

UNIT-I: Basics of python programming

Introduction: Introduction to computer system, algorithms and flowcharts, Ethics and IT policy in company, A Brief History of Python, Applications areas of python, The Programming Cycle for Python, Python IDE.

Elements of Python: Keywords and identifiers, variables, data types and type conversion, operators in python, Operator precedence and associativity, expressions in python, strings, Indexing and Slicing of Strings, Classes and object, constructor.

UNIT-II: Decision Control Statements

Conditional Statements: if statement, if-else statement, Nested-if statement and elif statements.

Loops: Purpose and working of loops, while loop, for loop, else with loop statement, Selecting an appropriate loop, Nested Loops, break, continue and pass statement.

UNIT-III: Function and Modules

Introduction of Function, calling a function, Function arguments, Mutability and Immutability, built in function, scope rules, Namespaces, Garbage Collection, Passing function to a function, recursion, Lambda functions, Map, filter, Reduce.

Modules and Packages: Importing Modules, writing own modules, Standard library modules, `dir()` Function, Packages in Python.

UNIT-IV: Basic Data structures in Python

Python Basic Data Structures: Sequence, Packing and Unpacking Sequences, Mutable Sequences, Strings, Basic operations, Comparing strings, string formatting, Slicing, Built-in string methods and function, Regular expressions, Lists, Tuples, Sets and Dictionaries with built-in methods, List Comprehension, Looping in basic data structures.

UNIT-V: File and Exception handling

Files and Directories: Introduction to File Handling, Reading and Writing files, Additional file methods, Working with Directories.

Exception Handling, Errors, Run Time Errors, Handling I/O Exception, Try-except statement, Raise, Assert.

Course Objective

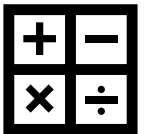
In this semester, the students will

Study the basic of computers and algorithms.



Study the history of Python and Programming Cycle for Python.

Gain the understanding of the Python IDE and understand how to write Python Program.



Learn the elements of Python like Keywords and Identifiers, Variables, Data types and Operators.

1. Introduction to computer system
2. Algorithms and flowcharts
3. Ethics and IT Policy in Company
4. A Brief History of Python
5. Applications areas of python
6. The Programming Cycle for Python, Python IDE
7. Elements of Python
 - Keywords and Identifiers
 - Variables
 - Data types and type conversion
 - Operators in Python
 - Operator precedence and associativity
 - Expressions in Python
 - Strings
 - Indexing and Slicing of Strings
 - Classes and Object
 - Constructor

Unit I Objective

In Unit I, the students will

- Gain the understanding of computer system.
- Study algorithms and flowchart.
- Learn Ethics and IT policy in company.
- Study history, applications areas, programming cycle for Python and Python IDE.
- Study elements of Python.

Topis Prerequisite

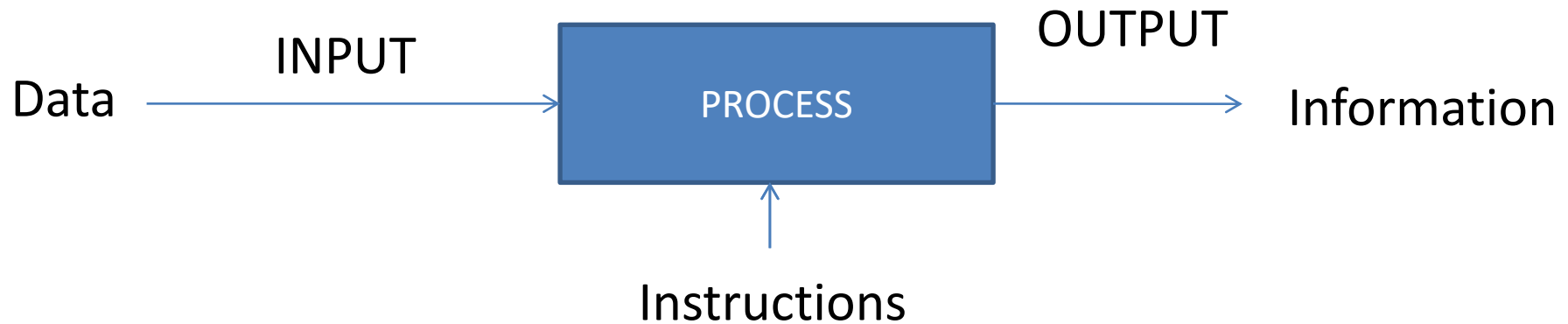
- Basic Understanding of terminology of digital computer.

Topic : Introduction to computer system

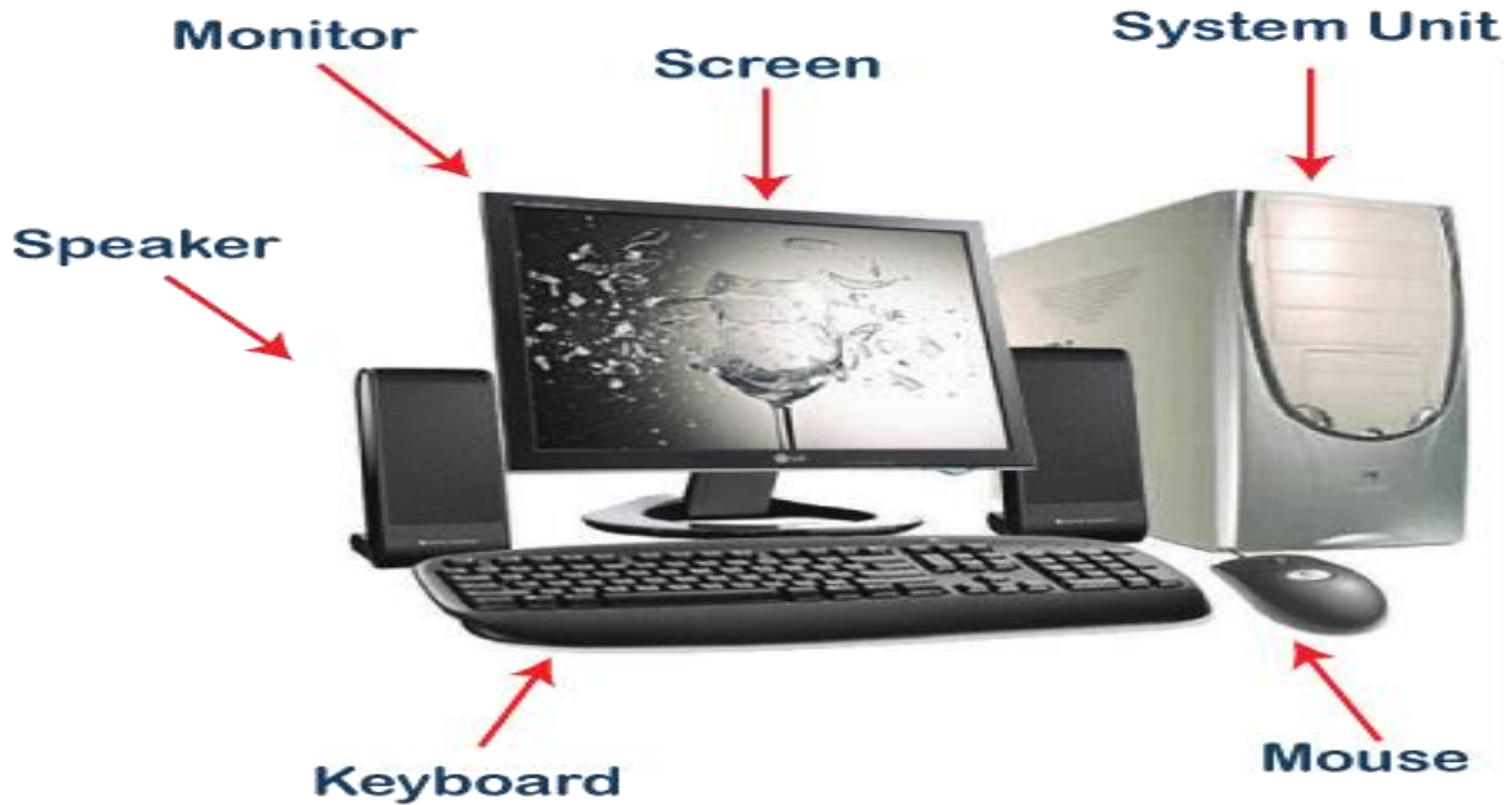
- In this topic, the students will gain the understanding of component of computer, working of its component, characteristics of computer.

Computer (CO1)

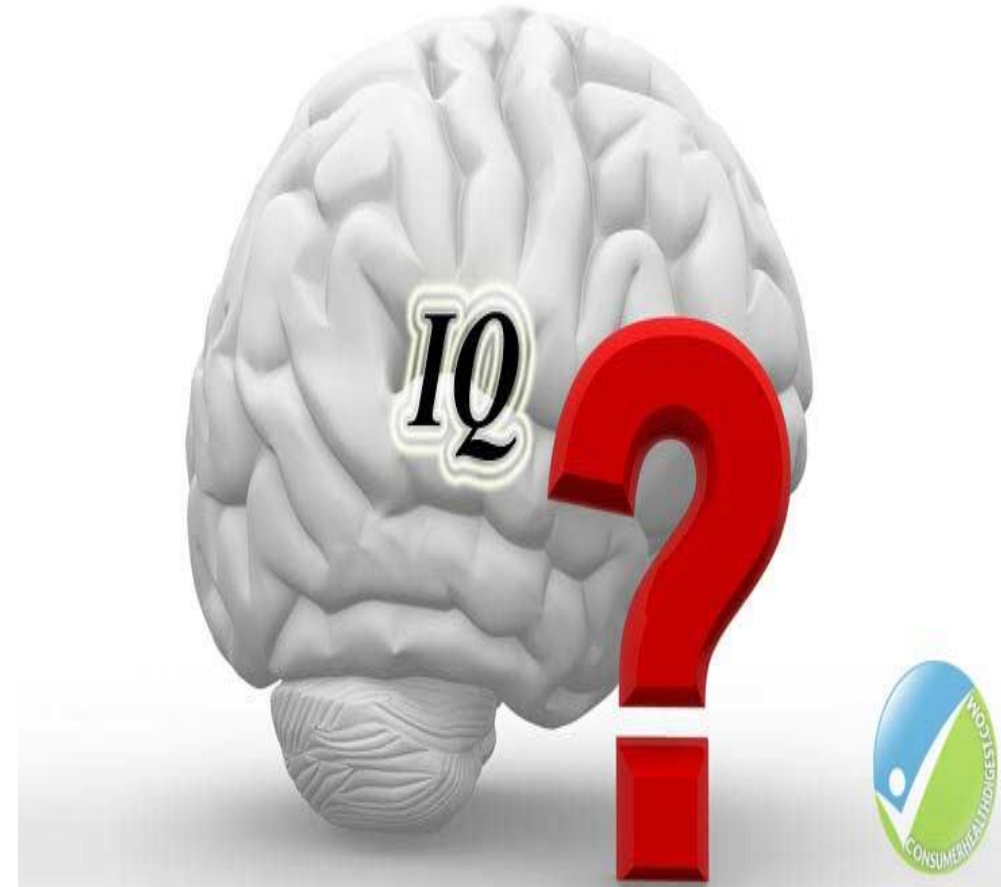
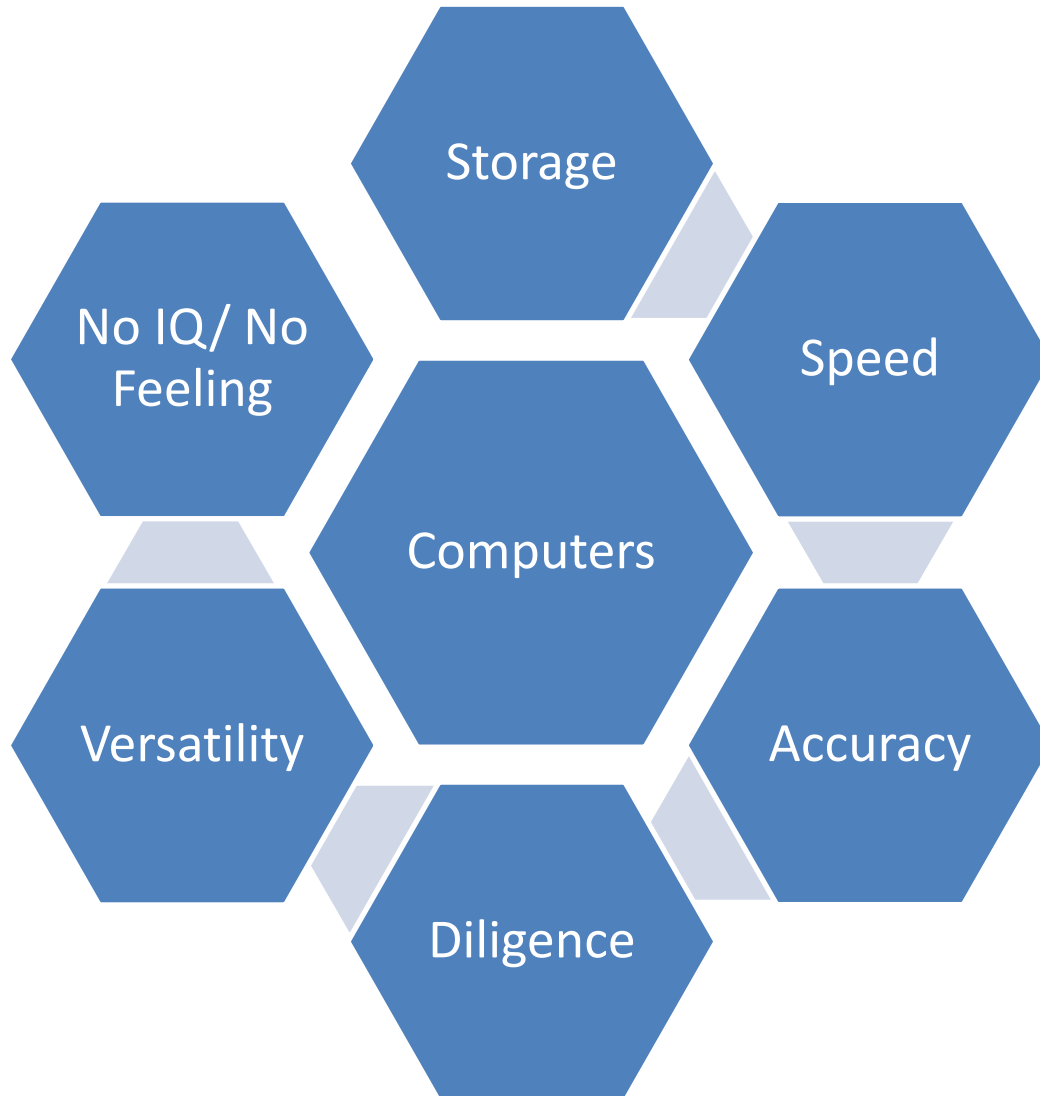
The term computer is derived from the term “**compute**”. Computer is a **programmable electronic device** that takes data and instruction as an input from the user and, process data, and provides useful information.



Computer (CO1)



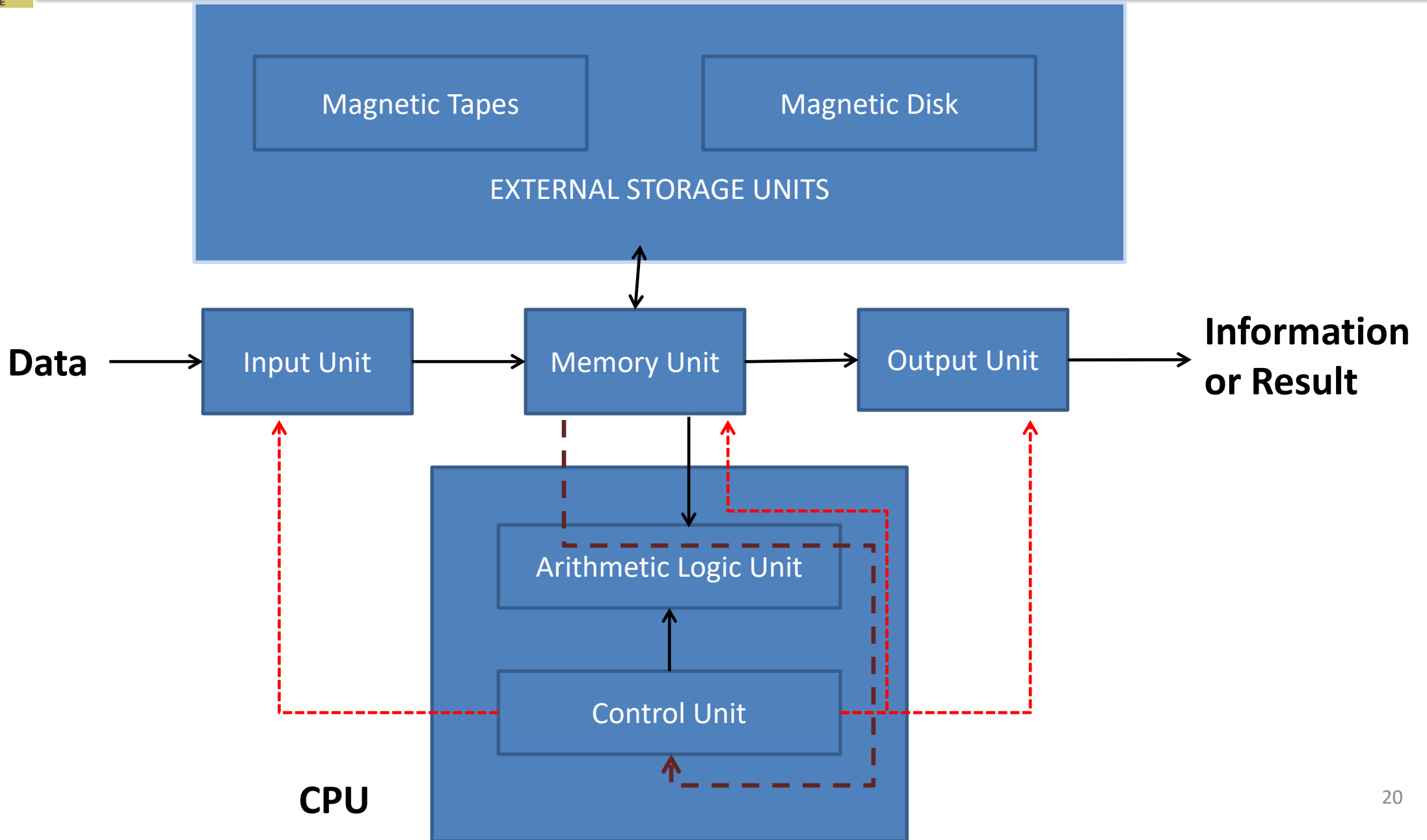
Characteristics of Computers (CO1)



Application of Computers (CO1)

- **Word Processing**
- **Internet**
- **Digital Audio/Video Compression**
- **Desktop Publishing**
- **Traffic Control**
- **Retail Business**
- **Hospitals**
- **Business and Industry**
- **Weather Forecasting**
- **Education**
- **Online Banking**
- **Robotics**
- **Expert Systems**

Block Diagram of Digital Computers (CO1)



Algorithm (CO1)

Algorithm is the well-defined computational procedures that takes some values or set of values as input, process it and produces some values or set of values as output.

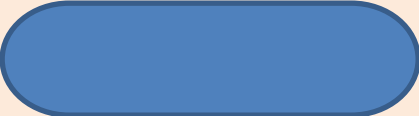




Characteristics of Algorithm (CO1)

1. Input
 - It takes zero or more values as input.
2. Output
 - It produces one or more values as output.
3. Finiteness
 - It should terminate after finite number of steps.
4. Definiteness
 - Each instruction must be clear, precise and unambiguous.
5. Effectiveness
 - Each instruction must be very basic but essential so that it can be carried out using pen and paper.

Flowchart (CO1)

- It is the pictorial representation of the algorithm.
- It is used by programmer as a programming planning tool for organizing the sequence of steps necessary to solve the problem.
- It is also known as the roadmap for programming.

Elements of Flowchart (CO1)

Picture	Name	Meaning
	Ellipse/Oval	Start/End
	Rhombus	Input and output
	Rectangle	Data Processing
	Diamond	Condition
	Arrows	Movement or flow of operations

Algorithm and flowchart (CO1)

1. Write the algorithm and draw the flowchart to compute the sum of two numbers.
2. Write the algorithm and draw the flowchart to compute the sum and average of five numbers.
3. Write the algorithm and draw the flowchart to compute the area of triangle using heron's formula.
4. Write the algorithm and draw the flowchart to compute the temperature in degree Fahrenheit when temperature in degree Celsius is given.
5. Write the algorithm and draw the flowchart to swap two numbers using 3rd variable.
6. Write the algorithm and draw the flowchart to swap two numbers without using 3rd variable.

7. Write the algorithm and draw the flowchart to check whether the number is odd or even.
8. Write the algorithm and draw the flowchart to compute the greater of two numbers.
9. Write the algorithm and draw the flowchart to compute the greatest of three numbers.
10. Write the algorithm and draw the flowchart to check whether the given year is leap year or not.
11. Write the algorithm and draw the flowchart to compute the sum of first N natural numbers.

Algorithm and flowchart(CO1)

12. Write the algorithm and draw the flowchart to compute the factorial of the given number.
13. Write the algorithm and draw the flowchart to compute the sum of digits of the given number.
14. Write the algorithm and draw the flowchart to compute the reverse of the given number. Also check the given number is in palindrome or not.
15. Write the algorithm and draw the flowchart to convert the decimal number to binary number.
16. Write the algorithm and draw the flowchart to convert the binary number to decimal number.

Algorithm and flowchart(CO1)

12. Write the algorithm and draw the flowchart to compute the factorial of the given number.
13. Write the algorithm and draw the flowchart to compute the sum of digits of the given number.
14. Write the algorithm and draw the flowchart to compute the reverse of the given number. Also check the given number is in palindrome or not.
15. Write the algorithm and draw the flowchart to convert the decimal number to binary number.
16. Write the algorithm and draw the flowchart to convert the binary number to decimal number.

Code of Ethics and Professional Conduct (CO1)

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way.

1. GENERAL ETHICAL PRINCIPLES

A computing professional should...

- 1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
- 1.2 Avoid harm.
- 1.3 Be honest and trustworthy.
- 1.4 Be fair and act not to discriminate.
- 1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
- 1.6 Respect privacy.

2. PROFESSIONAL RESPONSIBILITIES

A computing professional should...

- 2.1 Strive to achieve high quality in both the processes and products of professional work.
- 2.2 Maintain high standards of professional competence, conduct, and ethical practice.
- 2.3 Know and respect existing rules pertaining to professional work.
- 2.4 Accept and provide appropriate professional review.

2. PROFESSIONAL RESPONSIBILITIES

- 2.5 Foster public awareness and understanding of computing, related technologies, and their consequences.
- 2.6 Access computing and communication resources only when authorized or when compelled by the public good.
- 2.7 Design and implement systems that are robustly and usably secure.

3. PROFESSIONAL LEADERSHIP PRINCIPLES

- 3.1 Ensure that the public good is the central concern during all professional computing work.
- 3.2 Manage personnel and resources to enhance the quality of working life.
- 3.3 Create opportunities for members of the organization or group to grow as professionals.

4.1 Uphold, promote, and respect the principles of the Code Programming Code of Ethics

Key points of proper conduct for Computer Programmers-
A programmer must...

- Never create or distribute malware.
- Never write code that is intentionally difficult to follow.
- Never write documentation that is intentionally confusing or inaccurate.
- Never reuse copyrighted code unless the proper license is purchased, or permission is obtained.

.

Code of Ethics and Professional Conduct... (CO1)

- Acknowledge (verbally and in source code comments) the work of other programmers on which the code is based, even if substantial changes are made.
- Never intentionally introduce bugs with the intent of later claiming credit for fixing the bugs, or to stimulate the uptake of later versions.
- Never write code that intentionally breaks another programmer's code for the purpose of elevating one's status.
- Never hide known obstacles to a project's completion during any phase of development, especially the design phase.

Code of Ethics and Professional Conduct... (CO1)

- Report any illegal activities of the employer.
- Never falsely deny the presence of bugs.
- Never reveal the secret corporate knowledge of an employer.
- Never accept compensation from multiple parties for the same work unless permission is given.
- Never conceal from the employer their financial interest in development resources.

Code of Ethics and Professional Conduct... (CO1)

- Never maliciously injure the reputation of an employer or members of the development team.
- Never take credit for another's work.
- Never steal software, especially development tools.
- Never install third-party applications without the user's permission.

IT Policy (CO1)

- Companies provides and maintains technological products, services and facilities like Personal Computers (PCs), peripheral equipment, servers, telephones, Internet and application software to its employees for official use.
- The Information Technology (IT) Policy of the organization defines rules, regulations and guidelines for proper usage and maintenance of these technological assets to ensure their ethical and acceptable use and assure health, safety and security of data, products, facilities as well as the people using them.
- It also provides guidelines for issues like purchase, compliance, IT support and grievance redressal of the employees pertaining to technological assets and services used for office work.

- Acceptable Use Policy
- Security awareness
- DR/BCP (Disaster Recovery, Business Continuity plan)
- Change management.
- Equipment Usage policy
- PC standards
- The Internet Usage Policy
- Information security Policy
- Email and chat Policy
- The Software Usage Policy

Python Introduction (CO1)

What is Python Language?

- Python is a high-level general-purpose, interpreted, interactive, object-oriented and reliable language having wide range of applications from Web development, scientific and mathematical computing to desktop graphical user Interfaces.
- The syntax of the language is clean, and length of the code is relatively short.
- It allows to think about the problem rather than focusing on the syntax

A Brief History of Python (CO1)

- Guido Van Rossum was doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in Netherland.
- It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas.

A Brief History of Python Cont...(CO1)

- The programming language which Python is said to have succeeded is ABC Programming Language.
- He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features.
- After that what he did as very clever. He had taken the syntax of ABC, and some of its good features.

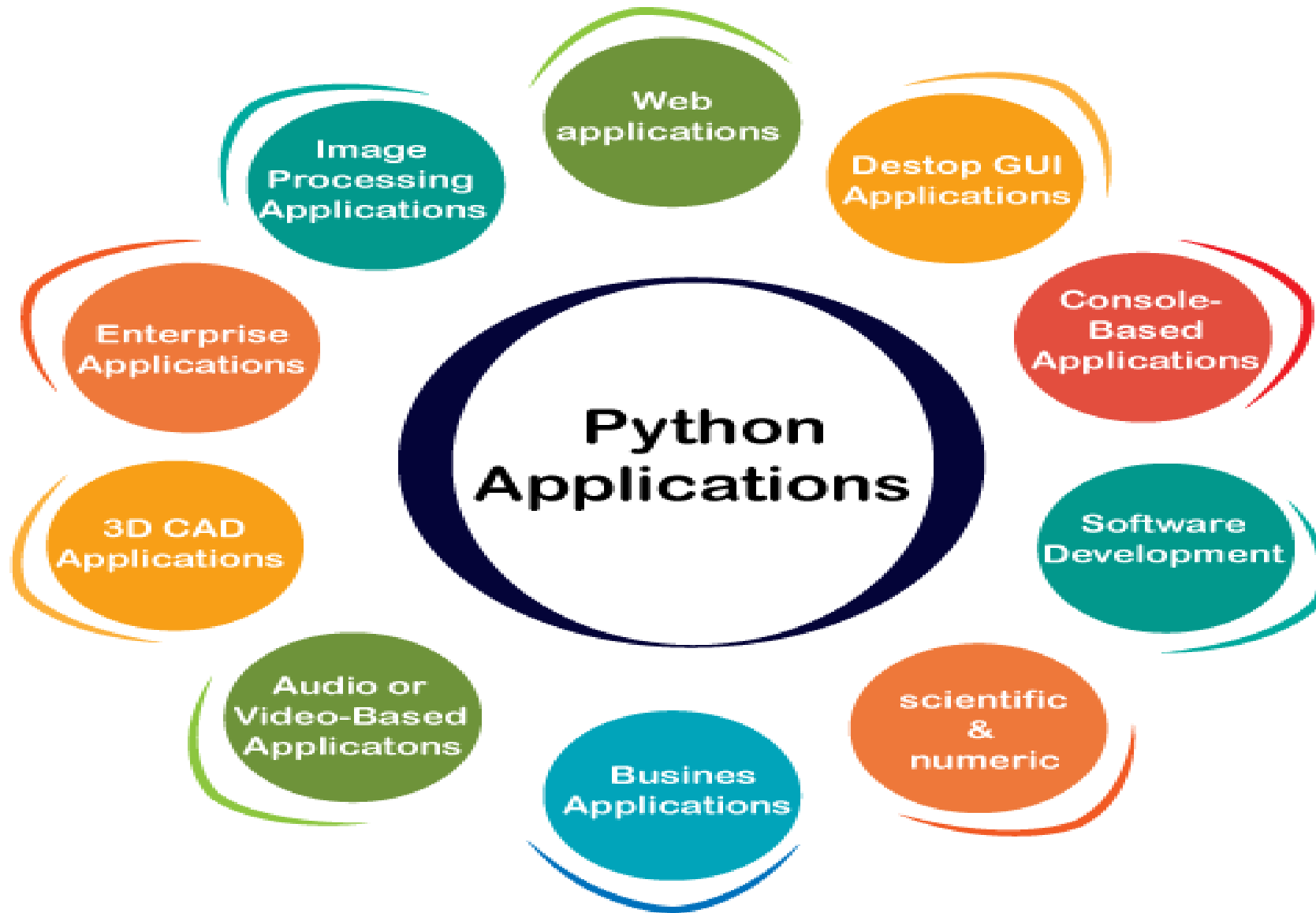
A Brief History of Python Cont...(CO1)

- It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language which had removed all the flaws.
- The inspiration for the name came from BBC's TV Show – 'Monty Python's Flying Circus', as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python!

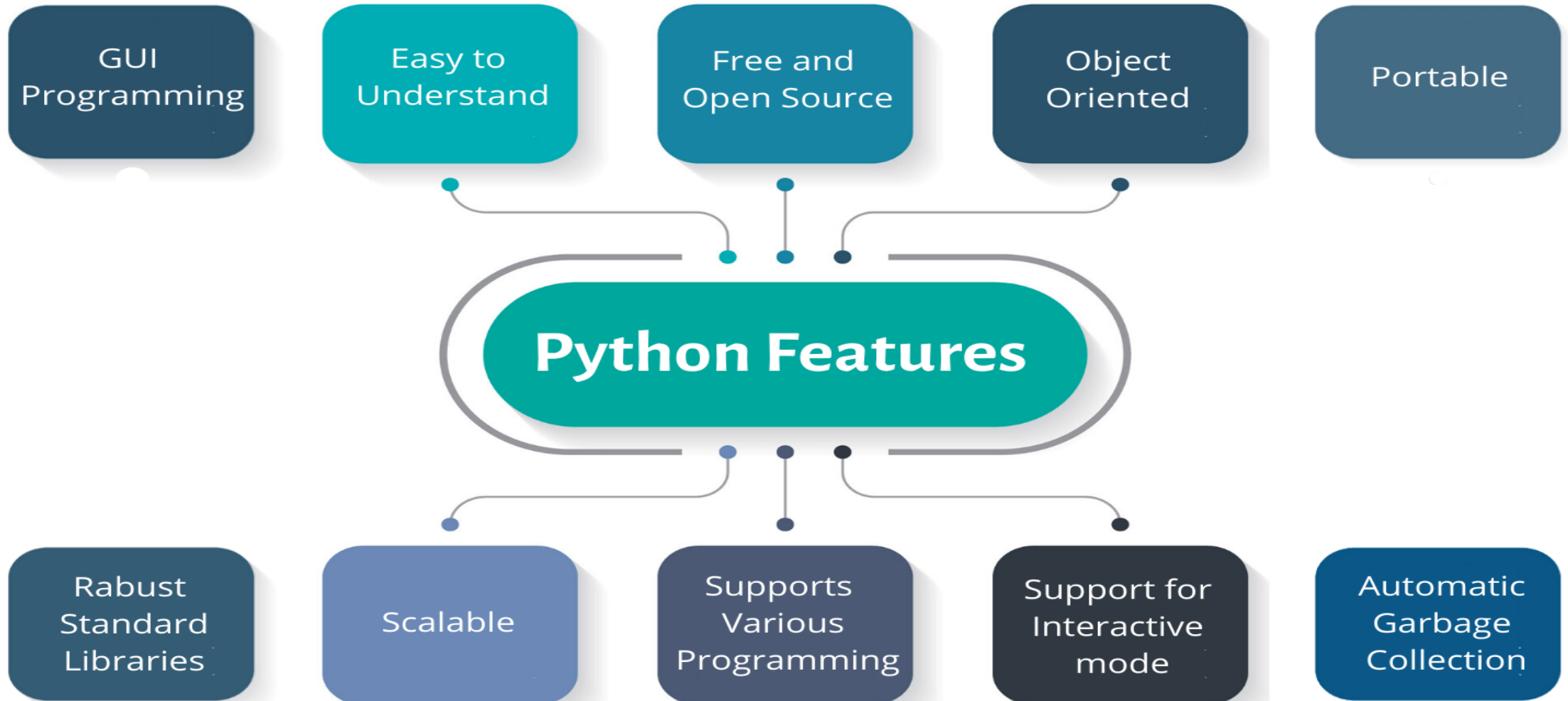
A Brief History of Python Cont... (CO1)

- The language was finally released in 1991.
- When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C.
- Its main objective is to provide code readability and advanced developer productivity.

Applications areas of python (CO1)



Features of Python (CO1)



Features of Python (CO1)

- A simple language which is easier to learn.
- Free and open-source.
- Portability.
- Extensible and Embeddable
 - easily combine pieces of C/C++ or other languages with Python code.
- A high-level, interpreted language.
- Large standard libraries to solve common tasks.

- Object-oriented
 - Everything in Python is an object.
 - Object oriented programming (OOP) helps to solve a complex problem intuitively.
 - Structure supports such concepts as polymorphism, operation overloading, and multiple inheritance.
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It supports automatic garbage collection.
- **Scalable**
 - Python provides a better structure and support for large programs than shell scripting.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
- **Databases**
 - Python provides interfaces to all major commercial databases.

Getting Python (CO1)

- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python

<https://www.python.org/>

- Python documentation can be downloaded from

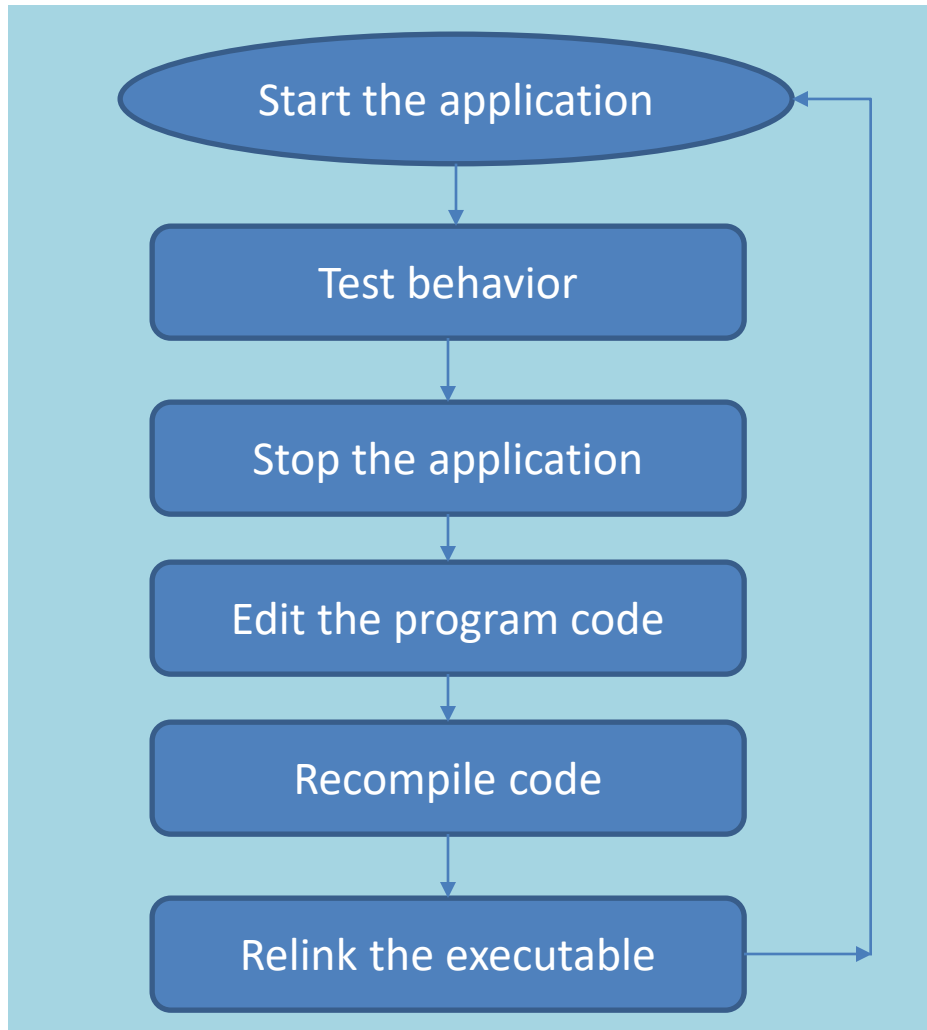
<https://www.python.org/doc/>

The Programming Cycle for Python (CO1)

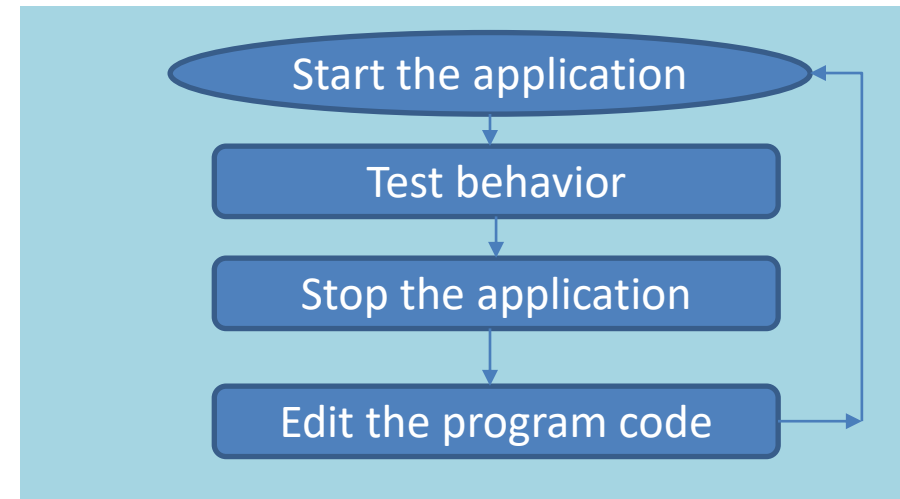
- Python's development cycle is dramatically shorter than that of traditional languages.
- In Python, there are no compile or link steps.
- Python programs simply import modules at runtime and use the objects they contain.
- Python programs run immediately after changes are made.
- And in cases where dynamic module reloading can be used, it's even possible to change and reload parts of a running program without stopping it at all.

The Programming Cycle for Python (CO1)

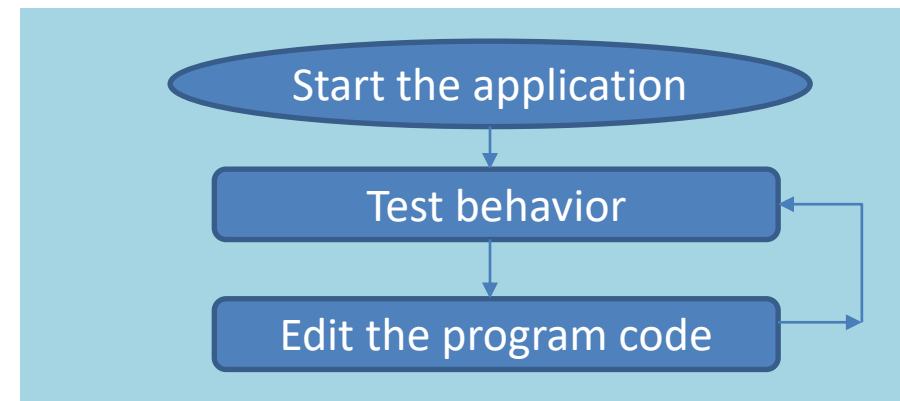
Traditional Development Cycle



Python's Development Cycle



Python's Development Cycle with Module Reloading



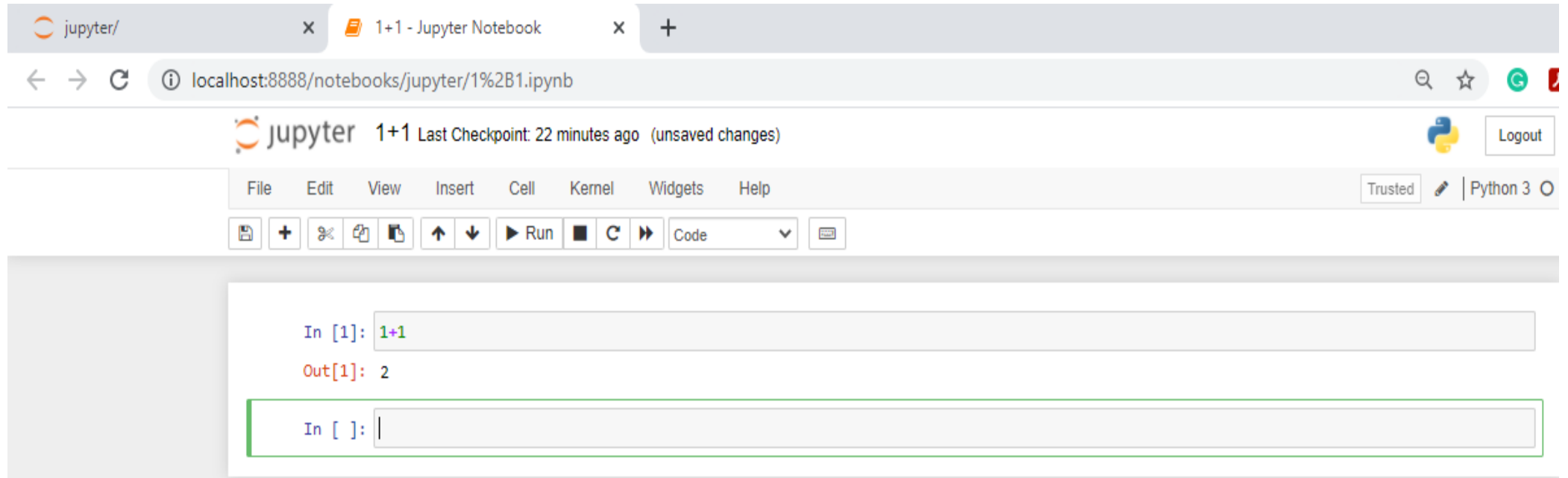
- **Spyder** 
– <https://www.spyder-ide.org/>
- **IDLE** 
– <https://docs.python.org/3/library/idle.html>
- **Sublime Text 3** 
– <https://www.sublimetext.com/3>
- **Jupyter** 
– <https://jupyter.org/install.html>

Interacting with Python Programs (CO1)

1. Open the command prompt.
2. Write **jupyter notebook** in command prompt and then press enter key.
3. Jupyter notebook will be opened in link of laptop browser. For e.g., <http://localhost:8888/tree> this kind of link will open up.
4. Select the path you want to save the file.
5. Click on the **new** option on the right most corner and select python3 from cursor. Untitled notebook will open up, you can change up the name by selecting it.

Elements of Python (CO1)

6. Click on the code cell write down the code **1+1** and then press shift + Enter from keyboard/run command from interface. Then check the output.



The screenshot displays a Jupyter Notebook interface in a web browser. The browser's address bar shows the URL `localhost:8888/notebooks/jupyter/1%2B1.ipynb`. The notebook's title bar indicates the file name `1+1 - Jupyter Notebook` and shows the last checkpoint was 22 minutes ago with unsaved changes. The interface includes a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for saving, creating new cells, deleting cells, undo, redo, and running code. The main area contains a code cell with the input `In [1]: 1+1` and the output `Out[1]: 2`. A new code cell is currently active, showing `In []: |`.

Elements of Python (CO1)

Topic Objective

The students will study the elements of Python like Keywords and Identifiers, Variables, Data types and Operators.

Prerequisite and Recap

- Expression
- Operators
- Constant and variable

- Keywords and Identifiers
- Variables
- Data types and type conversion
- Operators in Python
- Expressions in Python

Keywords (CO1)

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- All the keywords except True, False and None are in lowercase, and they must be written as they are.
- Example
 - and, as, assert, await, break, class, continue, def, del, elif, else, except, finally, for, global, if, import, in, is, lambda, not, or, pass, raise, return, try, while, with, yield

Identifiers (CO1)

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.

Rules for writing Identifiers (CO1)

- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.
- An identifier cannot start with a digit.
- Keywords cannot be used as identifiers.
- Cannot use special symbols like !, @, #, \$, % etc. in identifier.
- An identifier can be of any length.

Variables (CO1)

- A variable is a location in memory used to store some data (value).
- Unique names are given to them to differentiate between different memory locations.
- No need to declare a variable before using it. The declaration happens automatically when a value is assigned to a variable.
- The Assignment operator (=) is used to assign values to variables.

- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.
- For example
 - >>> num = 347 # An integer assignment
 - >>> x = 45.89 # A floating point
 - >>> name = "Aman" # A string

Multiple Assignment in Variables (CO1)

- A single value may be assigned to several variables simultaneously.
- For example

```
>>> a = b = c = 1
```

- Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.
- It also allows to assign multiple objects to multiple variables.
- For example –

```
>>> a, b, c = 23, 29.45, "Ram"
```

- Here, two integer objects with values 23 and 29.45 are assigned to variables a and b respectively, and one string object with the value "Ram" is assigned to the variable c.

- Every value in Python has a datatype, that are used to define the operations possible on them and the storage method for each of them.
- Since everything is an object in Python programming, data types are classes and variables are instance (object) of these classes

Standard Data types (CO1)

Data Types	Keyword
Text Type	str
Numeric Types	int, float, complex
Sequence Types	list, tuple, range
Mapping Type	dict
Set Types	set
Boolean Types	bool
Binary Types	bytes

- They store numeric values. Number objects are created when a value is assigned to them. For ex:

```
>>> var1 = 1
```

```
>>> var2 = 10
```

- They can be deleted the reference to a number object by using the del statement. The syntax of the del statement is

```
del var1[,var2[,var3[....,varN]]]
```

- A single object or multiple objects can be deleted by using the del statement. For example

```
>>> del var
```

```
>>> del var_a, var_b
```


Types of Number Types (CO1)

1. int (signed integers)
2. float (floating point real values)
3. complex (complex numbers)

Examples of Number Types (CO1)

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
0o80	32.3+e18	.876j
-0o490	-90	-.6545+0J
-0x260	-32.54E100	3e+26J

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

- For example

```
>>> str = 'Hello World!'
```

```
>>> print(str)      # Prints complete string
```

```
>>> print(str[0])   # Prints first character of the string
```

```
>>> print(str[2:5]) # Prints characters starting from 3rd to 5th
```

```
>>> print(str[2:])  # Prints string starting from 3rd character
```

```
>>> print(str * 2)  # Prints string two times
```

```
>>> print(str + 'TEST') # Prints concatenated string
```

Slicing

- A range of characters can be returned by using the slice syntax.
- Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

Program

```
b = "Hello, World!"  
print(b[2:5])
```

Program Output

llo

Program to demonstrate slicing of Strings (CO1)

```
# String slicing
```

```
String ='ASTRING'
```

```
# Using slice constructor
```

```
s1 = slice(3)
```

```
s2 = slice(1, 5, 2)
```

```
s3 = slice(-1, -12, -2)
```

```
print("String slicing")
```

```
print(String[s1])
```

```
print(String[s2])
```

```
print(String[s3])
```

Output of slicing of Strings (CO1)

Output:

String slicing

AST

SR

GITA

- A List is an ordered sequence of item, It contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are like arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- Declaring a list is straight forward. Items separated by commas are enclosed within brackets [].

- `>>> a = [1, 2.2, 'python']`
 - The values stored in a list can be accessed using the slice operator (`[]` and `[:]`) with indexes starting at 0 in the beginning of the list.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Lists (CO1)

```
>>> list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
```

```
>>> tinylist = [123, 'john']
```

```
>>> print(list)
```

Prints complete list

```
>>> print(list[0])
```

Prints first element of the list

```
>>> print(list[1:3])
```

Prints elements starting from 2nd till 3rd

```
>>> print(list[2:])
```

Prints elements starting from 3rd element

```
>>> print(list * 2)
```

Prints list two times

```
>>> print(list + tinylist)
```

Prints concatenated lists

- A tuple is similar to the list. A tuple consists of several values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as read-only lists.
- The only difference is that tuples are immutable. Tuples once created cannot be modified.
- Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.

Tuple (CO1)

```
>>> tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
```

```
>>> tinytuple = (123, 'john')
```

```
>>> print(tuple)
```

Prints complete tuple

```
>>> print(tuple[0])
```

Prints first element of the tuple

```
>>> print(tuple[1:3])
```

Prints elements starting from 2nd till 3rd

```
>>> print(tuple[2:])
```

Prints elements starting from 3rd element

```
>>> print(tuple * 2)
```

Prints tuple two times

```
>>> print(tuple + tinytuple)
```

Prints concatenated tuples

- Set is an unordered collection of unique items.
- Set is defined by values separated by comma inside braces { }.
- Items in a set are not ordered.
- Set operations can be performed like union, intersection on two sets. Set have unique values. They eliminate duplicates.
- indexing has no meaning as set are unordered collection. Hence the slicing operator [] does not work.

- For example

```
>>>a = {1,2,2,3,3,3}
```

```
>>> a
```

```
{1, 2, 3}
```

Dictionary (CO1)

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data.
- Dictionaries are optimized for retrieving data. It is must to know the key to retrieve the value.
- A dictionary key can be almost any Python type but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are defined within braces {} with each item being a pair in the form key: value. Key and value can be of any type.
- For example

```
>>> d = {1:'value','key':2}
>>> type(d)
<class 'dict'>
```

- The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.
- Python has two types of type conversion.
 - Implicit Type Conversion
 - Explicit Type Conversion

Implicit Type conversion (CO1)

- Python automatically converts one data type to another data type without any user involvement.
- It always converts smaller data types to larger data types to avoid the loss of data.
- Example

```
>>> a = 4
>>> b = 2.3
>>> c = a + b
```
- In above example, data type of a and b are int and float, respectively. Data type of c will be float and its value is 6.3

Explicit Type conversion (CO1)

- In Explicit Type Conversion, users convert the data type of an object to required data type.
- The predefined functions like `int()`, `float()`, `str()` are used to perform explicit type conversion.
- This type of conversion is also called typecasting because the user casts/changes the data type of the objects.
- Syntax
 <datatype>(expression)
- Example
 >>> a = 2.6
 >>> c = int(a)

Function of type conversion (CO1)

Function	Description
int(x)	Convert x to an integer
float(x)	Convert x to a float number
str(x)	Convert x to a string
tuple(x)	Convert x to a tuple
list(x)	Convert x to a list
set(x)	Convert x to a set
ord(x)	Convert x to ASCII code
bin(x)	Convert x to a binary
oct(x)	Convert x to an octal
hex(x)	Convert x to a hexadecimal
chr(x)	Convert x to a character

Python language supports the following types of operators.

- Arithmetic Operators
- Relational Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators (CO1)

Operator	Description	Example (a=5, b=3)
+	Adds values on either side of the operator.	$a + b = 8$
-	Subtracts right hand operand from left hand operand.	$a - b = 2$
*	Multiplies values on either side of the operator	$a * b = 15$
/	Divides left hand operand by right hand operand	$a / b = 1.6667$
%	Divides left hand operand by right hand operand and	$a \% b = 2$
**	Performs exponential (power) calculation on operators	$a ** b = 5^3 = 125$
//	The division of operands where the result is the quotient in which the digits after the decimal point are removed.	$a // b = 1$

Relational Operators (CO1)

Operator	Description	Example a=5, b=3
==	If the values of two operands are equal, then the condition becomes true	a == b is not true.
!=	If values of two operands are not equal, then condition becomes true.	a != b is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	a > b is true
<	If the value of left operand is less than the value of right operand, then condition becomes true.	a < b is not true
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	a >= b is true
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	a <= b is not true

Assignment Operators (CO1)

Operator	Description
=	$a = b$ Assigns values from right side operands(b) to left side operand (a)
+=	$a += b$ is same as $a = a + b$ It adds right operand to the left operand and assign the result to left operand
-=	$a -= b$ is same as $a = a - b$ It subtracts right operand from the left operand and assign the result to left operand
*=	$a *= b$ is same as $a = a * b$ It multiplies right operand with the left operand and assign the result to left operand
/=	$a /= b$ is same as $a = a / b$ It divides left operand with the right operand and assign the result to left operand

Assignment Operators (CO1)

Operator	Description
<code>%=</code>	$a\%=b$ is same as $a = a \% b$ It takes modulus using two operands and assign the result to left operand
<code>**=</code>	$a**=b$ is same as $a = a ** b$ Performs exponential (power) calculation on operators and assign value to the left operand
<code>//=</code>	$a//=b$ is same as $a = a // b$ It performs floor division on operators and assign value to the left operand

Logical Operators (CO1)

Logical operators are the and, or, not operators.

Operator	Description
and	True if both the operands are true
or	True if either of the operands is true
not	True if operand is false (complements the operand)

Logical Operators (CO1)

a	b	a and b	a or b	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

- Bitwise operators manipulate the data at bit level.
- These are applicable on integer values.
- Types
 - & (Bitwise and operator)
 - | (Bitwise or operator)
 - ^ (Bitwise XOR operator)
 - ~ (Bitwise one's complement operator)
 - << (Bitwise left-shift operator)
 - >> (Bitwise right-shift operator)

Bitwise Operators (CO1)

a	b	a & b	a b	a ^ b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Bitwise Operators (CO1)

Operator	Let $a = (92)_{10} = (0101\ 1100)_2$ and $b = (14)_{10} = (0000\ 1110)_2$	
&	$c = (a \& b) = 92 \& 14$	$ \begin{array}{r} 0101\ 1100 \\ \& 0000\ 1110 \\ \hline 0000\ 1100 = (12)_{10} \end{array} $
	$c = (a b) = 92 14$	$ \begin{array}{r} 0101\ 1100 \\ 0000\ 1110 \\ \hline 0101\ 1110 = (94)_{10} \end{array} $
^	$c = (a \wedge b) = 92 \wedge 14$	$ \begin{array}{r} 0101\ 1100 \\ \wedge 0000\ 1110 \\ \hline 0101\ 0010 = (82)_{10} \end{array} $

Bitwise Operators (CO1)

Operator	Let $a = (92)_{10} = (0101\ 1100)_2$ and $b = (14)_{10} = (0000\ 1110)_2$	
\sim	$c = \sim a = \sim 92$	$c = \sim(0101\ 1100) = 1010\ 0011$
\ll	$c = a \ll 1 = 46 \ll 1$	$C = 00101\ 110 \ll 1 = 01011100 = (92)_{10}$
\gg	$c = a \gg 2 = 92 \gg 2$	$C = 0101\ 1100 \gg 2 = 0001\ 0111 = (23)_{10}$

Membership Operators (CO1)

- **in** and **not in** are the membership operators in Python.
- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).
- In a dictionary, we can only test for presence of key, not the value.

Operator	Description	Example $x = \{2,3,5\}$	Result
in	True if value/variable is found in the sequence	5 in x	True
not in	True if value/variable is not found in the sequence	5 not in x	False

Identity Operators (CO1)

- Identity operators compare the memory locations of two objects.
- **is** and **is not** are the identity operators in Python.
- They are used to check if two values (or variables) are located on the same part of the memory.
- Two variables that are equal does not imply that they are identical.

Operator	Description	Example a = 'Hello' b = 'Hello'	Result
is	True if the operands are identical (refer to the same object)	a is b	True
is not	True if the operands are not identical (do not refer to the same object)	a is not b	False

Operator Precedence and associativity(CO1)

- When an expression contains two or more than two operators, then it is evaluated based on operator precedence and associativity.
- Each operator of same precedence is grouped in same level and operator of higher precedence is evaluated first.
- Two or more operators having equal precedence are evaluated either left-to-right(LTR) or right-to-left(RTL) based on their associativity.

Operator Precedence and associativity(CO1)

Operator	Description
()	Parenthesis
**	Exponentiation
~,+,-	Unary operators
*,/,//,%	Arithmetic multiply, division, floor and modulo division
+, -	Addition and subtraction
>>,<<	Bitwise left and right shift operator
&	Bitwise and operator
^	Bitwise Ex-or operator
	Bitwise or operator

Operator Precedence and associativity(CO1)

Operator	Description
<=,>=,<,>	Relational inequality operators
==, !=	Equal and not equal operators
=,*=,/=,//=,%=,+=,-=,**=,&=	Assignment operators
is, is not	Identity operators
in, not in	Membership operators
not	Logical not operator
and	Logical and operator
or	Logical or operator

- Expressions are representations of value.
- They are different from statement in the fact that statements do something while expressions are representation of value.
- Python expression contains identifiers, literals, and operators.
- For Example
 $x = a * b + c / d - f$ is an expression.

- The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

Syntax

class ClassName:

 <statement-1>

 .

 .

 <statement-N>

Object (CO1)

- The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.
- Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute `__doc__`, which returns the docstring defined in the function source code.
- When we define a class, it needs to create an object to allocate the memory. Consider the following example.

Example of Object (CO1)

```
class car:  
    def __init__(self,modelname, year):  
        self.modelname = modelname  
        self.year = year  
    def display(self):  
        print(self.modelname,self.year)
```

```
c1 = car("Toyota", 2016)  
c1.display()
```

Toyota 2016

Python Constructor(CO1)

- A constructor is a special type of method (function) which is used to initialize the instance members of the class.
- In C++ or Java, the constructor has the same name as its class, but it treats constructor differently in Python. It is used to create an object.
- Constructors can be of two types.
 - Parameterized Constructor
 - Non-parameterized Constructor
- Constructor definition is executed when we create the object of this class. Constructors also verify that there are enough resources for the object to perform any start-up task.

Creating Python Constructor(CO1)

- In Python, the method the `__init__()` simulates the constructor of the class. This method is called when the class is instantiated. It accepts the **self**-keyword as a first argument which allows accessing the attributes or method of the class.
- We can pass any number of arguments at the time of creating the class object, depending upon the `__init__()` definition. It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.
- Consider the following example to initialize the **Employee** class attributes.

Example Python Constructor(CO1)

```
class Employee:
```

```
    def __init__(self, name, id):
```

```
        self.id = id
```

```
        self.name = name
```

```
    def display(self):
```

```
        print("ID: %d \nName: %s" % (self.id, self.name))
```

```
emp1 = Employee("John", 101)
```

```
emp2 = Employee("David", 102)
```

```
# accessing display() method to print employee 1 information
```

```
emp1.display()
```

```
# accessing display() method to print employee 2 information
```

```
emp2.display()
```

Output of example(CO1)

ID: 101

Name: John

ID: 102

Name: David

Python Non-Parameterized Constructor (CO1)

The non-parameterized constructor uses when we do not want to manipulate the value or the constructor that has only self as an argument.

Example:

```
class Student:
```

```
    # Constructor - non parameterized
```

```
    def __init__(self):
```

```
        print("This is non parametrized constructor")
```

```
    def show(self,name):
```

```
        print("Hello",name)
```

```
student = Student()
```

```
student.show("John")
```

Output of Non-Parameterized Constructor (CO1)

This is non parametrized constructor
Hello John

- Constructor with parameters is known as parameterized constructor.
- The parameterized constructor takes its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

Example:

class Student:

 # Constructor - parameterized

def __init__(self, name):

print("This is parametrized constructor")

 self.name = name

def show(self):

print("Hello",self.name)

student = Student("John")

student.show()


Output:

This is parametrized constructor

Hello John

- The default constructor is a simple constructor which doesn't accept any arguments.
- Its definition has only one argument which is a reference to the instance being constructed.

- Youtube/other Video Links
- Evolution of Computer
 - <https://youtu.be/OQax5NF5aEw>
- Application of Computers
 - <https://youtu.be/ac7T3ocg9gk>
- Code of Conduct
 - <https://youtu.be/txNEiGwVtUA>
- Algorithm and flowchart
 - https://youtu.be/RwnY_mJ6ras
- Introduction to Python
 - <https://youtu.be/xxBb7OyKXY>

1. In Python, 'Hello', is the same as "Hello"
 - a) True 
 - b) False

2. _____*_____Operator is used to multiply numbers.

Weekly Assignment 1.1

1. Give classification of computers. (CO1)
2. Explain memory hierarchy with the help of a diagram. (CO1)
3. What is a digital computer? Draw the block diagram of a digital computer and explain its each component. (CO1)
4. Explain the need of cache memory in a computer system. (CO1)

Weekly Assignment 1.1

5. Define the term software and hardware. Briefly explain system software and application software with at least one example of each. (CO1)
6. What is operating system? List functions and types of operating system. (CO1)
7. Define algorithm. Write Properties of an algorithm. (CO1)
8. Write an algorithm and draw a flow-chart to swap two numbers without using third variable. (CO1)

Weekly Assignment 1.1



9. Write an algorithm and draw a flow-chart to check whether given year is a leap year or not. (CO1)
10. Write an algorithm and draw a flow-chart to compute factorial of a given number. (CO1)
11. Write an algorithm and draw a flow-chart to check whether the given number is an armstrong number or not. (CO1)
12. Write an Algorithm and design a flow-chart to compute prime factors of the given number. (CO1)

Weekly Assignment 1.2


1. What is Python? What are the benefits of using Python?
(CO1)
2. Discuss about feature and application area of python
(CO1)
3. What is PEP 8? How is memory managed in Python?
(CO1)
4. Explain types of operators used in python.
(CO1)

Weekly Assignment 1.2

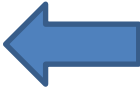
5. What is class and objects? Explain role of constructor. (CO1)
6. What do you mean by tokens? Explain different elements of python. (CO1)
7. Explain operator precedence and associativity with suitable example. (CO1)
8. Write short notes on “Ethics and IT policy in a company”. (CO1)
9. Explain the rules of naming identifier in Python. (CO1)
10. Write a python program to compute the surface area and volume of cone. (CO1)
11. Write a Python Program to swap values of two variable using comma(,) operator. (CO1)

1. What is the correct file extension for Python files?
 - a) .pyt
 - b) .py 
 - c) .pt
 - d) .pyth
2. What is the output for –
 'python' [-3]?
 - a) 'o'
 - b) 't'
 - c) 'h' 
 - d) Negative index error.

3. How do you create a variable with the numeric value 5?

- a) `x = 5`
- b) `x = int(5)`
- c) Both option (a) and (b) are correct 
- d) No option is correct

4. Which one is NOT a legal variable name?

- a) `My_var`
- b) `My-var` 
- c) `_Myvar`
- d) `Myvar`

5. Which operator can be used to compare two values?

a) $<>$

b) $><$

c) $==$ 

d) $=$

6. Which one is a legal identifier?


a) `ab#cd`

b) `abc` 

c) `S.l`

d) `Area of circle`

7. Which of these collections defines a SET?

a) {1, 2, 3, 4} 

b) {'1':'12','2':'45'}

c) (1,2,3,4)

d) [1,2,3,4]

1. In Python what is slicing?
[NIET Autonomous 2020-2021(odd), 1 marks]
2. Write name of any two Python Editors(IDE)?
[NIET Autonomous 2020-2021(odd), 1 marks]
3. How is Python an interpreted language?
[NIET Autonomous 2020-2021(odd), 1 marks]
4. What does `[::-1]` do?
[NIET Autonomous 2020-2021(odd), 1 marks]
5. What is `__init__`?
[NIET Autonomous 2020-2021(odd), 1 marks]

6. What is negative index in Python?
[NIET Autonomous 2020-2021(odd), 1 marks]
7. Define floor division with example?
[NIET Autonomous 2020-2021(odd), 2 marks]
8. Define the Programming Cycle for Python?
[NIET Autonomous 2020-2021(odd), 2 marks]
9. Discuss format specifiers and escape sequences with examples.
[NIET Autonomous 2020-2021(odd), 6 marks]
10. Explain Ethics and IT policy in company.
[NIET Autonomous 2020-2021(odd), 10 marks]

11. Explain the following:

- i. Implicit and Explicit type-casting
- ii. Rules for naming an Identifier

[NIET Autonomous 2020-2021(odd), 10 marks]

12. What is the difference between list and tuples?

[AKTU 2019-2020(odd), 2 marks]

13. In some languages, every statement ends with semicolon(;). What happens if you put a semi-colon at the end of a Python statement?

[AKTU 2019-2020(odd), 2 marks]

15. Mention five benefits of using Python.

[AKTU 2019-2020(odd), 2 marks]

16. How is Python an interpreted language?

[AKTU 2019-2020(odd), 2 marks]

17. What type of language is Python ?

[AKTU 2019-2020(odd), 2 marks]

18. Define floor division with example.

[AKTU 2019-2020(odd), 2 marks]

19. Write short notes with example: The Programming cycle for Python, Elements of Python, Type conversion in Python, operator precedence and Boolean expression.

[AKTU 2019-2020(odd), 10 marks]

Expected Questions for University Exam

1. Define operator. Explain various types of operators in Python with suitable example.
2. Explain standard data types in Python.
3. Explain Programming cycle of Python.
4. Explain the type conversion in Python.
5. Names some frequently used Python IDEs.

- Introduction to computer system
- Algorithms and flowcharts
- Ethics and IT policy in company
- The Programming Cycle for Python
- Python IDE
- Features of Python
- Elements of Python

References

1. Allen B. Downey, “Think Python: How to Think Like a Computer Scientist”, 2nd edition, Updated for Python 3, Shroff/O’Reilly Publishers, 2016.
2. Robert Sedgewick, Kevin Wayne, Robert Dondero, “Introduction to Programming in Python: An Inter-disciplinary Approach” , Pearson India Education Services Pvt. Ltd., 2016.
3. Paul Barry, “Head First: A Brain Friendly Guide” O’Reilly publisher.
4. Reema Thareja, “Python Programming: Using Problem Solving Approach” 2nd Edition, Oxford University Press publisher.
5. Guido van Rossum and Fred L. Drake Jr, “An Introduction to Python”, Revised and updated for Python 3.2, Network Theory Ltd., 2011.

THANK YOU