Introduction to Computing using Python

(UE18CS101)

Dear friends,

This is an attempt to develop some lecture notes to aid you in “Introduction to Computing using Python. This notes is prepared by **Prof. N S Kumar** (Most of the information in this notes is given by him) **and myself .**

Any constructive feedbacks to make this better are always welcome.

For any further queries you may contact any one of us.

With warm regards,

Chitra G M N S Kumar

[chitragm8@gmail.com](mailto:chitragm8@gmail.com) [kumaradhara@gmail.com](mailto:kumaradhara@gmail.com)

* **What is Computing?**

Computing is a process of completing or accomplishing a task by utilizing the computer technology. Computing may involve computer hardware and or Software, but must involve some form of a computer system.

The task or job can be either Computational or Non-Computational.

**Computational**-The problems that can be solved. Inorder to solve a problem computationally you need representation and Algorithm.

**Non Computational** -The problem that can not be solved.

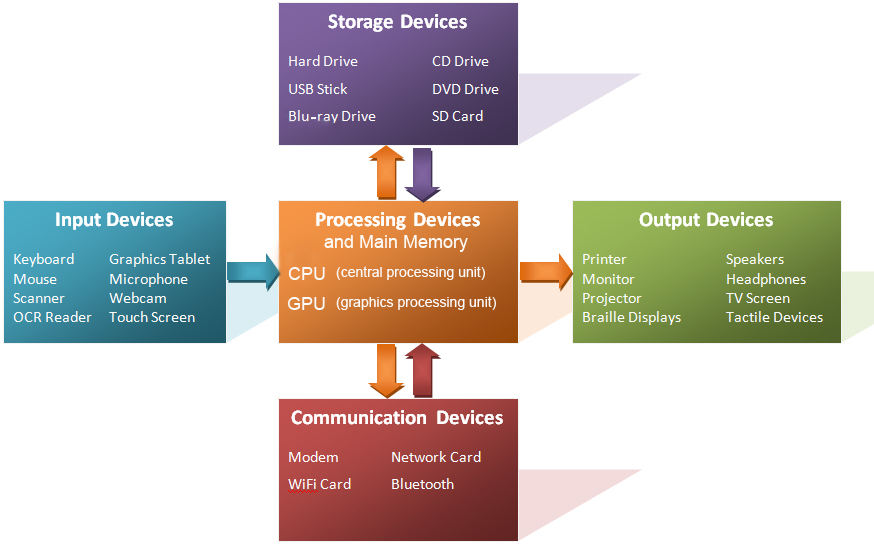
This semester as a part of CS101 course we will be dealing with the problems that can be solved ie,Computational Problem Solving**.**

It is evident that in order to solve a problem we need representation , Algorithm, Hardware and Software(in other words computer).

**Representation**- captures all the relevant aspects of the problem.

**Algorithm**- An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time. The word “algorithm” is derived from the ninth-century Arab mathematician, Al-Khwarizmi.

**Hardware-**Computer hardware comprises the physical part of a computer system. It includes the all-important components of the **central processing unit** (CPU) and **main memory**. It also includes **peripheral components** such as a keyboard, monitor, mouse, printer etc.



**Central processing unit (CPU)** – **the “brain” of a computer system**. Interprets and executes instructions.

**Main memory** – **is where currently executing programs reside**.It is *volatile*, the contents are lost when the power is turned off.

**Secondary memory** – **provides long-term storage of programs and data**. N*onvolatile*, the contents are retained when power is turned off. Can be magnetic (hard drive), optical (CD or DVD), or flash memory (USB drive).

**Input/output devices** – mouse, keyboard, monitor, printer**,** etc.

B**uses** – **transfer data between components within a computer system**. System bus (between CPU and main memory).

**Software:**

Computer software is a set of program instructions, including related data and documentation, that can be executed by computer. This can be in the form of instructions on paper, or in digital form.

While some software are intrinsic to a computer system called as **system software**(System software is general purpose software which is used to operate computer hardware. It provides platform to run application softwares), while other softwares fulfills users’ needs, called as **application software**(Application software is specific purpose software which is used by user for performing specific task).

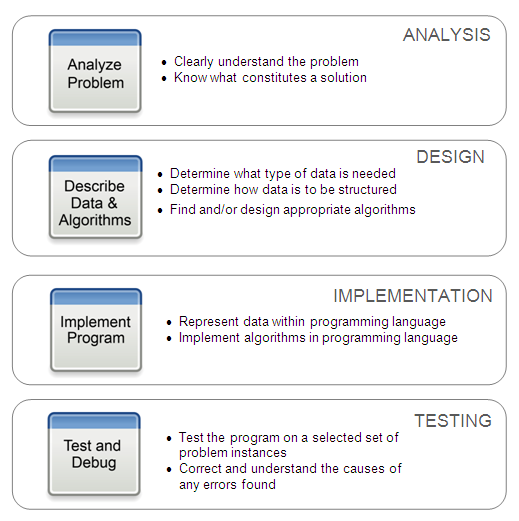
## Difference between System Software and Application Software

|  |  |  |
| --- | --- | --- |
| S.No. | System Software | Application Software |
| 1. | System software is used for operating computer hardware. | Application software is used by user to perform specific task. |
| 2. | System softwares are installed on the computer when operating system is installed. | Application softwares are installed according to user’s requirements. |
| 3. | In general, the user does not interact with system software because it works in the background. | In general, the user interacts with application sofwares. |
| 4. | System software can run independently. It provides platform for running application softwares. | Application software can’t run independently. They can’t run without the presence of system software. |
| 5. | Some examples of system softwares are compiler, assembler, debugger, driver, etc. | Some examples of application softwares are word processor, web browser, media player, etc. |

* **Process of Computational problem solving** :

does not simply involve the act of computer programming. It is a process, with programming being only one of the steps.

Before a program is written, a design for the program must be developed. And before a design can be developed, the problem to be solved must be well understood. Once written, the program must be thoroughly tested.



.

.

* **What is a computer?**

A computer is a device that can be instructed to carry out sequences of [arithmetic](https://en.wikipedia.org/wiki/Arithmetic) or [logical](https://en.wikipedia.org/wiki/Boolean_algebra) operations automatically via [computer programming](https://en.wikipedia.org/wiki/Computer_programming). Modern computers have the ability to follow generalized sets of operations, called [programs](https://en.wikipedia.org/wiki/Computer_program). These programs enable computers to perform an extremely wide range of tasks.

* **How do we communicate with the computer?**

We communicate by giving instruction/commands and we need language(programming) to do so.

* **What is programming?**

Program is a sequence of instructions to solve a computational problem.

“Programming is a human activity” said Dijkstra – one of the greatest computer scientists ever. It is all about thinking. It is a creative activity. No two individuals would not carry an activity the same way.

There is a mythological story of Shiva Parvathi asking their children Ganesha and Skanda to go around the world and come back. Skanda goes on his motorbike (i. e. Peacock) round the world and Ganesha goes round his parents as he considers his parents to represent the world! This Ganesha provides a faster solution whereas my namesake fails (like me!).

Any creative, logical, thinking activity should be fun.

* **Why should we learn program?**

Let us watch the video : [You Should Learn to Program: Christian Genco](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjV8uzPmuvVAhXDrI8KHWGcAcEQtwIIJzAA&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DxfBWk4nw440&usg=AFQjCNFCD9BTmYorZ4Xje1zkeoIYngfGBQ).

* We are considered literate today if and only if we know programming
* We can solve computational problems if and only if we know how to command the computer to do something for us. That requires programming skills.
* We want to sharpen our thinking skills.

**What is a programming language?**

It seems that young twins of my friend started developing their own way of communicating between them. It seems a doctor told my friend and his wife to separate the children lest they refuse to communicate with the parents!

We require a language to communicate. Deaf and dumb use sign language. Musicians have some interesting symbols to record the notes of a song – accepted and agreed upon by other fellow musicians.

We require a language to communicate with our dumb computers. They do have their y6777777777777777777own language – similar to the languages small babies have! That is called the machine language.

We find it very difficult to communicate to the computer in their language. We want to use a language which makes it easier for us to express our ideas. We want a language closer to the domain of application. These sort of languages are called high level languages.

**A programming language provides the necessary constructs to instruct the computer to do something useful.**

**A program is a sequence of instructions in a programming language.**

* **How does a computer understand a program in a high level language?**

It can not directly understand a program in a high level language as it only knows machine language.

We require a mechanism to translate a program in a high level language to a program in a machine language. These are called translators.

\\\\\ii

It is similar to what happens in the real world. A small child talks in a language which is understandable to its mother alone. She has to translate the ‘child talk’ to her friends so that her friends can appreciate the intelligence of the child – does not really matter whether the child is a prodigy!

A translator can convert a program in a high level language completely at one go into a program in the machine language. Or it may convert in small chunks called statements. The former is called a compiler and the latter is called an interpreter. It is also possible to have translators which may convert bigger chunks or convert to some other intermediate form. These are hybrid models.

It is sufficient to realize that we require some translator to make the computer understand programs written in some high level language.

* **which programming language(PL)?**

choose the right language based on the application.

* **What is Python?**

Python is a high level language developed by Guido Van Rossum. It is named after ‘Monty Python circus Show- a satirical show aired on BBC in the 60s and 70s. You may want to watch them on youtube.

[Top 10 Monty Python Movie Moments – YouTube](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwjRqMm3-_PVAhVMuI8KHQzcDZQQtwIIPjAE&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DXCnK3B7Ftow&usg=AFQjCNHqN4nzNvt6BLGLF3V6tncX2MF5cA)

Python is a simple yet powerful language well received by both the academia and the Industry. We will revisit the characteristics of Python later in these lectures.

* **Why learn Python?**

Life is all about selecting between the alternates. You must have debated before selecting Engineering over medicine, selecting college A over college B. I am sure you would have some criteria before choosing between the alternates – Your parents run a hospital, your uncle is a software specialist, your neighbours whom you do not like boast a lot about their child’s achievement or you tossed a coin to decide.

It is equally same here too. We have umpteen number of languages to choose from. After lots of experiments with various languages, we have decided that you should start your journey in programming with Python. Recently Stanford also changed over to Python as the first language to be taught.

We prefer Python for a number of reasons. Three most important reasons are:

* + - simple to learn
    - easy to master
    - extensively used in the industry
    - extensively used in research

**Features of python**

* Uses an elegant syntax, making the programs you write easier to read.
* Is an easy-to-use language that makes it simple to get your program working.
* Comes with a large standard library that supports many common programming tasks such as connecting to web servers, searching text with regular expressions, reading and modifying files.
* Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE.
* Runs anywhere, including Mac OS X, Windows, Linux, and Unix.
* Is free software in two senses. It doesn't cost anything to download or use Python, or to include it in your application. Python can also be freely modified and re-distributed, because while the language is copyrighted it's available under an open source license.
* A variety of basic data types are available: numbers (floating point, complex, and unlimited length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
* Python supports object-oriented programming with classes and multiple inheritance.
* Code can be grouped into modules and packages.
* The language supports raising and catching exceptions, resulting in cleaner error handling.
* Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
* Python contains advanced programming features such as generators and list comprehensions.
* Python's automatic memory management frees you from having to manually allocate and free memory in your code.

**Introduction to python programming:**

There are two ways of making the computer follow the commands of Python.

1. interactive mode

python

>>><here enter commands>

2. batch mode

create a file of commands in python

run them together

python <filename>

In interactive mode, the computer will keep displaying the results immediately. If we give a formula (called expression), it will find the value and display it.

In batch mode, we should explicitly indicate that we want to display something using a special command(function).

**check: 1\_intro.py**

A part of the file is shown for understanding.

3 \* 4 # does not show anything on the screen

# use print for displaying

print(5 \* 6)

# print : said to be a function

# does not give back anything - does not return anything

# it does display

**Program structure:**

We have used a function called print in the earlier program. The function concept is based on the concept of mathematics – example sine, cosine from trigonometry, distance formula given a pair of points. A function takes a number of arguments and gives a result. There are functions which will do something for us without returning any result.

print function takes a number of arguments, finds the value of these arguments and displays them on the computer screen. It does not give a value back.

We will now discuss the program structure.

**Refer to 3\_program\_structure.py**

* A program in python has number of statements. These statements are followed by the computer one after another in a sequence. We call this “executing a program” or “running a program”.
* Python distinguishes uppercase and lowercase characters – like ‘A’ and ‘a’. Most of the words we use in Python are in lower case.

Print("nalku") # gives an error.

**Refer to 2\_errors.py**

* Language has grammar rules called syntax. If the syntax is violated, the Python translator gives an error.

Example 1: A sequence of symbols(characters) is called a string. A string which does not change is referred to as a string constant or string literal. It has to be surrounded by quotes – single or double – should be same in the beginning as well as the end. If quotes are missed at one end or not properly matched, then the translator gives an error.

print('this will be syntax error")

Example 2: All statements unless special construct is used should start from the first column. If the rule is not followed, the translator gives an error.

print('another syntax error')

* If a program has no syntax error, then Python will execute the statements one by one. Sometimes we may get errors at run time. Then the program stops at that point and indicates the error. This is checked only when that point is reached during the program execution.

Example : print(10/0)

* Each statement by default (normally) should be on a line by itself.
  + We can not normally have more than one statement on a single line
  + We can not normally continue the statement on more than one line.
* To continue statement on multiple lines, either we use constructs with delimiters for the beginning and the end or we use special key - \ - to ignore the enter key – we escape the newline.

**Refer to 3\_program\_structure.py**

# 1. use constructs which has beginning and ending markers - like a pair of parentheses

print(

"five"

)

# 2. enter \ (backslash) before pressing <Enter> key - this is called escaping.

print \

(

"six"

)

* We can have multiple statements on a single line by using ; as a separator between statements.

# multiple statements on a single line

print("seven"); print("eight")

* In many languages, the execution starts from a special function which the programmer has to write called main. We have no such concept in Python. The execution starts from the first statement on top.
* A function name itself has a value. Any thing which has a value is called an expression. If we enter the function name on one line, it becomes a statement. But function will not be called unless we also use a pair of parentheses.

print # fn name => expr => stmt; no call !!

("no output") # string within parentheses is an expr ; no action

**concept of constant, variable and the type:**

You may remember that you learn arithmetic before moving onto algebra. In arithmetic, we talk about values which do not change. We find the area of a circle of radius 7 units, 14 units and so on. We then learn algebra(monomial, binomial and so on). We learn the identities like (a + b) ^ 2.

Even in programming, we talk about entities which do not change. They are considered as they are. They are said to be constants or literals.

A literal is a sequence of one or more characters that stands for itself(constants).

Literals can be :

Numeric literal

String Literal

Numeric Literals: Integer

Floating

Complex Numbers

* A numeric literal is a literal containing only the digits 0–9, an optional sign character ( 1 or 2 ), and a possible decimal point.
* If a numeric literal contains a decimal point, then it denotes a floating-point value , or “ float ” (e.g., 10.24); otherwise, it denotes an integer value (e.g., 10). **Commas are never used in numeric literals** .
* Complex numbers have a real and imaginary part.
* There is no limit to the size of an integer that can be represented in Python.
* Floating-point values,however, have both a limited range and a limited precision . Python uses a double-precision standard providing a range of 10 to the power of -308 to 10 to the power of 308 with 16 to 17 digits of precision.
* To denote such a range of values, floating-points can be represented in scientific notation, **9.0045602e15 ,1.006249505236801e8**
* Limitations of floating-point representation.

For example, the multiplication of two values may result in **arithmetic overflow** , a condition that occurs when a calculated result is too large in magnitude (size) to be represented.

* >>> 1.5e200 \* 2.0e210

inf

Note: inf- infinity

Similarly, the division of two numbers may result in arithmetic underflow , a condition that occurs when acalculated result is too small in magnitude to be represented.

* >>> 1.0e-230 / 1.0e100

0.0

**Valid Literals examples:**

>>> 1024 # (valid)

1024

>>> -1024 # (valid)

-1024

>>> .1024 # (valid)

0.1024

>>> 0.1024 # (valid)

0.1024

>>> 1,024.7 # (valid)

(1, 24.7)

>>> 1,024 # invalid

File "<stdin>", line 1

1,024

^

SyntaxError: invalid token

>>> 3+4j #valid

(3+4j)

>>> 3+4i #invalid

File "<stdin>", line 1

3+4i

^

SyntaxError: invalid syntax

**A string litera**l, or string, is a sequence of characters denoted by a pair of matching single or double (and sometimes triple) quotes in Python.

a=”python”

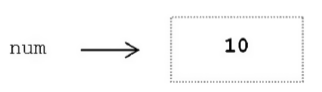
b=”PES University”

**Variable:**

A variable is a name (identifier) that is associated with a value.

A variable is created the moment you first assign a value to it.

There are some reserved words for Python and can not be used as variable name. (Keywords)



A variable can be assigned different values during a program’s execution—hence, the name “variable” .

Variables are assigned values by use of the assignment operator , ‘=’ .

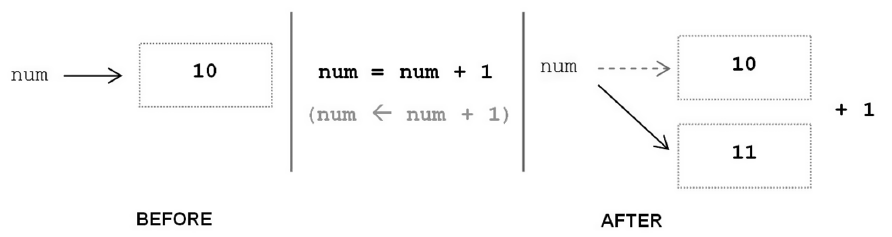
num=10

num=10+2

Assignment statements often look wrong to novice programmers. Mathematically, num =num + 1 does not make sense.

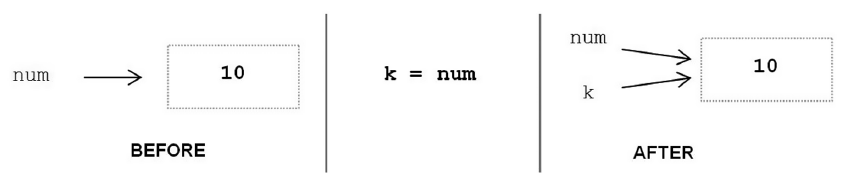
In computing, however, it is used to increment the value of a given variable by one.

It is more appropriate, therefore, to think of the ‘=’ symbol as an arrow symbol, as shown



From the above example it makes clear that the right side of an assignment is evaluated first, then the result is assigned to the variable on the left.

Variables may also be assigned to the value of another variable (or expression)



Variables num and k are both associated with the same literal value 10 in memory. One way to see this is by use of built-in function id,

>>>id(num)

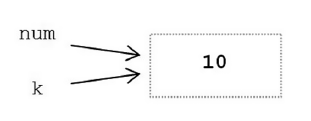
505494040(value may change)

>>>id(k)

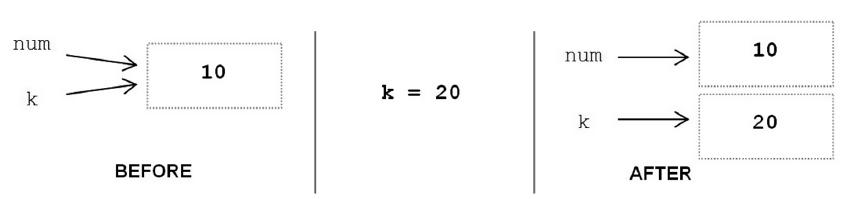
505494040

The id function produces a unique number identifying a specific value (object) in memory. Since variables are meant to be distinct, it would appear that this sharing of values would cause problems.

If the value of num changed, would variable k change along with it?



This cannot happen in this case because the variables refer to integer values, and integer values are immutable. An immutable value is a value that cannot be changed. Thus, both will continue to refer to the same value until one (or both) of them is reassigned,



If no other variable references the memory location of the original value, the memory location is deallocated (that is, it is made available for reuse).

**Identifiers:**

* An identifier is a sequence of one or more characters used to provide a name for a given program element.
* Python is case sensitive.

Ex: Line is different from line.

* Identifiers may contain letters and digits, but cannot

begin with a digit.

* The underscore character, \_, is also allowed to aid in the readability of long identifier names.
* It should not be used as the first character, however, as identifiers beginning with an underscore have special meaning in Python.

**Keywords**

A keyword is an identifier that has predefined meaning in a programming language. Therefore, keywords cannot be used as “regular” identifiers. Doing so will result in a syntax error.

Ex:

>>> and=10

File "<stdin>", line 1

and=10

^

SyntaxError: invalid syntax

Note: To display the keywords, type help() in the Python shell, and then type keywords.

**Type function**

type is a built in function ,which returns type of the given object.Type of a variable depends on the value assigned to it.

a = 10

print(type(a)) # int

a = 10.0

print(type(a)) # float

a = "python"

print(type(a)) # str

a = True

print(type(a)) # values of bool type : False True

a=False

print(type(a))

a = 2 + 3j

print(type(a)) # complex

The above ones are simple types also called as scalar as they have a single value.

---------------------------------------------------------------------------------------

The below ones are called as structured or reference types:having more than one value.

a = (10, 20, 30, 40)

print(type(a)) # tuple

a = [10, 20, 30, 40]

print(type(a)) # list

a = {10, 20, 30, 40}

print(type(a)) # set

a = {1:20,2:30}

print(type(a)) # dict

**output:**

we use function print for displaying.

As print is a function, we should call or invoke using parentheses following print.

**Characteristics:**

a) can take argument of any type

>>> print(100)

100

>>> print(2.5)

2.5

>>> print(True)

True

>>> print([11, 22])

[11, 22]

>>> print("enjoy")

enjoy

b) can take any number of arguments

print(1, 2, 3, 4)

1 2 3 4

c) each argument is evaluated as an expression

>>> print(2 + 2, 3 \* 4)

4 12

d) In the display, we observe a space between the output fields.

By default, output field separator is a space.

It can be changed by specifying sep = <val> in print.

e) After each print, we get a newline. This is called the output record separator.

This can be changed by specifying end = <val> in print.

Please run and check the output of the following program. Some of this code is tricky!

# file : 1\_output.py

# output

# output field separator : appears between fields in the output

# default : space

# use sep to change this

# output record seperator : appears at the end of each print

# default : newline

# use end to change this

"""

print("one", "two", "three")

print("four", "five")

print("one", "two", "three", sep = "-----", end = "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

print("four", "five", sep = "^^^^^^^^")

"""

"""

#end = "" # NO

print("to", end = "")

print("get", end = "")

print("her")

"""

"""

sep = "stupid"

print("rama", "krishna")

print("apple", "banana", sep = "fool") # does not create a variable called sep

print(sep) # stupid and not fool

"""

sep = " stupid "

# value of variable sep is substituted and nothing special

print("rama", "krishna", "parama ", "hamsa ", " not ", sep) # sep refers to the variable in the program

print("apple", "banana", "carrot", sep = sep) # field separator takes the value of the variable sep

**input**

#example 1

# find the area of a rectangle with given length and breadth

l=10

b=20

area=l\*b

print("area=",area)

#output

#area=200

#in the above program the values of variables are hard coded into the source code itself.

#To allow flexibility we might want to take the input from the user. In Python, we have the input() function to allow this. The syntax for input() is:

# input([prompt])

#where prompt is the string we wish to display on the screen. It is optional.

#input() : gets a string from the keyboard

# int(input()) converts a string to an int

l = int(input())

b = int(input())

a = l \* b

print("area : ", a)

#output

#10

#20

#area : 200

#We can specify a string as prompt,which makes program more readable

l = int(input("enter the length value"))

b = int(input("enter the breadth value"))

a = l \* b

print("area : ", a)

#output

enter the length value 10

enter the breadth value 20

area : 200