical Erasable PROM

Similar to UVEPROM but the content can be erased by applying a special high voltage.

Flash ROM

A special type of EEPROM that can be erased or programmed while in the application circuit. The contents remains unchanged even after a power failure.

Commonly used in modern PCs, various networking devices such as routers and firewalls and memory pens (also referred as memory sticks or USB pens).

Read Write Memory

Aka. RAM. Volatile.

2 types:

- Static RAM - uses transistors
- Dynamic RAM - uses capacitors. bulk of the PC memory is made using DRAM.

Transistors	Capacitors
Uses semiconductors	Uses semiconductors
High speed switching	Slower performance
Retains state forever (if power is supplied)	Discharges after some time, needs refreshing (in µs scale)
More reliable	Less reliable
Low transistor density	High capacitor density
High power consumption	Low power consumption
High cost per bit	Low cost per bit

Memory Modules

Set of memory ICs presented as a single memory block to the motherboard .

Type of memory modules:

- SIM - Single Inline Memory Module
- DIMM - Dual Inline Memory Module (64-bit wide 168-pin)
- DDR-DIMM - Double Data Rate-DIMM

Memory Characteristics

- Access speed – time taken for the CPU to read from or write to memory
- Cycle time – time taken to complete 1 memory access operation
- Packing Density - memory capacity per unit area
- Power consumption
- Cost - cost per unit of memory capacity

Memory Hierarchy

Modern CPUs are much faster than the speed of memory. The memory has to be organised in such a way that its slower speed does not reduce the performance of the overall system.

The ultimate objective of having a memory hierarchy is to have a memory system with a sufficient capacity and which is as cheap as the cheapest memory type and as fast as the fastest memory type. The main idea is to use a limited capacity of fast but expensive memory types and a larger capacity of slow but cheap memory types. Special methods are used to store the frequently used items in the faster devices and others in slower devices.

Computer Software

A program is an ordered sequence of instructions that the hardware can execute. Software can be further categorized as system software and application software.

Embedded System

TODO

Trends in Computing

TODO