Keyword: Algorithm
Definition: A step-by-step procedure or set of rules for solving a specific problem or completing a task.

Keyword: Variable
Definition: A named storage location in computer memory that holds a value which can be changed during program execution.

Keyword: Function
Definition: A block of organized, reusable code that performs a specific task.

Keyword: Loop
Definition: A programming construct that repeatedly executes a block of code until a specified condition is met.

Keyword: Conditional Statement
Definition: A statement that performs different actions depending on whether a condition is true or false.

Keyword: Array
Definition: A data structure that stores a collection of elements, each identified by an index or key.

Keyword: String
Definition: A sequence of characters, enclosed in quotes, that represents text or data.

Keyword: Boolean
Definition: A data type that represents one of two possible values: True or False.

Keyword: Iteration
Definition: The process of repeatedly executing a set of statements or a block of code.

Keyword: Recursion
Definition: A programming technique where a function calls itself to solve a problem.

Keyword: Class
Definition: A blueprint for creating objects that define their properties and behaviors.

Keyword: Method
Definition: A function associated with an object or a class.

Keyword: Inheritance
Definition: A mechanism that allows a class to inherit properties and methods from another class.

Keyword: Exception
Definition: An event that occurs during the execution of a program and disrupts the normal flow of instructions.

Keyword: Debugging
Definition: The process of identifying and fixing errors or defects in a program.

Keyword: Documentation
Definition: Written information about a program, including its functionality, usage, and examples.

1. A **language** is a means (and a tool) for expressing and recording thoughts.
2. A **natural language** is a language people use to communicate with each other in everyday life. English, Russian, German, Swahili, and Hindi are examples of natural languages.
3. A **programming language** is a language developed by humans and used to communicate with computers. A programming language has a set of means to instruct a computer what to do and how.
4. A **high-level programming language** is a programming language which operates on a high level of abstraction thereby allowing the developer to ignore the physical details of the computer's hardware, for example the CPU type, memory size and organization, etc. Python, JavaScript, and C/C++ are all examples of high-level programming languages.
5. A **machine language** is a language placed at the lowest level of computer programming. It's a sequence of bits (binary digits usually recognized as **0** and **1**) which directly forces the CPU to execute the desired elementary operations.
6. An **instruction list** (abbreviated to **IL**) is a list of all elementary (atomic) operations which can be executed by a certain CPU. For example, **x86** (used in personal computers) and **arm** (used in mobile devices) processors have different and incompatible instruction lists).
7. A **source code** is a text encoded in any of the programming languages (regardless of the

language's level). Usually, the source code is put inside a text file which resides inside the developer's computer filesystem, while the file name's extension reveals the programming language used to write the code (for example, files with names which ends with `.py` contain Python source code, while the `.cpp` extension marks files which hold C++ (usually pronounced as *see-plus-plus*) source code.

8. Any language (no matter if it's natural or artificial) is constituted by:
    - an **alphabet** understood as a set of symbols used to build words of a certain language (e.g. the Latin alphabet for English, the Cyrillic alphabet for Russian, Kanji for Japanese, and so on).
    - a **lexis**, also known as a **dictionary**, is a set of words the language offers its users (for example, the word "chat" is present both in English and French dictionaries, but its meaning is obviously different).
    - **syntax** is a set of rules used to determine if a certain sequence of words forms a valid sentence.
    - **semantics** is defined as a set of rules which settle whether or not a certain phrase or sentence makes sense in a given language.
9.
10. A source code **cannot be directly executed by a computer**. To make it possible the source code has to be **translated** into a machine code accepted by a target computer and its CPU. This task can be done using two different techniques:
    - **compilation** performed by a one-time translation of the source program; an executable binary file is created in effect – the file can be run at any time without the need to have the source code; the program that performs the above translation is called a **compiler** or **translator**.
    - **interpretation** which involves a dedicated program designed to translate the source program on-the-fly each time it has to be run; the program performing this task is called an **interpreter**; this means that the interpreter is needed whenever the source code has to be executed.
11.

12. A specific programming language is designed to be the object of either compilation or interpretation (this choice imposes certain distinctive features onto the language). For example, **Python** is an **interpreted** programming language, while **C++** is a compiled one.
13. The interpreter and its environment, created and distributed by the **Python Software Foundation** (PSF) is written mostly in the C programming language. It allows Python to be easily ported and migrated to all platforms providing the ability to compile and run C language programs. This is also why the PSF implementation is often referred to as **CPython**. CPython is the most widely used implementation of Python.
14. A **literal** is data whose value is **determined by the literal itself**. Different kinds of data are coded in different ways, enabling Python (and the human reader of the source code) to determine each literal's **type**. Knowing each argument's type is crucial to understand what operations are legal, and what results can be returned.
15. **Integer** (`int` for short) is a type dedicated to storing **integral numbers**, that is, numbers that lack fractional parts. For example, 1 is an integral number and 1.5 is not.
16. Most used integer literals in Python consist of a sequence of **decimal digits**. Such a sequence cannot include white spaces, but can contain any number of _ (underscore) characters. Note: there must not be more than one underscore between two digits, and the underscore must be neither the last nor the first character of the literal. Underscores don't change a literal's value, and their only role is to improve literal readability. Integer literals can be preceded by any number of `-` (minus) or `+` (plus) characters, which are applied to set the literal's sign.
17. Here are some examples of correct integer literals:
    - `1_111` and `1111` encode the same integer value (*one thousand one hundred and eleven*)
    - `-+-3` and `-3` denote the same integer value (*minus three*)
    - `+1` and `1` encode the same integer value (*one*)
18. Integer literals may be written using radices other than 10:

- if a literal starts with either a `0o` or `0O` digraph, it's an **octal** value (note: it must contain octal digits only!)
- `0o10` encodes an integer value equal to *eight*.
- if a literal starts with either a `0x` or `0X` digraph, it's a hexadecimal value (note: letters from **a** to **f** used as hexadecimal digits may be upper- or lower-case)
- `0X11` encodes an integer value equal to *seventeen*.
- if a literal starts with either a `0b` or `0B` digraph, it's a binary value (note: it must contain **0**s and **1**s only!)
- 0b111 encodes an integer value equal to seven.

6. **Floating point** (`float` for short) is a type designed to store **real** numbers (in the mathematical sense), that is, numbers whose decimal expansion is or can be non-zero. Such a class of numbers includes fractions (integers don't).
7. Float literals are distinguished from integers by the fact that they contain a **dot** (`.`) or the letter *e* (lower- or upper-case) or both. If the only digits which exist before or after the dot are zeros, they can be omitted. Like integers, floats can be preceded by any number of `-` and `+` characters, which determine the number's sign. White spaces are not allowed, while underscores are.
8. If the float literal contains the letter *e*, it means that its value is **scaled**, that is, it's multiplied by a **power of 10** while the exponent (which must be an integer!) is placed directly after the letter. A record like this:
9. m**E**n

   is treated as a value equal to:

   m × 10n

   This syntax is called *scientific notation* and is used to denote a number whose absolute value is extremely large (close to infinity) or extremely small (close to zero).

Here are some examples of correct float literals:

- `1.1` – *one and one-tenth*
- `1.0` (`1.` for short) – *one point zero*
- `0.1` (`.1` for short) – *one-tenth*
- `1E1` – *ten point zero*
- `1e-1` – *one-tenth*
- `-1.1E-1` – *minus eleven hundredths*

10. **String** literals are sequences (including empty ones) of characters (digits, letters, punctuation marks, etc.). There are two kinds of string literal:
    - **single-line**, when the string itself begins and ends in the same line of code: these literals are enclosed in a pair of `'` (apostrophe) or `"` (quote) marks.
    - **multi-line**, when the string may extend to more than one line of code: these literals are enclosed in a pair of trigraphs either `"""` or `'''`
    - strings enclosed inside apostrophes can contain quotes, and vice versa.
    - if you need to put an apostrophe inside an apostrophe-limited string, or a quote inside a quote-limited string, you must precede them with the `\` (backslash) sign, which acts as an escape character (a character which changes the meaning of the character that follows it); some of the most used escape sequences are:
        i. `\\` – backslash
        ii. `\'` – apostrophe
        iii. `\"` – quote
        iv. `\n` – newline character
        v. `\r` – carriage return character
        vi. `\t` – horizontal tab character

11. Here are some examples of correct string literals:
    - `"Hello world"`
    - `'Goodbye!'`
    - `''` (an empty string)
    - `"Python's den"`

    - `'Python\'s den'`
    - `"""Two lines"""`
12. **Boolean literals** denote **the only two possible values** used by the *Boolean algebra* – their only acceptable denotations are `True` and `False`.
13. The `None` literal denotes an **empty value** and can be used to indicate that a certain item contains no usable value.
14. A variable is **a named container able to store data**.
15. A variable's name can consist of:
    - **letters** (including non-Latin ones)
    - **digits**
    - **underscores** (_)
16. and **must start with a letter** (note: underscores count as letters). Upper- and lower-case letters are treated as different.
17. Variable names which start with underscores play a specific role in Python – don't use them unless you know what you're doing.
18. Variable names are not limited in length.
19. The name of the variable must not be any of Python's **keywords** (also known as **reserved words**). The complete list of Python 3.8 keywords looks as follows:
20. A **list** is a data aggregate that contains a certain number (including zero) of elements of any type.
21. Lists are **sequences** – they can be iterated, and the order of the elements is established.
22. Lists are **mutable** – their contents may be changed.

23. Lists can be **initialized** with list literals. For example, these two assignments instantiate two lists – the former is empty, while the latter contains three elements:
24. The list elements can be indexed with **negative numbers**, too. In this case, **-1** accesses the last element of the list, and **-2** accesses the one before the last, and so on. The alternative first list element's index is `-len(list)`.
25. An attempt to access a non-existent list element (when the index goes out of the permissible range) raises the `IndexError` exception.
26. A **slice** is a means by which the programmer can create a new list using a part of the already existing list.
27. The most general slice looks as follows
28. If any of the slice's indices exceeds the allowable range, **no exception is raised**, and the non-existent elements are not taken into consideration. Therefore, it is possible that the resulting slice is an **empty** list.
29. Assigning a list to a list **does not copy elements**. Such an assignment results in a situation when more than one name identifies the same data aggregate.
30. **Strings**, like lists, are sequences, and in many contexts they behave like lists, especially when they are indexed and sliced or are arguments of the `len()` function.
31. The `in` and `not` in operators can be applied to strings to check if any string is a **part of another string**. An empty string is considered a part of any string, including an empty one.
32. Strings are **immutable** and their contents cannot be changed.
33. A **tuple**, like a list, is a data aggregate that contains a certain number (including zero) of elements of any type. Tuples, like lists, are **sequences**, but they are **immutable**. You're not allowed to change any of the tuple elements, or add a new element, or remove an existing element. Attempting to break this rule will raise the `TypeError` exception.
34. Tuples can be initialized with tuple literals.

35. A **dictionary** is a data aggregate that gathers `pairs of values`. The first element in each pair is called the key, and the second one is called the `value`. Both keys and values can be of any type.
36. Dictionaries are **mutable** but **are not sequences** – the order of pairs is imposed by the order in which the keys are entered into the dictionary.