[For Quick Revision]

Value Added Modules (Python)

Python is a high-level, object-oriented programming language  
features of Python:  
 1. Easy to read: Python is known for its clear syntax and code readability.  
 2. Integrates well: Python integrates well with many different systems.  
 3. Extensible: Python can be extended in C or C++.  
 4. Supports multiple programming paradigms: Python supports procedural, functional, and object-oriented programming.  
   
Python was developed by 'Guido van Rossum' in 1991.  
   
Indentation:  
 it defines the way a code should be written  
   
Variable: A variable is a data-item, which stores value, it can be an alphabet or alphanumeric or string, the  
 value of the variable can change according to the code's condition or at runtime execution.  
   
Constant: A constant is a data-item, which value doesn't change and remains the same through-out the code.

**...OPERATORS...**  
   
Operators are special symbols used to perform specific operations on one or more operands.  
Operators are of two types:  
 1. Unary -- which are perform on one operand, such as a++ or ++a, a-- or --a.  
 2. Binary -- which are perform on two or more operands, a+b or a-b.  
   
 **...Types of Operators...**  
   
1. Arithmetic -- +, -, /, %, //, \*\*  
 2. Logical -- and, or, not  
 3. Bitwise -- &, |, ^, ~  
 4. Assignment -- +=, -=, \*=  
 5. Comparison -- <, >, <=,>=, ==  
 6. identity -- is, is Not  
 7. Membership -- in, not in

**Bitwise Operator:**

Python bitwise operators are used to perform bitwise calculations on integers. The integers  
are first converted into binary and then operations are performed on each bit or corresponding  
pairs of bits. Hence, they are called bitwise operators and the result displayed as decimal.  
  
AND -> a&b [same bits = 1 or 0, different bits = 0]  
OR -> a|b  
NOT -> a^b  
XOR -> ~a

Arithmetic Operations

**Addition (+)**

Adds two numbers together.

a = 5

b = 3

result = a + b # result is 8

**Subtraction (-)**

Subtracts the second number from the first.

a = 5

b = 3

result = a - b # result is 2

**Division (/)**

Divides the first number by the second. The result is a float.

a = 5

b = 2

result = a / b # result is 2.5

**Modulus (%)**

Returns the remainder of the division.

a = 5

b = 2

result = a % b # result is 1

**Floor Division (//)**

Divides the first number by the second and returns the largest integer less than or equal to the result.

a = 5

b = 2

result = a // b # result is 2

**Exponentiation (\*\*)**

Raises the first number to the power of the second.

a = 5

b = 2

result = a \*\* b

Logical Operations

**and (Logical AND)**

Returns True if both operands are true.

a = True

b = False

result = a and b # result is False

If both a and b were True, the result would be True.

**or (Logical OR)**

Returns True if at least one of the operands is true.

a = True

b = False

result = a or b # result is True

If both a and b were False, the result would be False.

**not (Logical NOT)**

Inverts the boolean value.

a = True

result = not a # result is False

If a was False, the result would be True.

**Assignment Operation**

**+= (Add and Assign)**

Adds the right operand to the left operand and assigns the result to the left operand.

a = 5

a += 3 # Equivalent to a = a + 3, now a is 8

**-= (Subtract and Assign)**

Subtracts the right operand from the left operand and assigns the result to the left operand.

a = 5

a -= 3 # Equivalent to a = a - 3, now a is 2

**\*= (Multiply and Assign)**

Multiplies the left operand by the right operand and assigns the result to the left operand.

a = 5

a \*= 3

Comparison Operators

**< (Less Than)**

Returns True if the left operand is less than the right operand.

a = 5

b = 10

result = a < b # result is True

**> (Greater Than)**

Returns True if the left operand is greater than the right operand.

a = 5

b = 10

result = a > b # result is False

**<= (Less Than or Equal To)**

Returns True if the left operand is less than or equal to the right operand.

a = 5

b = 5

result = a <= b # result is True

**>= (Greater Than or Equal To)**

Returns True if the left operand is greater than or equal to the right operand.

a = 10

b = 5

result = a >= b # result is True

**== (Equal To)**

Returns True if the operands are equal.

a = 5

b = 5

result = a == b

Identity Operators

**is (Identity Operator)**

Checks if two variables point to the same object in memory.

a = [1, 2, 3]

b = a

result = (a is b) # result is True

Both a and b point to the same list object.

**is not (Identity Operator)**

Checks if two variables do not point to the same object in memory.

a = [1, 2, 3]

b = [1, 2, 3]

result = (a is not b)

Membership Operators

**in (Membership Operator)**

Checks if a value or variable is present in a sequence (like a list, tuple, dictionary, set, or string).

fruits = ['apple', 'banana', 'cherry']

result = 'banana' in fruits # result is True

**not in (Membership Operator)**

Checks if a value or variable is not present in a sequence.

fruits = ['apple', 'banana', 'cherry']

result = 'grape' not in fruits

...STRING...

Immutable Datatype i.e. not changeable  
 Collection of immutable characters.  
 -> A String is a data structure in python programming language  
 Which represents a sequence of characters.  
   
**...String Methods...**

1. Capitalise ()  
 2. Casefold ()  
 3. Find ()  
 4. Replace ()  
 5. Max ()  
 6. Min ()  
 7. Index ()  
 8. Translate ()  
 9. Swapcase ()  
 10. Numbers ()  
 11. Uppercase ()  
 12. Lowercase ()

13. Isspace ()

...LIST...   
   
  
1. List are used to store multiple items/values in a single variable.  
2. List items are ordered changeable and allow duplicate values.  
   
**...ORDERED...**  
   
1. It means that items have a defined order and that order will not change.  
2. If you add a new item to the list, it will be placed at the end of the list.  
   
  
**...METHODS OF LIST...**  
   
1. Append () -- adds an element.  
 2. Clear () -- clear the whole list / remove all elements.  
 3. Copy () -- copies the content of one list to another list if another list is empty.  
 4. Count () -- count the elements of the list.  
 5. Extend () -- working on a single list from multiple list  
 6. Index () -- it gives the total index value of the list.  
 7. Insert () -- it inserts an element on a particular index.  
 8. Pop () -- removes a single element.  
 9. Remove () -- it removes an element when we know the element name.  
 10. Reverse () -- it changes the sequence of the list.  
 11. Sort () -- it sorts the data in ascending or descending order.  
   
  
 **...TUPLE...**  
   
1. A tuple is a collection of data items, which is ordered and unchangeable  
 and allow duplicate different values.  
   
  
**...Methods of Tuple...**   
1. Count ()  
2. Index ()  
   
 ...SET...  
   
1. A set is a collection of data items which is un-index, unordered, unchangeable and  
 it does not allow duplicate values.  
2. Once a set is created you cannot change its items but you can add or remove an item

**...IF-ELSE Statements...**

When we have to check a condition true or false then we apply conditional statements on it  
   
1. If () -- These are used to execute different blocks of code based on condition.  
   
2. Elif () -- The elif keyword is python way of saying, if the previous condition were not true  
 then try this condition.  
   
3. Else () -- The else keyword catches anything which is not which is not caught by  
 the preceding condition.  
   
 **...Short-hand if...**  
   
If you have only one statement to execute you can put it on the same line as the if statement.  
 syntax: if a>b: print("hello")  
   
 **...LOOPS...**  
   
**while loop** -- with the help of while loop we can execute a set of statements  
as long as the condition is true.

Syntax: while condition:

# code to be executed

**...FOR LOOP...**

A for loop in Python is a control flow statement that allows code to be executed repeatedly, iterating over a sequence (like a list, tuple, dictionary, set, or string).

Syntax => for <var-name> <membership> list:  
 print(<var-name>)  
   
 **...FUNCTIONS...**  
   
1. The main use of function comes in when we want to execute the same set of code, multiple times across our program.   
2. The main advantage of function are:  
 a. they break code into smaller modularity.  
 b. redundancy is avoided repeating the same set of code multiple times across the program is avoided by the function.  
   
**...Built-in Functions...**  
   
 Mathematical Function:  
 a. pow ()  
 b. sum ()  
 c. max ()  
 d. min ()  
 e. round ()

**…User-Defined Functions…**

**Syntax:**

def <func\_name> (arg1, arg2, arg3)

def area (l**,** b):  
 area1 = l\*b  
 print(area1)

OOPS

Object Oriented Programming Concepts

Class, Objects, Constructors: Define, Calling, Structure, Properties

**Classes and Objects: -**

**Class: -** A class is a collection of objects; a class contains the blueprint or the prototype from which the objects are being created it is a logical entity that contains some attributes and methods.

* Classes are created by keyword class.
* Attributes are the variables which belong to the class.
* Syntax: class student  
   #statement1  
   #statement2

**Objects: -**

The object is an entity that has a state and behaviour associated with it. It may be any real-world object like a keyboard and mouse. An object consists of state, behavior and identity.

State- It is represented by the attribute of the objects and it also reflects the properties of the objects.

Behavior- It is represented by the methods of the object. It also reflects the response of the object to another object.

Identity- It gives the unique name to an object and enables one object to interact with another object.

Inheritance

Class A:

\_\_\_\_\_  
\_\_\_\_\_  
  
Class B(A):  
\_\_\_\_\_  
\_\_\_\_\_

Inheritance is a property of OOPS in which one class can inherit the methods of other class.

Polymorphisms

Method Overloading: -

Multiple methods with the same name but different parameters.

Method Overwriting: -

A child class provides a different implementation of a method in the parent class.

Encapsulation

**Public Members: -**

Accessible from anywhere.

**Private members: -**

Accessible only within the class.

Abstraction

Abstract Classes: -

Cannot be initialised only inherited form

Abstract Methods: -

Must be implemented by the child classes

Composition

One Object contains another object

Exception Handling

06Nov2024 TRY: Except:

LIST v/s ARRAYs

Array: 50x faster than list and is contiguous

Libraries:

NumPy

Methods: -

**Transpose** (same as matrix transpose)

**Min** (retrieves smallest number)

**Max** (retrieves largest number)

**Sum** (performs addition)

**Multiply** (performs multiplication)

**Mean** (Calculates the Average)

**Reshape** (Transpose + opposite of it is also possible i.e. Row -> Column and vice-versa)

**Resize** (Used to resize array)

**Dimension** (Gives the dimensions of the matrix)

**Insert, append, pop**

**SLICING in Python (7/11/2024)**

MACHINE LEARNING (7/11/2024)

**DATA -> MODEL -> AI**

**DATA Cleaning**

**Label Data/set**

TYPES: -

1. Supervised M.L
2. Unsupervised M.L
3. Reinforcement M.L

PANDAS (Python Library)

Panel Data (PANDAS)

1. Series (Single dimension)
2. Data Frame (2/multi-D)
3. Dimension

**Series**: A one-dimensional array-like object for holding data.

**Data Frame**: A two-dimensional, size-mutable, and potentially heterogeneous tabular data structure with labelled axes (rows and columns).

Linear Regression

Logistic Regression

Polyminal Regression

Random Forest Regression

Support Model Regression

Decision Tree

[I.D Tree Algo]

* Calculate/Find the Root Node based on the data.
* Calculate/Find the sub-nodes.
* Calculate/Find the entire entropy of data-set.

Info Gain= (mathematical formula)

* Handle Leaf Node
* Pruning (removal of colluding data)

Machine Learning Notes

1. Introduction to Machine Learning

- Machine learning is a field of artificial intelligence (AI) that enables systems to learn and make decisions without explicit programming.

2. Types of Machine Learning

- Supervised Learning: Labelled data is used for training.

- Unsupervised Learning: No labels; patterns are identified in the data.

- Reinforcement Learning: Agents learn by interacting with an environment and maximizing rewards.

3. Supervised Learning

- Regression: Predict continuous values.

Example: Linear Regression and Logistic Regression.

- Classification: Predict discrete values or categories.

Example: Random Forest and Support Vector Machines (SVM).

4. Random Forest

- Random Forest is a supervised learning algorithm used for both classification and regression tasks.

- It creates multiple decision trees and combines their output (averaging for regression, majority voting for classification).

- Reduces overfitting, improves accuracy, and works well with large datasets.

5. Random Forest: Key Points

- Bagging: Uses random samples of data.

- Random Feature Selection: Splits nodes based on random subsets of features.

- Combines outputs from multiple trees.

6. Random Forest: Python Implementation

Classification Example:

```python from sklearn.ensemble import RandomForestClassifier rf = RandomForestClassifier() rf.fit(X\_train, y\_train) y\_pred = rf.predict(X\_test)

```

Regression Example:

```python from sklearn.ensemble import RandomForestRegressor rf = RandomForestRegressor()

rf.fit(X\_train, y\_train) y\_pred = rf.predict(X\_test)

```

7. Advantages of Random Forest

- Handles missing data effectively.

- Works well with both small and large datasets.

- Provides feature importance scores.

8. Applications of Random Forest

- Medical Diagnosis: Disease prediction.

- Fraud Detection: Identifying anomalies in transactions.

- Customer Segmentation: Grouping similar customers.