**Predefined functions**

A predefined function is a set of routines that the computer carries out to perform functions in a programming language.

An example of a predefined function is: **System.out.println(“Hello World”).**

System.out.print allows you to print out on screen whatever is in the speech marks.

Another example of a predefined function is **concat().**

**Programming Constructs**

The most common techniques that are used in programming are what are known as ‘constructs’. There are three main constructs that programmers use, these are Sequence, Selection and Iteration.

Sequence: a simple definition of a sequence is that it is a set of instructions that are in order, one instruction after the other. Once one instruction has been executed, it will take a few seconds before the next one is executed.

Selection: A selection is when a question is asked and depending on the answer, there may be more than one outcome. To put it in a more simpler way it is also known as an **if statement**.

Iteration: An iteration is a computer loop that goes through a group of instructions which gets executed over and over again. Iteration involves the use of two variables which are called ‘while’ and ‘do while’. An example is a **while loop**.

**Expression**

An expression in a programming language is a combination of one or more explicit values, constants, variables, operators and functions that the programming language interprets and computes to give another value. E.g 5 - 3 = 2, 4 != 4 = false.

**Datatypes in Ruby**

Strings: Usually text placed within either single or double quotes. E.g ‘Hello ‘, “Hello “

Integers: Is a datatype that consists of whole numbers. E.g 25

Float: or floating point number is numbers with decimals, E.g 6.8

Boolean: is a datatype that returns either true or false

Symbols: These are lightweight strings, they are often used as identifiers where other languages

would use Strings. E.g symbol:

**Static Testing**

Static testing is a software testing method that involves the examination of the programs code and its associated documentation but does not require the program to be executed. Static testing may be done manually or by one of many software testing tools. Code analysis, inspection, code reviews and walkthroughs are specific types of static testing.

**Dynamic Testing**

Dynamic testing is a method of assessing the feasibility of a software program by giving input and examining the output. The dynamic method requires that the code be compiled and run.

**Black-box testing**

Black-box testing treats software under test as a black-box without knowing it’s internals. Tests are using software interfaces and trying to ensure they work as expected. As long as functionality of interfaces remain unchanged, tests should pass even if internals are changed. Tester should have knowledge of what the program should do but does not have the knowledge of how it does it.

Some of the **advantages of black-box testing** are:

• Efficient for large segments of code

• Code access is not required

• Separation between user’s and developer’s perspectives

Some of the **disadvantages of black-box testing** are:

• Limited coverage since only a fraction of test scenarios is performed

• Inefficient testing due to tester’s lack of knowledge about software internals

• Blind coverage since tester has limited knowledge about the application

**White-box Testing**

White-box testing looks inside the software that is being tested and uses that knowledge as part of the testing process. White-box testing requires internal knowledge of the system and programming skills. It provides internal perspective of the software under test.

Some of the **advantages of white-box testing** are:

• Efficient in finding errors and problems

• Required knowledge of internals of the software under test is beneficial for thorough testing

• Allows finding hidden errors

• Programmers introspection

• Helps optimizing the code

• Due to required internal knowledge of the software, maximum coverage is obtained

Some of the **disadvantages of white-box testing** are:

• Might not find unimplemented or missing features

• Requires high level knowledge of internals of the software under test

• Requires code access

**Software testing levels**

There are four levels of software testing: Unit >> Integration >> System >> Acceptance

***~~Acceptance Testing~~***

***~~System Testing~~***

***~~Integration Testing~~***

***~~Unit Testing~~***

Unit Testing: A level of the software testing process where individual units/components of a software/system are tested. Its purpose is to validate that each unit of software performs as designed

Integration Testing: A level of the software testing process where units are combined and tested as a group. The purpose of this testing is to expose faults in interaction between integrated units.

System Testing: A level of the software testing process where a complete, integrated software/system is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements.

Acceptance Testing: A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and to asses whether it is acceptable for delivery.

**Testing and Documenting Solutions**

Testing: To test a program, you should test it using normal, extreme and exceptional data. The data that falls into these categories depends on what your program is designed to do.

For example if you designed a program to process students’ test scores out of 50 then normal, extreme and exceptional dat might be as follows;

Test Case

Explanation

Example where score should be between 0 and 50

Normal

Data that you would expect to work or be accepted.

2, 45

Extreme

Data at the upper and lower limits of the range

0, 50

Exceptional

Data that should not be accepted by the program

-7, blah

**Errors**

Writing code can often lead errors while developing or running code. There are three different types of errors that can occur;

Syntax Error: shows that your code has broken a rule about how to use your programming language.

Examples: Misspelling a keyword or missing a character that is needed

You will not normally be able to run a program that has a syntax error. The translator software will keep pointing out the error until you fix it. Once fixed, and as long as there are no other syntax errors, the translator will translate the program.

Execution/Run time error: is an error that occurs when the program is running. For example, asking the program to divide by 0 (unless your in JavaScript of course) or trying to open a text file that can’t be found on the system. The program will be translated successfully but when the code is run, it will crash.

Logic Error: causes a program to operate incorrectly but does not crash it. Examples are:

• sequencing errors, where a line of code is correctly formed but in the wrong place

• errors in calculations, for example subtracting instead of adding

• errors when checking conditions, for example using < instead of ≤

**Event Driven Programming**

Event driven programming is used to structure a program around various events. These events include user input events in graphical user interfaces and networking requests from websites and other online properties. Because users can click virtually anywhere, the program must be ready for nearly any input. In addition, information isn’t always entered in the same manner; event-driven programming makes it easier to handle a diverse range of inputs.

**Good Practice in Programming**

Internal commentary: allows the programmer to add comments to their code that will not be translated when the program is run. These comments can be used to add descriptions, notes or explanations for anyone who has access to the code. Can be useful to help other programmers get used to your code.

Meaningful identifiers: make sure to use variable names that describe what the variable contains. Poor variable names can make it very difficult to know what is going on in your programs. It is always useful to use names that relate to the value that you are going to store. Function names should also be descriptive and give an idea of what the function does.

Indentation: move parts of your code such as loops and if statements to the right to make the overall structure of your code easier to see. Using proper indentation can make your code a lot easier to read.

White Space: is used to help make it clear where code is placed. White space is used to separate different subprograms and functions so it is easy to see each section(module) of code. Using white space makes your code more readable.