Python Vocabulary Terms

Algorithm - a well-defined sequence of steps for solving a particular type of problem, which can be carried out via a computer program.

Argument - a value provided to a function when that function is called. This value is assigned to the corresponding parameter in the function definition.

Assignment - the process of assigning a value to a variable or data structure.

Boolean - a data type with a value of either “True” or “False”.

Boolean Expression - an expression which results in either “True” or “False”; in other words, an expression that returns a boolean.

Bug - an error in a program.

Class - a blueprint for creating objects in object-oriented programming. It defines a set of attributes (data members) and methods (functions) that the objects created from it will have. Created by using the keyword “class”.

Comment - written 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.) A comment is begun with a hash symbol `# like this.` They can also be written on multiple lines within triple quotes

```
'''

like this.

'''
```

Compile - to translate a program written in a high-level language into a low-level language so it can later be executed.

Concatenate - to join two or more objects together, typically strings. Example:

```
“Hello_” + “world” == “Hello_world”
```

Constructor - a special method in a class that is called when an object is created. It initializes the object's attributes. The constructor method has the same name as the class and is defined using the **init**() function.

Control Flow - methods of executing code in a specific order based on different conditions or criteria. Includes branching (conditional statements) and looping.

Data Structure - a container that holds data in a particular way, allowing for efficient storage, access, and modification of data. Examples of data structures include lists, tuples, and dictionaries.

Data Type - a classification of data based on its nature and intended use. Python has several built-in data types, including integers, floats, strings, booleans, and more.

Debug - the process of finding and removing programming errors or bugs. Debugging often involves using tools like print statements or a dedicated debugger to identify issues and fix them.

Debugger - a program or tool (often built into an IDE) that helps identify and resolve errors in a program. Debuggers allow developers to execute code step-by-step, inspect variables and memory, and track program flow, making it easier to identify bugs.

Declaration - a statement that creates a new variable and optionally assigns an initial value to it. In Python, variables can be declared simply by assigning a value to them, without explicitly declaring their type. For example, x = 5 creates a new variable named x and assigns it the value 5.

Dictionary - an ordered and mutable data structure that stores key-value pairs; unique keys mapped to values. Dictionaries are enclosed in curly braces, key-value pairs are separated by commas, and keys are paired with values with a colon. Example:

```
dict1 = {“key1”:5, “key2”:“blue”, “key3”:[list1]}
```

Dictionary Key - an object that appears in a dictionary as the first part of a key-value pair. Must be of an immutable data type, such as strings or numbers. Lists and dictionaries cannot be used as keys, for example. Meanwhile, values can be of any type.

Exception - an event that occurs during the execution of a program that disrupts the normal flow of instructions. Examples include division by zero and trying to access a non-existent file.

Float - short for ‘floating point number’. A data type representing real fractional numbers (numbers with decimals.)

Function - a block of code that performs a specific task and can be called (reused.) Functions help break code into smaller, more manageable pieces. They may or may not take one or more arguments and may or may not return a resulting value. Many functions are built into Python by default. Additional functions are defined by the programmer.

Function Header - the first line of a function definition, specifying its name and parameters. Created by using "def".

Function Parameter - a variable declared in the function definition that receives a value (an argument) when the function is called. Parameters allow different values to be passed in as arguments each time the function is called.

Function Body - the sequence of statements inside a function definition.

Return Value - the value given by a function when it is completed. Created by using “return”.

Function Call - a statement that executes a function. It consists of the function name followed by an argument list.

High-level Language - a programming language that is designed to be easy for humans to read and write. They are typically closer to human language than machine language, and include constructs that abstract away the details of the computer's architecture. Examples include Python, Java, and C#.

IDE - integrated development environment. An application designed for writing, testing, and debugging code. Consists of a source code editor, build automation (to compile and run tests on the code,) and a debugger. A good IDE provides visual aids such as syntax highlighting and other conveniences.

Index - a number used to represent the position of an element in a sequence, such as a string or a list. In Python, indexing starts at 0, so the first element in a sequence has an index of 0, the second has an index of 1, and so on. Negative indexing can also be used, where -1 represents the last element in a sequence, -2 represents the second-to-last element, and so on.

Inheritance - a mechanism in object-oriented programming that allows a new class to be based on an existing class. The new “child” class inherits the attributes and methods of the existing “parent” class, and can then be given additional attributes and methods.

Integer - a data type representing non-decimal numbers.

Interpret - to execute a program in a high-level language by translating it one line at a time.

Iterable - an object capable of returning its members one at a time, permitting it to be iterated over in a for-loop. Includes lists and dictionaries.

Iteration - the process of repeatedly executing a set of statements until a specific condition is met; in other words, the repeated execution of a block of code using either a recursive function call or a loop.

Keywords - reserved words that are used by the compiler to parse a program. Cannot be used as variable names or function names.

Looping - executing a code block repeatedly, either for a set number of iterations or until a certain condition is met. Includes “for” loops and “while” loops.

> **For Loop** - used for iterating over a sequence (string, list, dictionary, etc.) Executes the same block of code for every item in that sequence.
>
> **While Loop** - used to repeatedly execute a block of code as long as a specified condition (the “while condition”) is true.
>
> **Break Statement** - used to exit a loop prematurely with the “break” keyword.
>
> **Infinite Loop** - a loop in which the terminating condition is never satisfied.

List - a data structure representing a sequence of values, each associated with an index. Created with square brackets. Examples:

```
list1 = [1999, 2001, 2008, 2014, 2018]

list2 = [“red”, “blue”, “gold”, “silver”]

list3 = [0, “mushroom”, [list1]]
```

Low-level Language - a programming language designed to be closer to the binary machine language of the computer. They provide more direct access to the hardware and are often used for systems programming or other tasks that require direct manipulation of hardware resources. Examples include assembly language and machine language. “High level” and “low level” is a relative term, and changes over time; languages that were once considered high level are now considered low level languages. C++ is considered a high level language, but it is low level compared to Python, for example.

Method - a function that is associated with an object and called using dot notation. Often used interchangeably with “function.”

Module - a file containing Python definitions and statements, intended to be imported and used in other Python programs. Modules can contain functions, classes, and other definitions, and can be organized into packages to help manage larger programs.

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.

Mutability - if the value of an object can be changed, the object is called mutable. If the value cannot change, the object is called immutable. Lists and dictionaries are mutable. Numbers, strings, and tuples are immutable. When you change the value of a variable assigned to an immutable object, you are actually creating a new object in memory with the new value, rather than modifying the original object.

Operator - special symbols that carry out arithmetic and logical computations on values and variables (operands.)

Arithmetic Operator - symbols used to perform mathematical operations. Includes addition, subtraction, multiplication, division, modulus, exponentiation, and floor division. (+, -, *, /, %, **, //)

Comparison Operator - symbols which compare their operands and return either `True` or `False` (includes ==, !=, >, <, >=, and <=). Also called "relational operator."

Logical Operator - keywords used on conditional statements and returns either `True` or `False` (includes *and*, *or*, and *not*).

Assignment Operator - symbols used to assign values to variables. The simplest is a single equal sign = but also includes incremental operators like += and -=.

Parameter - a name used inside a function to refer to the value passed as an argument. When the function is called, the arguments in the call are substituted for the parameters.

Print - to display a value to the console / terminal.

Program - a set of instructions that specifies a computation.

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.

Refactor - to rewrite a program so that it mostly behaves the same way on the outside, but has been internally improved. These improvements can include stability, performance, or reduction in complexity.

Regular Expressions - a pattern that describes a set of strings. Used to search and manipulate text.

Semantics - the intended meaning of a program.

> **Semantic Error** - an error in a program where the code runs without any syntax errors, but the output is not what the programmer intended due to incorrect logic or incorrect use of variables.

Sequence - an order set; that is, a set of values where each value is identified by an integer index. (Strings and lists are examples of sequences.)

> **Item** - one of the values in a sequence. Also called an element.
>
> **Index** - an integer value used to select an item in a sequence, such as a character in a string.

Slice - a part of a sequence specified by a range of indexes.

Source Code - the code of a program written in a high-level language.

String - a data type representing a sequence of characters.

Syntax - the set of rules (the "grammar") that defines how a Python program will be written and interpreted.

> **Syntax Error** - an error due to incorrect syntax. Detected while the program is running.

Terminal - where the output of the program is displayed. Also called the console or visual display.

Tuple - ordered and immutable data structure (unlike lists and dictionaries, which are mutable.) Created with parentheses. Example:

```
tuple1 = (3, 4)
```

```
tuple2 = (0, 0, 4.9, {dict1})
```

Type Hinting - a feature introduced in Python 3.5 that allows the programmer to indicate the data types of function arguments, return values, and class attributes. This is done by adding annotations to the code, which can be used by IDEs and static analysis tools to detect errors and provide hints to the programmer.

Value - one of the basic units of data, like a number or a string, that a program manipulates.

Variable - a name or label (defined by the programmer) that refers to a value.

> **Global Variable** - a variable defined outside a function. Global variables can be accessed from any function.
>
> **Local Variable** - a variable defined inside a function. A local variable can only be used inside its function.

Version Control - a system that tracks changes to code or documents over time, allowing users to go back to previous versions, compare changes, and collaborate on the same codebase or document. Examples include Git, SVN, and Mercurial.

Commonly Used Keywords

and - logical operator; "if all of these statements are **True**"

as - give an alternate name to a module, function, or class that is being imported. For example, **import numpy as np**. Alternatively, bind a target to the result of an expression. For example, **with open("file.txt") as file:**

assert - for testing and debugging

break - break out of a loop

class - define a class

continue - continue to the next iteration of a loop

def - define a function

elif - for conditional statements, same as "**else if**"

else - for conditional statements

except - used with "**try**." what to do when an exception occurs

False - boolean value, result of comparison operations

for - create a for loop

from - import specific parts of a module

if - make a conditional statement

import - import a module

in - check if a value is present in a list, tuple, etc.

None - represents a null value

not - logical operator; “if this statement is not **True**”

or - logical operator; “if either of these statements are **True**”

pass - a null statement, a statement that will do nothing

return - exit the function and return a value

True - boolean value, result of comparison operations

try - make a “try-except” statement

while - create a while loop

with - used to simplify the use of resources that need to be acquired and released in a certain order. The basic syntax is **with [expression] as [variable]:**. For example, **with open("file.txt") as file:**

Commonly Used Built-in Functions

abs() - return the absolute value of a number. The argument may be an integer or a float.

all() - return `True` if all elements of the iterable are true (or if the iterable is empty.)

any() - return `True` if any element of the iterable is true. If the iterable is empty, return `False`.

enumerate() - return an enumerate object that contains tuples of (index, item) from the specified iterable.

float() - make a float out of a string or number and return it. If no argument is given, `0.0` is returned.

input() - take input from the user, convert that input to a string, and return it. The argument is optional and acts as a prompt for the user.

int() - make an integer out of a string or number and return it. If no argument is given, `0` is returned.

isinstance() - check if an object is an instance of a specified class or of a subclass thereof.

len() - return the length (the number of items) of an object. The argument may be a dictionary, list, string, or other sequence.

max() - return the largest item in an iterable, or the largest of two or more arguments.

min() - return the smallest item in an iterable, or the smallest of two or more arguments.

print() - print the argument to the terminal. Returns `None`

range() - return a sequence of numbers, starting from 0 (by default,) and increment by 1 (by default,) and stop before a specified number. The parameters are start, stop, and step.

> Start - optional. An integer number specifying at which position to start. 0 by default.
>
> Stop - required. An integer number specifying at which position to stop (not inclusive.)
>
> Step - optional. An integer number specifying the incrementation. 1 by default.

reversed() - return a reverse iterator of the specified sequence.

round() - return a float that is a rounded version of the specified number, with the specified number of decimals (optional.)

slice() - return a slice of a sequence.

sorted() - return a sorted list of the specified iterable.

str() - make a string out of the argument and return it.

sum() - sum the items of an iterable from left to right and return the total.

type() - return the data type of the argument.