Chapter-based glossary compiled from *Think Python* by Allen B Downey

Chapter 1: The way of the program.

problem solving: The process of formulating a problem, finding a solution, and expressing it.

high-level language: A programming language like Python that is designed to be easy for humans to read and write.

low-level language: A programming language that is designed to be easy for a computer to run; also called ``machine language'' or ``assembly language''.

portability: A property of a program that can run on more than one kind of computer.

interpreter: A program that reads another program and executes it

prompt: Characters displayed by the interpreter to indicate that it is ready to take input from the user.

program: A set of instructions that specifies a computation.

print statement: An instruction that causes the Python interpreter to display a value on the screen.

operator: A special symbol that represents a simple computation like addition, multiplication, or string concatenation.

value: One of the basic units of data, like a number or string, that a program manipulates.

type: A category of values. The types we have seen so far are integers (type `int`}), floating-point numbers (type `float`), and strings (type `str`).

integer: A type that represents whole numbers.

floating-point: A type that represents numbers with fractional parts.

string: A type that represents sequences of characters.

natural language: Any one of the languages that people speak that evolved naturally.

formal language: Any one of the languages that people have designed for specific purposes, such as representing mathematical ideas or computer programs; all programming languages are formal languages.

token: One of the basic elements of the syntactic structure ofa program, analogous to a word in a natural language.

syntax: The rules that govern the structure of a program.

parse: To examine a program and analyze the syntactic structure.

bug: An error in a program.

debugging: The process of finding and correcting bugs.

Chapter 2: Variables, expressions and statements

variable: A name that refers to a value.

assignment: A statement that assigns a value to a variable.

state diagram: A graphical representation of a set of variables and the values they refer to.

keyword: A reserved word that is used to parse a program; you cannot use keywords like `if`, `def`, and `while` as variable names.

operand: One of the values on which an operator operates. expression: A combination of variables, operators, and values that represents a single result.

evaluate: To simplify an expression by performing the operations in order to yield a single value.

statement: A section of code that represents a command or action. So far, the statements we have seen are assignments and print statements.

execute: To run a statement and do what it says.

interactive mode: A way of using the Python interpreter by typing code at the prompt.

script mode: A way of using the Python interpreter to read code from a script and run it.

script: A program stored in a file.

order of operations: Rules governing the order in which expressions involving multiple operators and operands are evaluated.

concatenate: To join two operands end-to-end.

comment: Information in a program that is meant for other programmers (or anyone reading the source code) and has no effect on the execution of the program.

syntax error: An error in a program that makes it impossible to parse (and therefore impossible to interpret).

exception: An error that is detected while the program is running.

semantics: The meaning of a program.

semantic error: An error in a program that makes it do something other than what the programmer intended.

Chapter 3: Functions

function: A named sequence of statements that performs some useful operation. Functions may or may not take arguments and may or may not produce a result.

function definition: A statement that creates a new function, specifying its name, parameters, and the statements it contains.

function object: A value created by a function definition. The name of the function is a variable that refers to a function object.

header: The first line of a function definition.

body: The sequence of statements inside a function definition.

parameter: A name used inside a function to refer to the value passed as an argument.

function call: A statement that runs a function. It consists of the function name followed by an argument list in parentheses.

argument: A value provided to a function when the function is called. This value is assigned to the corresponding parameter in the function.

local variable: A variable defined inside a function. A local variable can only be used inside its function.

return value: The result of a function. If a function call is used as an expression, the return value is the value of the expression.

fruitful function: A function that returns a value.

void function: A function that always returns `None`.

`None`: A special value returned by void functions.

module: A file that contains a collection of related functions and other definitions.

import statement: A statement that reads a module file and creates a module object.

module object: A value created by an {import} statement that provides access to the values defined in a module.

dot notation: The syntax for calling a function in another module by specifying the module name followed by a dot (period) and the function name.

composition: Using an expression as part of a larger expression, or a statement as part of a larger statement.

flow of execution: The order statements run in.

stack diagram: A graphical representation of a stack of functions, their variables, and the values they refer to.

frame: A box in a stack diagram that represents a function call. It contains the local variables and parameters of the function.

traceback: A list of the functions that are executing, printed when an exception occurs.

Chapter 4: Case study – interface design

method: A function that is associated with an object and called using dot notation.

loop: A part of a program that can run repeatedly.

encapsulation: The process of transforming a sequence of statements into a function definition.

generalization: The process of replacing something unnecessarily specific (like a number) with something appropriately general (like a variable or parameter).

keyword argument: An argument that includes the name of the parameter as a ``keyword''.

interface: A description of how to use a function, including the name and descriptions of the arguments and return value.

refactoring: The process of modifying a working program to improve function interfaces and other qualities of the code.

development plan: A process for writing programs.

docstring: A string that appears at the top of a function definition to document the function's interface.

precondition: A requirement that should be satisfied by the caller before a function starts.

postcondition: A requirement that should be satisfied by the function before it ends.

Chapter 5: Conditionals and recursions

floor division: An operator, denoted `//`, that divides two numbers and rounds down (toward negative infinity) to an integer.

modulus operator: An operator, denoted with a percent sign `%`, that works on integers and returns the remainder when one number is divided by another.

Boolean expression: An expression whose value is either `True` or `False`.

relational operator: One of the operators that compares its operands: `==`, `!=`, `>`, `<`, `>=`, and `<=`.

logical operator: One of the operators that combines Boolean expressions: `and`, `or`, and `not`.

conditional statement: A statement that controls the flow of execution depending on some condition.

condition: The Boolean expression in a conditional statement that determines which branch runs.

compound statement: A statement that consists of a header and a body. The header ends with a colon (:). The body is indented relative to the header.

branch: One of the alternative sequences of statements in a conditional statement.

chained conditional: A conditional statement with a series of alternative branches.

nested conditional: A conditional statement that appears in one of the branches of another conditional statement.

return statement: A statement that causes a function to end immediately and return to the caller.

recursion: The process of calling the function that is currently executing.

base case: A conditional branch in a recursive function that does not make a recursive call.

infinite recursion: A recursion that doesn't have a base case, or never reaches it. Eventually, an infinite recursion causes a runtime error.

Chapter 6: Fruitful functions

temporary variable: A variable used to store an intermediate value in a complex calculation.

dead code: Part of a program that can never run, often because it appears after a {return} statement.

incremental development: A program development plan intended to avoid debugging by adding and testing only a small amount of code at a time.

scaffolding: Code that is used during program development but is not part of the final version.

guardian: A programming pattern that uses a conditional statement to check for and handle circumstances that might cause an error.

Chapter 7: Iteration

reassignment: Assigning a new value to a variable that already exists.

update: An assignment where the new value of the variable depends on the old.

initialization: An assignment that gives an initial value to a variable that will be updated.

increment: An update that increases the value of a variable (often by one).

decrement: An update that decreases the value of a variable.

iteration: Repeated execution of a set of statements using either a recursive function call or a loop.

infinite loop: A loop in which the terminating condition is never satisfied.

algorithm: A general process for solving a category of problems.

Chapter 8: Strings

object: Something a variable can refer to. For now, you can use ``object'' and ``value'' interchangeably.

sequence: An ordered collection of values where each value is identified by an integer index.

item: One of the values in a sequence.

index: An integer value used to select an item in a sequence, such as a character in a string. In Python indices start from 0.

slice: A part of a string specified by a range of indices.

empty string: A string with no characters and length 0, represented by two quotation marks.

immutable: The property of a sequence whose items cannot be changed.

traverse: To iterate through the items in a sequence, performing a similar operation on each.

search: A pattern of traversal that stops when it finds what it is looking for.

counter: A variable used to count something, usually initialized to zero and then incremented.

invocation: A statement that calls a method.

optional argument: A function or method argument that is not required.

Chapter 9: Case study – Word play

file object: A value that represents an open file.

reduction to a previously solved problem: A way of solving a problem by expressing it as an instance of a previously solved problem.

special case: A test case that is atypical or non-obvious (and less likely to be handled correctly).

Chapter 10: Lists

list: A sequence of values.

element: One of the values in a list (or other sequence), also called items.

nested list: A list that is an element of another list.

accumulator: A variable used in a loop to add up or accumulate a result.

augmented assignment: A statement that updates the value of a variable using an operator like +=.

reduce: A processing pattern that traverses a sequence and accumulates the elements into a single result.

map: A processing pattern that traverses a sequence and performs an operation on each element.

filter: A processing pattern that traverses a list and selects the elements that satisfy some criterion.

object: Something a variable can refer to. An object has a type and a value.

equivalent: Having the same value.

identical: Being the same object (which implies equivalence).

reference: The association between a variable and its value.

aliasing: A circumstance where two or more variables refer to the same object.

delimiter: A character or string used to indicate where a string should be split.

Chapter 11: Dictionaries

mapping: A relationship in which each element of one set corresponds to an element of another set.

dictionary: A mapping from keys to their corresponding values.

key-value pair: The representation of the mapping from a key to a value.

item:	In a dictionary, another name for a key-value pair.
key:	An object that appears in a dictionary as the first part of a key-value pair.
value:	An object that appears in a dictionary as the second part of a key-value pair. This is more specific than our previous use of the word ``value''.
implementation:	A way of performing a computation.
hashtable:	The algorithm used to implement Python dictionaries.
hash function:	A function used by a hashtable to compute the location for a key.
hashable:	A type that has a hash function. Immutable types like integers, floats and strings are hashable; mutable types like lists and dictionaries are not.
lookup:	A dictionary operation that takes a key and finds the corresponding value.
reverse lookup:	A dictionary operation that takes a value and finds one or more keys that map to it.
raise statement:	A statement that (deliberately) raises an exception.
singleton:	A list (or other sequence) with a single element.
call graph:	A diagram that shows every frame created during the execution of a program, with an arrow from each caller to each callee.
memo:	A computed value stored to avoid unnecessary future computation.
global variable:	A variable defined outside a function. Global variables can be accessed from any function.
global statement:	A statement that declares a variable name global.
flag:	A Boolean variable used to indicate whether a condition is true.
declaration:	A statement like {global} that tells the interpreter something about a variable.

Chapter 12: Tuples

tuple: An immutable sequence of elements.

tuple assignment: An assignment with a sequence on the right side and a tuple of variables on the left. The right side is evaluated and then its elements are assigned to the variables on the left.

gather: An operation that collects multiple arguments into a tuple.

scatter: An operation that makes a sequence behave like multiple arguments.

zip object: The result of calling a built-in function {zip}; an object that iterates through a sequence of tuples.

iterator: An object that can iterate through a sequence, but which does not provide list operators and methods.

data structure: A collection of related values, often organized in lists, dictionaries, tuples, etc.

shape error: An error caused because a value has the wrong shape; that is, the wrong type or size.

Chapter 13: Case study – Data structure selection

deterministic: Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom: Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

default value: The value given to an optional parameter if no argument is provided.

override: To replace a default value with an argument.

benchmarking: The process of choosing between data structures by implementing alternatives and testing them on a sample of the possible inputs.

rubber duck debugging: Debugging by explaining your problem to an inanimate object such as a rubber duck. Articulating the problem can help you solve it, even if the rubber duck doesn't know Python.

Chapter 14: Files

persistent: Pertaining to a program that runs indefinitely and keeps at least some of its data in permanent storage.

format operator: An operator, `%`, that takes a format string and a tuple and generates a string that includes the elements of the tuple formatted as specified by the format string.

format string: A string, used with the format operator, that contains format sequences.

format sequence: A sequence of characters in a format string, like { \%d}, that specifies how a value should be formatted.

text file: A sequence of characters stored in permanent storage like a hard drive.

directory: A named collection of files, also called a folder.

path: A string that identifies a file.

relative path: A path that starts from the current directory.

absolute path: A path that starts from the topmost directory in the file system.

catch: To prevent an exception from terminating a program using the `try` and `except` statements.

database: A file whose contents are organized like a dictionary with keys that correspond to values.

bytes object: An object similar to a string.

shell: A program that allows users to type commands and then executes them by starting other programs.

pipe object: An object that represents a running program, allowing a Python program to run commands and read the results.

Chapter 15: Classes and objects

class: A programmer-defined type. A class definition creates a new class object.

class object: An object that contains information about a programmer-defined type. The class object can be used to create instances of the type.

instance: An object that belongs to a class.

instantiate: To create a new object.

attribute: One of the named values associated with an object.

embedded object: An object that is stored as an attribute of another object.

shallow copy: To copy the contents of an object, including any references to embedded objects; implemented by the `copy` function in the `copy` module.

deep copy: To copy the contents of an object as well as any embedded objects, and any objects embedded in them, and so on; implemented by the `deepcopy` function in the `copy` module.

object diagram: A diagram that shows objects, their attributes, and the values of the attributes.

Chapter 16: Classes and functions

prototype and patch: A development plan that involves writing a rough draft of a program, testing, and correcting errors as they are found.

designed development: A development plan that involves high-level insight into the problem and more planning than incremental development or prototype development.

pure function: A function that does not modify any of the objects it receives as arguments. Most pure functions are fruitful.

modifier: A function that changes one or more of the objects it receives as arguments. Most modifiers are void; that is, they return {None}.

functional programming style: A style of program design in which the majority of functions are pure.

invariant: A condition that should always be true during the execution of a program.

assert statement: A statement that checks a condition and raises an exception if it fails.

Chapter 17: Classes and methods

object-oriented language: A language that provides features, such as programmer-defined types and methods, that facilitate object-oriented programming.

object-oriented programming: A style of programming in which data and the operations that manipulate it are organized into classes and methods.

method: A function that is defined inside a class definition and is invoked on instances of that class.

subject: The object a method is invoked on.

positional argument: An argument that does not include a parameter name, so it is not a keyword argument.

operator overloading: Changing the behavior of an operator like + so it works with a programmer-defined type.

type-based dispatch: A programming pattern that checks the type of an operand and invokes different functions for different types.

polymorphic: Pertaining to a function that can work with more than one type.

information hiding: The principle that the interface provided by an object should not depend on its implementation, in particular the representation of its attributes.

Chapter 18: Inheritance

encode: To represent one set of values using another set of values by constructing a mapping between them.

class attribute: An attribute associated with a class object. Class attributes are defined inside a class definition but outside any method.

instance attribute: An attribute associated with an instance of a class.

veneer: A method or function that provides a different interface to another function without doing much computation.

inheritance: The ability to define a new class that is a modified version of a previously defined class.

parent class: The class from which a child class inherits.

child class: A new class created by inheriting from an existing class; also called a ``subclass''.

IS-A relationship: A relationship between a child class and its parent class.

HAS-A relationship: A relationship between two classes where instances of one class contain references to instances of the other.

dependency: A relationship between two classes where instances of one class use instances of the other class, but do not store them as attributes.

class diagram: A diagram that shows the classes in a program and the relationships between them.

multiplicity: A notation in a class diagram that shows, for a HAS-A relationship, how many references there are to instances of another class.

data encapsulation: A program development plan that involves a prototype using global variables and a final version that makes the global variables into instance attributes.

Chapter 19: The Goodies

conditional expression: An expression that has one of two values, depending on a condition.

list comprehension: An expression with a {for} loop in square brackets that yields a new list.

generator expression: An expression with a {for} loop in parentheses that yields a generator object.

multiset: A mathematical entity that represents a mapping between the elements of a set and the number of times they appear.

factory: A function, usually passed as a parameter, used to create objects.

Chapter 20: Debugging

analysis of algorithms: A way to compare algorithms in terms of their run time and/or space requirements.

machine model: A simplified representation of a computer used to describe algorithms.

worst case: The input that makes a given algorithm run slowest (or require the most space).

leading term: In a polynomial, the term with the highest exponent.

crossover point: The problem size where two algorithms require the same run time or space.

order of growth: A set of functions that all grow in a way considered equivalent for purposes of analysis of algorithms. For example, all functions that grow linearly belong to the same order of growth.

Big-Oh notation: Notation for representing an order of growth; for example, $O(n)$ represents the set of functions that grow linearly.

linear: An algorithm whose run time is proportional to problem size, at least for large problem sizes.

quadratic: An algorithm whose run time is proportional to n^2, where n is a measure of problem size.

search: The problem of locating an element of a collection (like a list or dictionary) or determining that it is not present.

hashtable: A data structure that represents a collection of key-value pairs and performs search in constant time.