Python Operators

- Arithmetic Operators
    - % – Returns the remainder weh the left operand is divided by the right operand
    - // - Return the integer division result, discarding the fractional part
- Bitwise Operators
    - & (AND) – Performs a bitwise AND operation between bits of two integers
    - | (OR) - Performs a bitwise AND operation between bits of two integers
    - ^ (XOR) - Performs a bitwise AND operation between bits of two integers
    - ~ (NOT) – Flips the bits of an integer
    - << (Left Shift) – Shifts the bits of an integer to the left by a specified number of positions
    - >> (Right Shift) Shifts the bits of an integer to the right by a specified number of positions
- Logical Operators
    - And – Returns true if both operands are true
    - Or – Returns true if at least one of the operands is true
    - Not – Returns the opposite Boolean value of the operand

Python Functions

- **Assert** – A keyword to check if a condition is true. If the condition is false, it raises an AssertionError
- **Enumerate** – A function used to iterate over an iterable while keeping track of the index and corresponding value
- **Filter** – A function that filters elements from an iterable based on a provided function or condition, returning the elements that satisfy the condition
- **Functools.reduce** – A function that applies a binary function cumulatively to the elements of an iterable, reducing it to a single value
- **Issubclass** – Checks if a class is a subclass of another class
- **Iter** – Built-in function that creates an iterator object from an iterable, allowing you to iterate over its elements
- **Itertools.count** – A function from the itertools module that generates an infinite sequence of numbers starting from a specified value
- **Map** – A function used to apply a given function to each item in an iterable and return an iterator of the results
- **Next** – A function used to retrieve the next item from an iterator
- **Sorted** – An function used to sort elements in an iterable and return them as a list
- **Try** – A keyword used to create exception handling blocks
- **Type** – A built-in function used to determine the data type of an object
- **Yield** – A keyward used in generator functions to yield a value to the caller, allowing the fucnction to be paused and resumed
- **Zip** – A function used to combine multiple iterables element-wise into tuples

.

Python Methods

- **Getitem** – A special method used to define custom behavior for indexing elements of an object
- **Init** – A special method used to initialize objects of a class
- **Iter** – A special method used to define custom iteration behavior for objects
- **Len** – A special method used to define the length of objects
- **Next** – A special method used to define custom behavior for retrieving the next element during iteration

Python Concepts

- **Classes** – Classes are a fundamental concept in OOP and are used to create objects with attributes and methods
- **Exceptions** – Events that occur during program execution that disrupt the normal flow of code
- **Iterables** – Objects that can be iterated over, such as list, tuples, and strings
  - 3 categories – Indexable, Nonindexable, Generators/Iterators
- **Iterators** – Objects that represent a stream of data and allow you to iterate through it one element at a time
  - Lazy and exhaustible
- **Generators** – A way to create iterators in Python, allowing you to iterate over a potentially large sequence of values without generating them all at once

Function Arguments and Parameters

- Arguments and Parameters
  - Asterisks (*) in the parameters packs the arguments into a tuple
  - Asterisks (*) in the arguments unpacks the tuple into arguments
  - Double asterisks (**) in the parameters packs the arguments into a dictionary
  - Double asterisks (**) in the arguments unpacks the dictionary into arguments
- Syntax Rules
  - Position => Name => Unpacking Iterables => Unpacking the Dictionary
- Semantic Rules
  - Position/Unpacking Iterables => Variable Name => Unpacking Dictionary