

# DOCUMENTATION

**Presented by ): Kumarsen Parida.(3rd yr.)**

**SYLLABUS(0-1)**

* Computer System Overview
* Data Representation
* Insight into Program Execution
* Computation Thinking And Getting Started With Python
* Python Fundamentals
* Data Handling
* Conditional Statements
* Tinker
* Project

Computation Thinking ):

Computational thinking divided into four type , i.e. -

1. Decomposition
2. Pattern Recognition
3. Abstraction
4. Algorithms

1.Decomposition):

Decomposition means to divide or break the whole problem into several parts or divide into modules for better understanding and to solve the problem easily.

2.Pattern Recognition):

After decomposition of the whole problem into several parts , the process to recognize the similarity between two or more parts is otherwise called as Pattern Recognition.

3.Abstraction):

Abstraction means to hide unnecessary information from end user that means to give that much information which data is necessary to end user.

4.Algorithm):

Algorithm is the set of instruction or rules which is used to perform a specific task to get solution of program.

Getting Started with Python ):

Generally there is two way to get start with Python

* With installing python compiler like pycharm
* With installing python runner application like Anaconda Navigator(IDE)

To install the above apps we just need to search in Google and then have to download the app in pc and ready to get start after installing them.

Python Fundamentals):

1. Python Style rules and Conventions –

* Indentations :
  + 1. Same block of code will have same spaces (there is no curly braces rules in Python like other programming language).
    2. Use two blank-line(space) between top level definition.
    3. One blank-line between methods.
    4. White space around operations (like (a + b)).
* Statement Terminated : Statement terminated is just break statement inside a loop.
* Max length 79 Character : In python maximum length of any character is 79 character inside a program.
* Case sensitive: python is very case sensitive means there is no work-done if any mismatch occurs (i.e. Python != python).
* Doc-strings: It is used for comment passing inside the program , In python the statement pass inside (‘’’) is treated as comment (i.e. ‘’’ comment ‘’’).
* Naming convention: In python the naming convention Is in Camel-Case.

2.Literals: Literal is a raw data given in a variable or constant in Python.

3.Identifiers (variables): In Python, we don't assign values to the variables, whereas Python gives the reference of the object (value) to the variable , In python some of the variable objects are predefined and can be access-able by programmer i.e. python developers are already done the memory management , there is no required of another object to store the same element.

4.keywords: 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.

5.Punctuation: punctuation is a pre-initialized string used as string constant. In Python, string. punctuation will give the all sets of punctuation.

6.Most programs today use a dialog box as a way of asking the user to provide some type of input . While python provides us two inbuilt functions to read the input from the key-board .

* + 1. raw\_input(prompt):

It works in older version of python ( i.e. python 2.x). the return type of raw\_input is always string .

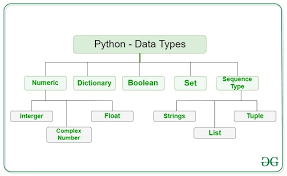
* + 1. input(prompt)

It works in latest version of python. the return type of input need not be string only. Python will judge as to what data type will it fit the best.

In both of the case we have to use type-casting to other numeric data.

Data Handling ):

1. Data-type
2. Mutable and Immutable object
3. Operators
4. Expressions
5. Standard Python Libraries

1.Data-type -Data types are the classification or categorization of data items. Data types represent a kind of value which determines what operations can be performed on that data. 

**There are 5types of data-types in Python**

**i.e.**

1. **Numeric**
2. **Sequence type**
3. **Dictionary**
4. **Boolean**
5. **Set**

**1.Numeric type –**

A number is an arithmetic entity that lets us measure something. Python allows us to store the **integer (e.g. 1,2,3 etc.), floating (e.g. 4.0 ), and complex (like a+1j)** numbers and also lets us convert between them. Since Python is dynamically-typed, there is no need to specify the type of data for a variable**.**

**2.Sequence type –**

Sequences allow you to store multiple values in an organized and efficient fashion. There are several types of Sequence type data-type: -

**String :-**

In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. In Python we can not assign/change/replace a string , only able to append a string with another.

**List:-**

Anything under square-brace , List is a collection which is ordered and changeable and allows duplicate members**.(e.g. A=[1,2,3,4]).**

**Tuple:-**

Anything under parenthesis, Tuple is a collection which is ordered and unchangeable**.(e.g. A=(11,22)).**

**3.Dictionary type –**

A dictionary is a data-structure in python , Which is a collection of key-value pairs, separated by comma , inside curly-braces. All operations of a dictionary is done in a particular address. That means the read-write operation inside a dictionary is doesn’t affect to its address.

Example- Dictionary1 = {“name”:”Krishna”,”roll”:22} , here name and roll are the keys and Krishna and 22 are values of that respective keys.

**4.Boolean –**

In Python,  boolean variables are defined by the True and False keywords. The output <class '**bool**'> indicates the variable is a boolean data type. The keywords True and False must have an Upper Case first letter.

**5.Set –**

In Python, Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements.

2.Mutable and Immutable object –

* **Mutable** – If the value of the object is changed after its creation within a the same address then i.e. Mutable.

Some data types which are following this concept are –

List

Dictionary

Set

* **Immutable** – Here the value of the object is not changeable after creation then .

Some data types which are following this concept are –

Integer

Float

Boolean

String

Tuple

3.Operators –

There are several types of operators in Python:

* Python Arithmetic Operator
* Python Relational Operator
* Python Assignment Operator
* Python Logical Operator
* Python Membership Operator
* Python Identity Operator
* Python Bitwise Operator

**Arithmetic Operators –**

**Addition(+)** – Adds the values on either side of the operator.

**Subtraction(-)** –Subtracts the value on the right from the one on the left.

**Multiplication(\*)** - Multiplies the values on either side of the operator.

**Division(/) -** Divides the value on the left by the one on the right. Notice that division results in a floating-point value.

**Exponentiation(\*\*) -** Raises the first number to the power of the second.

**Floor Division(//) -** Divides and returns the integer value of the quotient. It dumps the digits after the decimal.

**Modulus(%)** - Divides and returns the value of the remainder.

**Relational Operator** - carries out the comparison between operands. They tell us whether an operand is greater than the other, lesser, equal, or a combination of those.

**Less than(<) -** This operator checks if the value on the left of the operator is lesser than the one on the right.

**Greater than(>)** - It checks if the value on the left of the operator is greater than the one on the right.

**Less than or equal to(<=)** - It checks if the value on the left of the operator is lesser than or equal to the one on the right.

**Greater than or equal to(>=)** - It checks if the value on the left of the operator is greater than or equal to the one on the right.

**Equal to(= =)** - This operator checks if the value on the left of the operator is equal to the one on the right. 1 is equal to the Boolean value True, but 2 isn’t. Also, 0 is equal to False.

**Not equal to(!=)** - It checks if the value on the left of the operator is not equal to the one on the right. The Python operator <> does the same job, but has been abandoned in Python 3,When the condition for a relative operator is fulfilled, it returns True. Otherwise, it returns False. You can use this return value in a further statement or expression.

**Assignment Operator –**

An assignment operator assigns a value to a variable. It may manipulate the value by a factor before assigning it. We have 8 assignment operators- one plain, and seven for the 7 arithmetic python operators.

**a. Assign(=)**

Assigns a value to the expression on the left. Notice that = = is used for comparing, but = is used for assigning.

**b. Add and Assign(+=)**

Adds the values on either side and assigns it to the expression on the left. a+=10 is the same as a=a+10.

**c. Subtract and Assign(-=)**

Subtracts the value on the right from the value on the left. Then it assigns it to the expression on the left.

**d. Divide and Assign(/=)**

Divides the value on the left by the one on the right. Then it assigns it to the expression on the left.

**e. Multiply and Assign(\*=)**

Multiplies the values on either sides. Then it assigns it to the expression on the left.

**f. Modulus and Assign(%=)**

Performs modulus on the values on either side. Then it assigns it to the expression on the left.

**g. Exponent and Assign(\*\*=)**

Performs exponentiation on the values on either side. Then assigns it to the expression on the left.

**h. Floor-Divide and Assign(//=)**

Performs floor-division on the values on either side. Then assigns it to the expression on the left.

**Logical Operator –**

These are conjunctions that you can use to combine more than one condition. We have three Python logical operator – and, or, and not that come under [python](https://www.python.org/about/gettingstarted/) operators.

**a. and**

If the conditions on both the sides of the operator are true, then the expression as a whole is true.

**b. or**

The expression is false only if both the statements around the operator are false. Otherwise, it is true.

‘and’ returns the first False value or the last value; ‘or’ returns the first True value or the last value

**c. not**

This inverts the [Boolean value](https://data-flair.training/blogs/python-set-and-booleans-with-examples/) of an expression. It converts True to False, and False to True. As you can see below, the Boolean value for 0 is False. So, not inverts it to True.

**Membership Operator –**

These operators test whether a value is a member of a [**sequence**](https://data-flair.training/blogs/python-sequence/). The sequence may be a **list**, a **string**, or a **[tuple](https://data-flair.training/blogs/python-tuple/)**. We have two membership python operators- ‘in’ and ‘not in’.

**a. in**

This checks if a value is a member of a sequence. In our example, we see that the string ‘fox’ does not belong to the list pets. But the string ‘cat’ belongs to it, so it returns True. Also, the string ‘me’ is a substring to the string ‘disappointment’. Therefore, it returns true.

-> pets=[‘dog’,’cat’,’ferret’]

-> ‘fox’ in pets

Output: False

-> ‘cat’ in pets

Output: True

-> ‘me’ in ‘disappointment’

Output: True

**b. not in**

Unlike ‘in’, ‘not in’ checks if a value is not a member of a sequence.

**Identity Operator –**

Let us proceed towards identity Python Operator.

These operators test if the two operands share an identity. We have two identity operators- ‘is’ and ‘is not’.

**a. is**

If two operands have the same identity, it returns True. Otherwise, it returns False.

**b. is not**

2 is a number, and ‘2’ is a string. So, it returns a True to that.

-> 2 is not ‘2’

Output: True

**Bitwise Operator –**

**a. Binary AND(&)**

It performs bit by bit AND operation on the two values. Here, binary for 2 is 10, and that for 3 is 11. &-ing them results in 10, which is binary for 2. Similarly, &-ing 011(3) and 100(4) results in 000(0).

->2&3

Output: 2

-> 3&4

Output: 0

**b. Binary OR(|)**

It performs bit by bit OR on the two values. Here, OR-ing 10(2) and 11(3) results in 11(3).

-> 2|3

Output: 3

**c. Binary XOR(^)**

It performs bit by bit XOR(exclusive-OR) on the two values. Here, XOR-ing 10(2) and 11(3) results in 01(1).

-> 2^3

Output: 1

**d. Binary One’s Complement(~)**

It returns the one’s complement of a number’s binary. It flips the bits. Binary for 2 is 00000010. Its one’s complement is 11111101. This is binary for -3. So, this results in -3. Similarly, ~1 results in -2.

->~-3

Output: 2

Again, one’s complement of -3 is 2.

**e. Binary Left-Shift(<<)**

It shifts the value of the left operand the number of places to the left that the right operand specifies. Here, binary of 2 is 10. 2<<2 shifts it two places to the left. This results in 1000, which is binary for 8.

-> 2<<2

Output: 8

**f. Binary Right-Shift(>>)**

It shifts the value of the left operand the number of places to the right that the right operand specifies. Here, binary of 3 is 11. 3>>2 shifts it two places to the right. This results in 00, which is binary for 0. Similarly, 3>>1 shifts it one place to the right. This results in 01, which is binary for 1.

Type-Casting in python –

The process of converting the value of one data type to another is called as Type-casting or type conversion. Python is not strongly typecast-able

Standard libraries –

The Python Standard Library is a collection of script modules accessible to a Python program to simplify the programming process and removing the need to rewrite commonly used commands.

Conditional Statements –

**if Statements -**An if statement in python takes an expression with it. If the expression am ounts to True, then the block of statements under it is executed. If it amounts to False, then the block is skipped and control transfers to the statements after the block. But remember to indent the statements in a block equally. This is because we don’t use curly braces to delimit blocks. Also, use a colon(:) after the condition.

**if-else Statements -** What happens when the condition is untrue? We can mention that it in the block after the else statement. An else statement comes right after the block after ‘if’.

**Chained Conditionals (elif ladder) -** Python allows the elif keyword as a replacement to the else-if statements in java or C++. When we have more than one condition to check, we can use it. If condition 1 isn’t True, condition 2 is checked. If it isn’t true, condition 3 is checked.

**Nested if Statements in Python -** You can put an if statement in the block under another if statement. This is to implement further checks.

**Single Statement Condition in Python -** If you only need to write a single statement under if, you can write it in the same line using single statement python decision making constructs.

**range() –**

It is a built-in function of Python. It is used when a user needs to perform an action for a specific number of times. range() in Python(3. x) is just a renamed version of a function called xrange in Python(2).

**Syntax –** range(start ,stop):

**Loops –**

**For loop -**

**While loop -** A while loop in python iterates till its condition becomes False. In other words, it executes the statements under itself while the condition it takes is True.

**Syntax –** while(condition):

**Loop else -** When the condition becomes false, the block under the else statement is executed.

**Syntax –** while(condition):

(Logic)

Else

(logic)

**break statement -**

When you put a break statement in the body of a loop, the loop stops executing, and control shifts to the first statement outside it. You can put it in a for or while loop.

**continue statement -**

When the program control reaches the continue statement, it skips the statements after ‘continue’. It then shifts to the next item in the sequence and executes the block of code for it. You can use it with both for and while loops.

**Tkinter**

**Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.**

**.**