**Terminology of Programming Languages**

1. Closure

A function which can be passed to another block of code and which carries the context in which it was defined, allowing it to reference variables within that context.

1. Computer program

Set of instructions to a computer which solve a specific problem.

1. Concurrent programming

Programs are designed using interacting processes or threads. Execution is parallelized.

1. Control structure

Allow programmers to dictate which path an algorithm takes through the executing code. If, else, while, do, switch, etc.

1. Declarative programming

Opposite of algorithm. A programmer details what the outcome of the program should be rather than how the code should be done.

1. Dynamic Typing

Type-checking occurs at run time rather than compile time. Any method can be called on a given variable, and exceptions will be thrown for incompatibilities, or alternatively undesired and unexpected behavior may occur.

1. Dynamic programming language

A class of high-level language that performs actions at runtime that are traditionally performed at compile time. Use of 'eval'-like functions which process and run source code during program execution is a dynamic programming approach.

1. Explicit Typing

Types are not inferred by the language. Variables must be declared as the type of data they are expected to hold. Implies static typing.

1. 4th-generation programming language

Often domain or problem specific, fourth generation languages are a further separation from computer-centered programming and allow for more abstract commands than earlier generations.

1. Functional Programming

Functions return something and have no side-effects. Functions are treated as data-structures, meaning that they can return data or other functions. Designed to reduce the time and effort of programming.

1. Garbage Collection

Finds resources which are no longer referenced and automatically deletes them. Frees the programmer from having to explicitly control the persistence of objects in memory.

1. Generic programming

Code is written to be type-agnostic. Allows for modularity and reuse of code.

1. Higher Order Function

Functions which take as arguments or return other functions.

1. Imperative programming

Code is a series of instructions which are executed in order.

1. Implicit Typing

Types of variables are determined by the compiler based on the context that it sees them in. For example, putting 3.5 into a variable would make that variable be a floating point number.

1. Iterative

The practice of moving from place to place with no real home.

1. Lambda calculus

Provides a simple way to define a function so that the foundations of mathematics can be studies. Functions explain the computability of problems.

1. Logic Programming

A form of declarative programming wherein the program is given a set of implications and asked to use a theorem prover to see if they are consistant with a conclusion using backwards chaining.

1. Multithreading

A program is split into multiple threads or processes which are run simultaneously and can share resources and/or interact.

20.   Object-oriented programming

As indicated by the name, the object oriented paradigm is characterized by the use of objects as a data structure. Objects are used to model what in a simple sentence’s description of a problem would be a noun. Objects consist of data fields and methods.

21.   Paradigm

A way of thinking or theory of programming.

22.   Procedural Programming

The procedural programming paradigm derives from structured programming, which is based upon the concept of the function call.

23.   Programming Language

A set of semantic and syntactical rules which define a method of communicating instructions to a compiler or interpreter, which in turn communicates in byte code to the computer the same instructions.

24.   Programming methodology

The analysis, design, and implementation in code of programs. Waterfall, Agile, etc.

25.   Recursion

Where you recurse.

The method of writing a function which calls itself (hopefully conditionally).

26.   Reflection

A computer program can observe an modify its own structure and behavior, such as to view the code behind functions or to view metadata of an executable or dll.

27.   Regular Expression

Also regex or regexp. Concise syntax for representing and matching patterns of strings of text such as characters, words, or patterns of characters.

28.   2nd-generation programming language

A refinement of (a) first-generation language to include logical structures (see structured programming)

29.   Semantics

The meaning of languages, as opposed to their form or syntax. Represent the different ways various programming languages represent the same instructions.

30.   Side-effects

In addition to producing a value, a function or expression modifies some state or interacts with functions outside the block of code which represents it.

31.   Static Typing

Type-checking occurs at compile time instead of run time.

32.   Structured programming

Instead of relying on the goto statement, this method of programming uses sequence, selection, and iteration as juxtaposed or nest-able modules.

33.   Syntax

The specific ordering and arrangements of syntactic terms which differentiate a language’s code from that of other languages.

34.   3rd-generation programming language

A language which has evolved past functionality and logical structure and on to friendliness to the programmer. The computer deals with non-essential details rather than the programmer. High-level language is a synonym.

35.   Turing machine

A read head, a write head, an (infinite) tape on which to write, a means by which to move the tape back and forth, symbols.

36.   Type safety

The extent to which a programming language discourages or prevents type errors. A language is type safe only if it prevents actions on types which are not explicitly allowed by the class representing the type.

37.   Type system

A manageable ordering/arrangement system for classifying objects according to the kinds of values they represent.

38. Algorithm

A specific set of instructions by which a problem can be solved.

39. Assistive Software

Programs designed to assist physically or mentally challenged users. Examples are in the Start->Programs->Accessories->Accessibility folder of the start menu in most modern windows distributions.

40. Antipattern

A prevalent solution to a problem which is inefficient or ineffective as a solution.