1.1.1.1. Programming - absolute Basic.

* How does a computer program work ?
→ This course aims to show you what the Python language is and what it is used. for lets start from the absolute Basic.

* A program makes a computer usable. without a computer, even the most powerful one is nothing more than an object. Similary, without a player, a piano is nothing more than a wooden box.

* It can execute only extemely simple operations e.g. a computer cannot evaluate the value of a complicated mathematical function by itself, although this isn't beyond the realms of possibility in the near future.

Note :-
* accept a number representing the distance.
* accept a number representing the travel time;
* divide the former value by the latter and store the result in the memory
* display the result (representing the average speed) in a readable format.

Language is the keyword.

06

SATURDAY • JUNE

JUNE 2020

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

{1.1.1.2 programming - absolute basic}

Natural language Vs. programming languages

* Machine languages are developed by Humans

* Newwords are created every day and words disappear. These languages are called Natural Languages.

{* 1.1.1.3 Programming - absolute basics *)}

What makes a languages?

* we can say that each languages (machine or natural, it doesn't matter) consist of the following elements.

* an alphabet: a set of symbol used to build words of a certain language (e.g. the latin alphabet for English, ~~the~~ the Cyrillic alphabet for Russian. Kanji for japaneses and so on)

07 SUNDAY

* a lexis: (aka a dictionary) a set of words the languages offers its user (E.g. the word word "chat" is present both in English and French dictionary, but their meaning are different)

* a Syntax: a set of rules (formal or informal, written or felt intuitively used to determine if a certain String of words a valids setence ("I am a python" is a Syntactically correct phrase, while "I am python am" isn't)

* Semantics: a set of rules determining if a certain phases makes sense (Eg. "I ate a doughnut" makes sense, but "A doughnut ate me" doesn't)

→ The IL is, in fact the ~~al~~ alphabet of a machine language. This the simplest and most primary set of Symbols we can use to give Commands to a Computer it's the Computer's mouth tongue.

Imp Note: A program written in a high-level progmming is called. Source code (in contrast to the machine code executed by Computers), similary the file Containing the source code is called the Source file.

09

TUESDAY • JUNE

JUNE

M	T	W	T
1	2	3	4
8	9	10	11
15	16	17	18
22	23	24	25
29	30		

{ 1.1.1.4 programming - absolute basic
{ Compilation Vs. Interpretation

* Compilation vs. Interpretation
→ Computer programming is the act of Composing the Selected programming's languages elements in the order that will Cause the desired effect Could be effect Could be, different in every Specific case - Its up to the programm imagination, knowledge and experience

of Course, Such a Composition has to be Correct in many Senses:

* Alphabetically * lexically * Syntactically
* Semantically

* Alphabetically :- a program needs to be written in a recognized Script, Such as Roman, Cyrillic, etc.

* lexically :- each programming language has its dictionary and you needs to master it; much Simple and Smaller than the dictionary and you need to master it,

* Syntactically :- each language has its rules and must be obeyed
* Semantically :- the program has to makes sense.

There are two different ways of tranforming a program from a high-level programming language into machine language.

* Compilation *

The Source program is translated once by getting a file containing the machine code. now you can distribute the file world wider The program that perform this translation is called a Compiler or translator

* interpretation *

you (or any user of the code) can translate the source program each time it has to be run. the program performing this kind of transformation is called an interpereter as, it interprets the code every time it is intended to be executed also means interpreter to execute it.

Due to some very fundametal resons a particular's high-level programming language is design to fall into one of these two categories.

11

THURSDAY • JUNE

There are very few languages ~~that~~ that can be both compiled and interpreted usually, a programming language is projected with this factor in its ~~facts~~ its constructors' minds - will it be compiled or interpredted?

1.1.1.5 Compilations vs. interpretation

what does the Interpreter actualy do?
→ let assume once more that you have written a program. Now it exists as a Computer file: a Computer program is actually a piece of text, so the Source Code is usaly placed in text file.

it has to be pure text, without any decorations like diffenent fonts, colors embedded images or other media. Now you have to invoke the interpreter and let it read your Source file.

File of all interpreter checks if all Sub Sequent line are Correct (using the four aspects Covered eaiier).

if the Compiler find an error, it finishes its works immediately. The only result in this case is an error message.

The interpreter will inform you where the error is located and what caused it. However, these message may be misleading as the interpreter isn't able to follow your exact intentions and may detect error at some distance from their real causes.

if the line looks goods. the interpreter tries to execute it.

it it also possiable that a Significant part may be executed Succesfully before The interpreter find and error. This is normal hahavior in this execution model.

You may ask now: which is better, The Compilling" model or the " interpreting' model? There is no obvious answer. if there had been, one of thoses model would have ceased to exist a long time ago. Both of them have their advatages and their disadvantages.