Memory Hierarchy :

- Secondary storage versus main memory (RAM) versus cache
- Big data: streaming versus non-streaming

Compiled versus Interpreted Python is an interpreted language. We can interactively execute Python code in the REPL (Read-Eval-Print-Loop).

Names and variables

- Variables refer to python objects.
- Variables have scope.
- Variable names should be descriptive and follow a set of restrictions (can contain letters, numbers, and '_'; cannot start with a number; cannot be a Python reserved keyword)
- Variable names have conventions. Constants are all capital. Variables start with a lowercase letter, and words are separated by '_'

Strings

- String literals can be defined with ' ' or " "
- The escape character is \, which is used to encode meanings like tabs and newlines
- Printing a string displays it, which interprets all special characters (newline, tabs); a string literal shows the escaped characters without interpreting them.
- We can convert other objects into a string by using the `str()` function.
- Strings are immutable, but we can operate on strings in many ways:
  - Combine two strings: `+`
  - Repeat a string: `*`
  - Format a string ``''{}''.format(...)``
  - Slice strings `[start:end]`
  - `split` strings into a list based on a delimiter
  - `join` a list of strings together with a common delimiter
  - `find` substrings within a string

Conditionals, booleans Objects can be compared for

- Equality: `==`, `!=`
- Order (some objects): `<, >, <=, >=`
- Identity: `is`
- Membership: `in`
- In a conditional context, anything that is not one of `False, None, 0, '', []`, is `True`
- `if`, `elif`, and `else` Selectively execute regions of code
- Indentation defines scope

Loops

- `while <condition>`
- `for <variable> in <iterable>` where *iterable* might be a `range`, `list`

Functions and Modules Functions and modules provide ways to organize and isolate code and operations logically while also providing a means for abstraction.

- Use `def` to define a function. Functions have parameters. Calling a function, these parameters are bound to arguments.
- Function can `return` a value (they act like a mathematical function) or not (we call them for their side-effects)
- We `import` modules so functions in the module are available in our namespace
- Using `if name = '__main__':` creates code block that is not executed when the file is imported as a module

Lists: Operations inlude

- Index: `lst[i]`
- Slice: `lst[i:j]`
- Calculate the length: `len(lst)`
- Add things to lists: `lst.append(obj), lst.insert(i, obj)`
- Remove things: `lst.pop(), lst.remove(obj), del lst[i]`
- Sort lists: `sorted(lst)` or `lst.sort()`

Search

- Linearly searching through a list must examine every element
- Binary search on sorted lists cuts the search space in half each step

Files

- Open files `with open(...) as f:`
- Files objects are iterable

Classes Classes let us define new data types and specify their behavior

- `self.variable` is how we define and access member variables. Member variables let us maintain an object's internal state
- Using `self`, we can access and modify the object's internal state
- The `__init__(self)` method is automatically called when we create an object This lets us initialize an object's internal state. The `self` parameter is the object we are creating.
- We can use the `dot notation` to:
  - Call a method on an object: `obj.method()`, `pt1.distance(pt2)`
  - Access an object's member variables: `obj.variable`, `pt.x = 3`, `chart.slices.pop()`

CSV

- readers
- writers