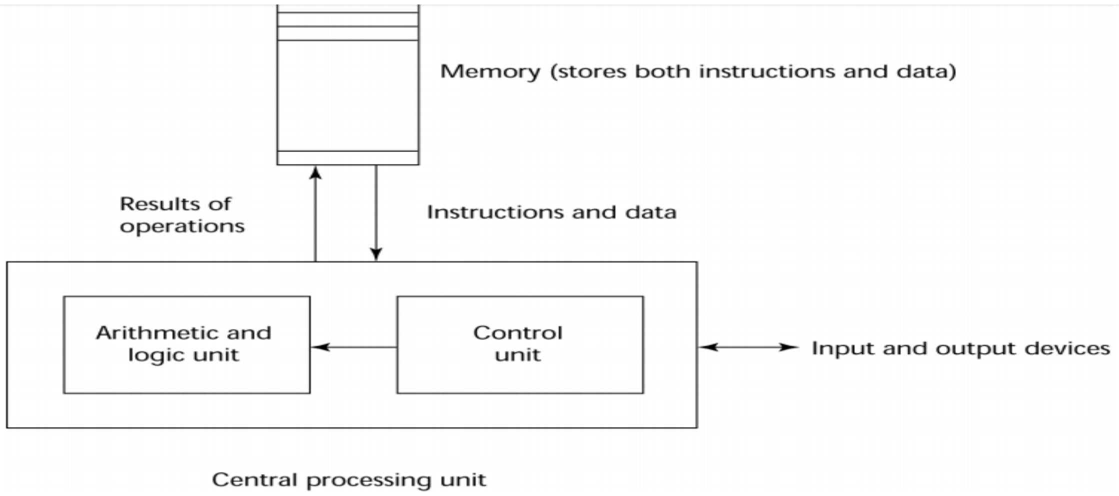
1. Is the computer smart?
    a. No. understands 1s and 0s
    b. Power of computing comes from its fast execution of rather primitive calculations
2. Von nuemann architecture
    a. Most well known and widely used computer architecture
    b. Data and programs are stored in memory
    c. Memory is separate from cpu
    d. Instructions and data are piped from memory to cpu
    e. Basis for imperative languages
        i. Variables model memory cells
        ii. Assignment statements model piping
        iii. Iteration is efficient



3. Numbers
    a. Bit- basic unit of computer storage
        i. It can contain one of two values: 0 or 1
        ii. All other storage in a computer is based on collection of bits
        iii. Given enough bits, it is amazing how many things a computer can render
            1. Numbers, letters, images, movies, sounds documents
    b. Byte: collection of 8 bits
        i. Bit is foundational, but it is too small for actual efficiency. So cue the byte
        ii. Tradiationally, one byte to store one character or symbol.
        iii. Also, used to represent size of disk or memory
            1. Bytes can be further grouped as:
                a. Kb 1024 bytes
                b. Mb 1024 kb
                c. Gb 1024 mb
                d. Tb 1024 gb
    c. Word: the given computers architecture's native storage unit.
        i. Generally made up of one or more bytes\
        ii. In 32 bit processor, a word is 32 bits 4 bytes.
        iii. In 64 bit, word is 64 bits, and 8 bytes.
4. Present-day computers actually use a variety of storage technologies. Each technology is geared toward a specific function, with speeds and capacities to match.
    a. These technologies are:
        i. CPU registers
        ii. Cache memory

        iii. RAM
        iv. Hard drives
        v. Off-line backup storage (tape, optical disk, etc.)

5. In terms of capabilities and cost, these technologies form a spectrum. For example, CPU registers are:
    - Very fast (access times of a few nanoseconds)
    - Low capacity (usually less than 200 bytes)
    - Very limited expansion capabilities (a change in CPU architecture would be required)
    - Expensive (more than one dollar/byte)
6. However, at the other end of the spectrum, off-line backup storage is:
    - Very slow (access times may be measured in days, if the backup media must be shipped long distances)
    - Very high capacity (10s - 100s of gigabytes)
    - Essentially unlimited expansion capabilities (limited only by the floorspace needed to house the backup media)
    - Very inexpensive (fractional cents/byte)
7. Registers:
    a. The number of CPU registers (and their uses) are strictly dependent on the architectural design of the CPU itself.
    b. There is no way to change the number of CPU registers, short of migrating to a CPU with a different architecture.
        i. For these reasons, the number of CPU registers can be considered a constant, as they are changeable only with great pain and expense.
8. Cache:
    a. Acts as a buffer between the very limited, very high-speed CPU registers and the relatively slower and much larger main system memory — usually referred to as RAM
    b. Cache memory config'd s.t whenever data is to be read from RAM, the system hardware first checks to determine if the desired data is in cache.
        i. If the data is in cache, it is quickly retrieved, and used by the CPU.
        ii. However, if the data is not in cache, the data is read from RAM and, while being transferred to the CPU, is also placed in cache (in case it is needed again later).
    c. Cache- where you store the most recently used items, in the hope that it will be used again.
9. Main Memory:
    a. RAM makes up the bulk of electronic storage on present-day computers.
    b. It is used as storage for both data and programs while those data and programs are in use.
    c. The speed of RAM in most systems today lies between the speed of cache memory and that of hard drives, and is much closer to the former than the latter.
10. Hard Drives:
    a. All the technologies discussed so far are *volatile* in nature. In other words, data contained in volatile storage is lost when the power is turned off.
    b. Hard drives, on the other hand, are *non-volatile* — the data they contain remains there, even after the power is removed.
        i. Because of this, hard drives occupy a special place in the storage spectrum. Their non-volatile nature makes them ideal for storing programs and data for longer-term use.

ii. Another unique aspect to hard drives is that, unlike RAM and cache memory, it is not possible to execute programs directly when they are stored on hard drives; instead, they must first be read into RAM.

11. Off-Line backup storage:
    a. Off-line backup storage takes a step beyond hard drive storage in terms of capacity (higher) and speed (slower).
        i. Store as much as you want, as long as you can get that much space.
    b. Tech: magnetic tape, optical disk.

12. History:

Who invented computers?
– Contribution from many inventors – A computer is a complex piece of machinery made up of many parts, each of which can be considered a separate invention.

In 1936, Konrad Zuse made a mechanical calculator using three basic elements:
a control, a memory, and a calculator for the arithmetic and called it Z1, the first binary computer
– First freely programmable computer – Konrad Zuse wrote the first algorithmic programming language called 'Plankalkül' in 1946, which he used to program his computers
– He wrote the world's first chess-playing program using Plankalkül

2. ABC computer- made up of a bunch of lightbulbs, and other mechanisms, Several innovations in computing, including a binary system of arithmetic, parallel processing, and a separation of memory and computing functions.
it could calculate about one operation every 15 seconds, today a computer can calculate 150 billion operations in 15 seconds

3. Eniac I(Electrical Numerical Integrator And Calculator)- The ENIAC contained 17,468 vacuum tubes, along with 70,000 resistors, 10,000 capacitors, 1,500 relays, 6,000 manual switches and 5 million soldered joints

4. 1948 transistor invented: Semiconductor device used to amplify and switch electronic signals – The transistor is the key active component in practically all modern electronics, and is considered by many to be one of the greatest inventions of the twentieth century

5. 1958: chip invented: the integrated circuit. – Major breakthrough in the technology and business worlds – Allowed companies to perform multiple tasks at the same time on a single machine

6. ARPAnet: origin of internet, Packet-switching development – ARPA introduces network for defense and develops e-mail and US universities join network in 1970

7. 1971- The first microprocessor, In 1972, Intel introduces world’s first 8-bit microprocessor, Intel 8008

8. In 1983, The first home computer with a GUI, graphical user interface, was developed by Apple. In 1985, Microsoft begins the friendly war with Apple with its launch of Microsoft Windows operating system

9. • In 1993, “Mosaic” by Marc Andreessen, the first properly developed web-browser, takes Internet by storm. In 1994, Netscape changed the picture of computing industry

Finally the era of the Internet arrived – In a 2007 PC World column, the original Netscape Navigator was considered the "best tech product of all time" due to its impact on the Internet

10. 1995-2005 : Internet and the 32-bit era 11. 2007 ~ 2014: Smart devices and Cloud computing 12. We live in the age of “embedded computer” world now

1. What does a computer do?
    - Fundamentally- preforms billions of calculations per second
    - Remembers the results- (100s of gb's of storage)
    - What kinds of calculations?
        - Built into the language!
        - Defined by the programmer
        - COMPUTERS KNOW ONLY WHAT YOU TELL THEM!
2. Types of Knowledge:
    - Declarative knowledge: Statements of fact
        - The room is yellow
    - Imperative knowledge: A recipe or "How to" list
        - Buy yellow paint
        - Buy paint brush
        - Open paint can
        - Dip brush in
        - Paint walls yellow.
    - The number example- guessing the square root of a number
        - Sqrt(x)= y such that y*y=x
        - Recipe:
            - Start with a **guess**, g
            - **If** g*g is close enough to x, stop and say g is the answer
            - **Else** get a new g: by averaging g and x/g
            - Use the new guess to **repeat** the process until close enough.
    - So: What is a recipe?
            - Sequence of simple **steps.**
            - **Flow of control** process
                - Tells you which step to execute next
                - When each step is executed
            - A way of knowing **when to stop**
        - The above three bullet points—combined: algorithm
    - How can we automate the recipe?
        - **Fixed program Computer:**
            - Calculator
        - **Stored program Computer:**
            - Machine is able to **store** and **execute** instructions.
            - We will focus on this! It is far more interesting than a fixed program computer
    - Stored Program Computer:
        - A set of instructions are **stored** inside the computer
            - These instructions are derived from a predefined set of primitive instructions
                - Arithmetic and logic (+ - / * > < <= >= …)
                - Simple tests (if …. Else ….)
                - Moving data
            - A special program (**the interpreter**) **executes each instruction in order**
                - Uses the tests (refer to above) to change the flow of control through the sequence
                - Stop the program when done.
3. How to make the recipe:
    - A programming language has a set of **primitive operations:**
        - Assign a value to a variable
        - Calling a function (will discuss later)

        - Performing arithmetic operation
            - + - * / %
        - And more… discuss later
    - Expressions:
        - Complex but legal combinations of the above primitives in the given programming language.
    - Using the above two, you use : expressions and computation to get **values**
4. Comparing Programming to English:
    - **Primitive constructs**: the basic building blocks
        - English: words
        - Programming language: numbers, strings, simple operators
    - **Syntax:**
        - English:
            - "Cat dog boy" -> Not syntactically valid
                - Noun noun noun
            - "Cat hugs boy" -> syntactically valid
                - Noun verb noun
        - Programming Language:
            - "hi"5 -> not syntactically valid
                - String number
                - dataType dataType
            - 3.2*5 -> syntactically valid
                - dataType operation dataType
    - **Static Semantics:** which syntactically valid (refer to above) strings have meaning!
        - English:
            - "I are hungry" -> syntactically valid, but static sematic error
        - Programming language:
            - 3 + "hi" -> static semantic error, even if syntactically correct (dataType operation dataType)
            - 3.2 *5 -> syntactically valid and semantically valid
    - **Semantics:** is the meaning associated with a syntactically correct string of symbols with no static semantic errors
        - Given that your sentence/expression is correct (with no syntax or static semantic errors) what does your sentence actually mean?
            - English example:
                - "Flying planes can be dangerous"
            - Programming languages:
                - Have only one meaning but may not be what the programmer intended!
5. Catching errors:
    - Syntactic errors:
        - Common, and easily caught
        - Your interpreter will catch these errors quickly. Program will not run.
    - Static semantic errors:
        - Some languages check for these before running program
        - Can cause unpredictable behavior
            - If it starts to run, and then realizes it can't carry the static semantic error expression.
    - No semantic errors: BUT **different meaning than what programmer intended!**
        - Program runs forever
        - Program crashes
        - Program outputs an answer, but is different from what is expected.
6. Python:

- Defines a program as a sequence of definitions (assigning values to variables) and commands
  - The definitions are evaluated
  - The commands are executed by Python interpreter in a shell (IDLE)
- Commands- instruct the interpreter to do something
- Can be typed directly into a shell, or stored in a file that is read into the shell and evaluated

7. Objects:
   - Programs manipulate **data objects**
   - Objects have a type that defines the kinds of things programs can do to them
     - Han solo is a human, so he can walk, speak English, etc.
     - Chewbacca is a wookie so he can walk, speak “mwraaaarh”, etc
   - Objects are:
     - Scalar: cannot be broken into further smaller pieces
     - Non-scaler: has an internal structure that can be accessed
       - Smaller pieces that make up the larger non-scalar object
       - These smaller pieces can be accessed
8. Scalar Objects: ***** You can use type() to see the type of an object
   - Int – represent an integer: 5
     - Type(5)
     - int
   - Float- represent real numbers: 3.27
   - Bool – represent Boolean values: True, False
   - NoneType- special and has one value: None
9. Type Conversion: Cast
   - Allows you to convert object of one type to another
     - Float(3)- converts the integer 3 to the float 3.0
     - Int(3.9) truncates float 3.9 to integer 3
10. Printing to Console:
    - To show the output fromm the code to a user, use the print command:
      - Print(3+2)
      - 5
    - If you do not use print:
      - 3+2
      - Out: 5
      - It shows you the answer, only in the shell! Not actually to the user to see the output.
11. Expressions:
    - Combination of objects and operators to form expressions
      - An expression has a value, which has a type
    - Syntax for a simple expression:
      - <object> <operator> <object>

Bibliography:

1. Ana Bell, Eric Grimson, and John Guttag. *6.0001 Introduction to Computer Science and Programming in Python.* Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.

2. Jin Woo Kim, *CSCI 375 Operating Systems.* Fall 2017. John Jay College of Criminal Justice. http://jjcweb.jjay.cuny.edu/jwkim/class/mat375-fall-17/lec-375-01.pdf

3. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Introduction_to_System_Administration/s1-memory-spectrum.html