**Python doccument**

1. **What is python?**

* Python is a popular and fast growing programming language, high level, interpreted and easy to read.
* It was created by Guido van Rossum and released in 1991.
* It is open source including procedural, object oriented and functional programming.
* Python have multiple advanced application.

It is use for Web development ,Software development ,Mathematics, System scripting, Data science, Machine learning etc….

1. **What are Data types in python?**

Data types means information have different types like string type, integer etc..

It have different data types

1.Numeric type: int, float, complex.

Int: it has store only integer values.

Ex: a=10,b=5

Float:it has store data in the form of decimal values.

Ex:a=10.2,b=1.1

Complex: it have store data like

Ex: a=2+8j,b=1+4i.

2.Text: str.

3. Sequence types: List, tuple, range.

**List:** To store ordered collection of value of any type.

It is mutable.

In list we can store different types of values like strings, integers, float values.

Syntax: listname = [element1, element2, element3]

EX: h1=[4,5,6,’hari’,6.8]

**Function of list:** len(),count(),index(),max(),min(),sum(),clear(),copy(),append(),insert(),exstend(),pop(),

Sort(),sorted()

**Tuple:** To store ordered collection of values of any types.

It is immutable.

Once tuple is create we can`t change the values and it has fixed size.

It is define by ().

Syntax: tuplename = (element1, element2, element3)

Ex: L1= (1, "krishna", 3.14, True)

**Operation:** concatenation(),repletion(),indexing(),slicing()

**Function of tuple**: len(),count(),copy(),max(),sum(),index(),sorted()

**Range:** range() function used to generate sequence of numbers and it commonly used in loop mostly used in for loop.

It is immutable and we should convert data to list or tuple.

Syntax: range(start, stop, step)

Ex:

num = range(1, 10, 2)

# Generates 1, 3, 5, 7, 9

print(list(num)) # Output: [1, 3, 5, 7, 9]

**3.Mapping type:** dict.

**set: i** n python is set of unordered collection of Unique items no duplicates are allowed in set .

* The set has itself is mutable. We can add or remove items from
* Set can be used to perform mathematical set operation like union, Intersection, symmetric difference etc.

EX:s1={4,7,’s’,”hari”,7.8}

EX2: s2=set([‘s’,’mango’,4,6,’apple’])----set constructor.

**Set function:**

1.Add():add an element to set

2.Clear():remove all elements from a set.

3.Copy(): return show copy of set.

4.Difference():return difference two or more sets as new set.

4.Update():to insert elements into a set elements are immutable.

5.Discard():It is remove a specific element from the set if the element is not available it does not effect on the set.

6.Intersection():return intersection of two sets as a new set.

7.Isdisjoin():return true if two set have null intersection.

8.Issubset():return true if another set contains this set.

9.Pop():it remove the element which placed as 1st element.

10.Remove():it remove a specific element from the set if the element is not available it raises an exception key error.

11.Symmetric\_difference():return the symmetric difference of two sets as a new set.

12.Union():it will combine set one or more set and it will return the unique element.

13.Update():update a set with the union of itself and others.

14.Len():return the length of set.

15.max():return largest element in the set.

16.sorted():it return new sorted element in set.

* #set example  
  hari={1,2,23,4,2}  
  k=set(hari)  
  print(k)  
  # output{1,2,23,4}  
  # Add()  
    
  my = {1, 2, 3}  
  my.add(4)  
  print(my)  
  #output: {1, 2, 3, 4}  
  # Clear()  
    
  my1 = {1, 2, 3}  
  my1.clear()  
  print(my1)  
  # output: set()  
    
  # Copy()  
  a = {1, 2, 3}  
  new = a.copy()  
  print(new) # output: {1, 2, 3}  
    
  # difference  
  set1 = {1, 2, 3}  
  set2 = {3, 4, 5}  
  print(set1.difference(set2)) # output: {1, 2}  
    
  # Discard()  
  my1 = {1, 2, 3}  
  my1.discard(2)  
  my1.discard(4)  
  print(my1)  
  # Output: {1, 3}  
    
  # Intersection()  
  set1 = {1, 2, 3}  
  set2 = {2, 3, 4}  
  print(set1.intersection(set2))  
  # 0utput: {2, 3}  
  # isdisjoint  
  set1 = {1, 2, 3}  
  set2 = {4, 5, 6}  
  print(set1.isdisjoint(set2))  
  # Output: True  
  # pop()  
  hari = {1, 2, 3}  
  element = hari.pop()  
  print(element)  
  # Output: 1  
  print(hari)  
  # Output: {2, 3}  
  # remove()  
  my\_set = {1, 2, 3}  
  my\_set.remove(2)  
  print(my\_set)  
  # Output: {1, 3}  
    
  # Symmetric difference()  
  set1 = {1, 2, 3}  
  set2 = {3, 4, 5}  
  print(set1.symmetric\_difference(set2))  
  # Output: {1, 2, 4, 5}  
  # union  
  set1 = {1, 2, 3}  
  set2 = {3, 4, 5}  
  print(set1.union(set2))  
  # Output: {1, 2, 3, 4, 5}  
    
  # Update()  
  set1 = {1, 2, 3}  
  set2 = {3,4, 5}  
  set1.update(set2)  
  print(set1)  
  # Output: {1, 2, 3, 4, 5}  
  # length  
  b = {1, 2, 3, 4}  
  print(len(b))  
  # Output: 4  
  # max()  
  krishna = {10, 20, 5, 7}  
  print(max(krishna))  
  # Output: 20  
    
  # sorted()  
  j = {3, 1, 4, 2}  
  print(sorted(j))  
  # Output: [1, 2, 3, 4]

**Dictionary:**

* Dictionary is data type data can store in the form of key value pair.
* Dictionary items are ordered, changeable, and do not allow duplicates.
* Key should be immutable.
* Value should be mutable it has duplicate values and different type of data.
* Key will act as index .
* No slicing because of it has no index.
* Key are unique.

EX: {}

Ex: a={‘a’:123,1:’abc’};

**Dictionary functions:**

1.Get():to retrieve the value of a particular key, we will use get(),if there is no key available get() will return none.

2.Update():we can insert new pair of key and value or update the existing key with a new value.

3.Pop(): In dictionary deal with key.

4.popItem(): deletes the last insert element in the dictionary.

5. All the key are retrieved from the dictionary as addict.values object. We can type cast list also.

6.Keys():All the key are retrieved from the dictionary as addict.key object. We can type cast list also.

7.Items():All the keys and values are retrieved from the dictionary as items object we can typecast to list also.

* # dictionary  
  a= {'a':123 , 1:'abc'}  
  print(a)  
    
    
  # get()  
  my = {'a': 1, 'b': 2, 'c': 3}  
  print(my.get('b'))  
  # Output: 2  
  print(my.get('d'))  
  # Output: None  
    
  # update()  
  k = {'a': 1, 'b': 2}  
  k.update({'b': 10, 'c': 3})  
  print(k)  
  # Output: {'a': 1, 'b': 10, 'c': 3}  
  # pop()  
    
  hari = {'a': 1, 'b': 2, 'c': 3}  
  value = hari.pop('b')  
  print(value)  
  # Output: 2  
  print(hari)  
  # Output: {'a': 1, 'c': 3}  
    
  #popitem()  
  h = {'a': 1, 'b': 2, 'c': 3}  
  key\_value\_pair = h.popitem()  
  print(key\_value\_pair)  
  # Output: ('c', 3)  
  print(h)  
  # Output: {'a': 1, 'b': 2}  
    
  #values()  
  my\_dict = {'a': 1, 'b': 2, 'c': 3}  
  values = my\_dict.values()  
  print(list(values))  
  # Output: [1, 2, 3]  
    
  #keys()  
  my\_dict = {'a': 1, 'b': 2, 'c': 3}  
  keys = my\_dict.keys()  
  print(list(keys))  
  # Output: ['a', 'b', 'c']  
    
  #items()  
  my\_dict = {'a': 1, 'b': 2, 'c': 3}  
  items = my\_dict.items()  
  print(list(items))  
  # Output: [('a', 1), ('b', 2), ('c', 3)]

6.Boolean type: bool

Bool: to store value either True or false.

EX: X=True.

Y=False.

7.Binary types: bytes, byte array

Bytes: to store collection of numbers in the range 0-256.

It is mutable.

Ex: b=[4,3,1,87,98]

Byte Array: same like bytes as to store collection on of numbers in the range 0-256.

It is mutable.

Ex:b=[4,3,1,87,98]

**Operators in Python**

In Python, operators are symbols used to perform operations on variables and values. Python supports several types of operators:

**1. Arithmetic Operators**

These operators perform basic mathematical operations like addition, subtraction, multiplication, etc.

* + (Addition): Adds two operands.
* - (Subtraction): Subtracts the second operand from the first.
* \* (Multiplication): Multiplies two operands.
* / (Division): Divides the first operand by the second.
* // (Floor Division): Divides and returns the largest integer smaller than or equal to the result.
* % (Modulus): Returns the remainder when the first operand is divided by the second.
* \*\* (Exponentiation): Raises the first operand to the power of the second.
* # 1. Addition (+)  
  a = 5  
  b = 3  
  result = a + b  
  print(result) # Output: 8  
    
  # 2. Subtraction (-)  
  a = 5  
  b = 3  
  # result = a - b  
  print(result) # 5-3 Output: 2  
    
  # 3, Multiplication (\*)  
  a = 5  
  b = 3  
  # result = a \* b  
  print(result) # Output: 15  
    
  # 4. Division (/)  
  a = 5  
  b = 2  
  result = a / b  
  print(result) # Output: 2.5  
    
  # 5. Floor Division (//)  
  a = 5  
  b = 2  
  result = a // b  
  print(result) # Output: 2  
    
  # 6. Modulus (%)  
  a = 5  
  b = 2  
  result = a % b  
  print(result) # Output: 1

# 7. Exponentiation (\*\*)  
a = 2  
b = 3  
# result = a \*\* b  
print(result) # Output: 8

**2. Comparison (Relational) Operators**

These operators compare two values and return a boolean result (True or False).

* == (Equal to): Returns True if the values of two operands are equal.
* != (Not equal to): Returns True if the values of two operands are not equal.
* > (Greater than): Returns True if the first operand is greater than the second.
* < (Less than): Returns True if the first operand is less than the second.
* >= (Greater than or equal to): Returns True if the first operand is greater than or equal to the second.
* <= (Less than or equal to): Returns True if the first operand is less than or equal to the second.

# 1. Equal to (==)

a = 5

b = 5

result = a == b

print(result) # Output: True

# 2. Not equal to (!=)

a = 5

b = 3

result = a != b

print(result) # Output: True

# 3. Greater than (>)

a = 5

b = 3

result = a > b

print(result) # Output: True

# 4. Less than (<)

a = 5

b = 7

result = a < b

print(result) # Output: True

# 5. Greater than or equal to (>=)

a = 5

b = 5

result = a >= b

print(result) # Output: True

# 6. Less than or equal to (<=)

a = 5

b = 7

result = a <= b

print(result)

**3. Logical Operators**

These operators are used to combine conditional statements.

* and: Returns True if both operands are True.
* or: Returns True if at least one operand is True.
* not: Reverses the logical state (i.e., returns True if the operand is False, and vice versa).

# 1. AND+

a = True

b = False

result = a and b

print(result) # Output: False

# 2. OR

a = True

b = False

result = a or b

print(result) # Output: True

# 3.NOT

a = True

result = not a

print(result) # Output: False

**4. Assignment Operators**

These operators are used to assign values to variables.

* =: Assigns the value of the right operand to the left operand.
* +=: Adds the right operand to the left operand and assigns the result to the left operand.
* -=: Subtracts the right operand from the left operand and assigns the result to the left operand.
* \*=: Multiplies the left operand by the right operand and assigns the result to the left operand.
* /=: Divides the left operand by the right operand and assigns the result to the left operand.
* //=: Floor divides the left operand by the right operand and assigns the result to the left operand.
* %=: Takes the modulus of the left operand by the right operand and assigns the result to the left operand.
* \*\*=: Raises the left operand to the power of the right operand and assigns the result to the left operand.
* # 1. Assign (=)  
  x = 5  
  print(x) # Output: 5  
  # 2. Add and Assign (+=)  
  x = 5  
  x += 3  
  print(x) # Output: 8  
  # 3. Subtract and Assign (-=)  
  x = 5  
  x -= 3  
  print(x)  
  # Output: 2  
  # 4. Multiply and Assign (\*=)  
  x = 5  
  x \*= 3  
  # 5. Divide and Assign (/=)  
  x = 10  
  x /= 2  
  print(x) # Output: 5.0  
  # 6. Modulus and Assign (%=)  
  x = 10  
  x %= 3  
  print(x) # Output: 1

**5. Bitwise Operators**

Bitwise operators are used to perform operations on binary numbers.

* & (AND): Performs a bitwise AND.
* | (OR): Performs a bitwise OR.
* ^ (XOR): Performs a bitwise XOR (exclusive OR).
* ~ (NOT): Performs a bitwise NOT (inverts the bits).
* << (Left shift): Shifts the bits to the left by the specified number of positions.
* >> (Right shift): Shifts the bits to the right by the specified number of positions.

# 1. AND (&)

a = 5 # In binary: 0101

b = 3 # In binary: 0011

result = a & b # Performs bitwise AND

print(result) # Output: 1 (In binary: 0001)

# 2. OR (|)

a = 5 # In binary: 0101

b = 3 # In binary: 0011

result = a | b # Performs bitwise OR

print(result) # Output: 7 (In binary: 0111)

# 3. XOR (^)

a = 5 # In binary: 0101

b = 3 # In binary: 0011

result = a ^ b # Performs bitwise XOR

print(result) # Output: 6 (In binary: 0110)

# 5. Left Shift (<<)

a = 5 # In binary: 0101

result = a << 1 # Shifts bits of a to the left by 1

print(result) # Output: 10 (In binary: 1010)

# 6. Right Shift (>>)

a = 5 # In binary: 0101

result = a >> 2 # Shifts bits of a to the right by 1

print(result) # Output: 2 (In binary: 0010)

**6. Membership Operators**

These operators test whether a value is in a sequence (like a list, tuple, string, etc.).

* in: Returns True if the value is found in the sequence.
* not in: Returns True if the value is not found in the sequence.

# Membership operators

a = [1, 2, 3, 4, 5]

print(3 in a) # True

print(6 not in a) # True

**7. Identity Operators**

These operators compare the memory locations of two objects.

* is: Returns True if both operands refer to the same object in memory.
* is not: Returns True if both operands refer to different objects.

# 1. is

a = [1, 2, 3]

b = a

result = a is b

print(result) # Output: True

a = [1, 2, 3]

b = [1, 2, 3]

result = a is b

print(result) # Output: False

# 2. is not

a = [1, 2, 3]

b = [1, 2, 3]

result = a is not b

print(result) # Output: True

a = [1, 2, 3]

b = a

result = a is not b

print(result) # Output: False

**Conditional statement and Control statement**

Conditional statement: conditional statement are also known decision making statement we need to use these conditional statement to execute the specific block of code if the given conditional is true or false.

Conditional statement:

1.if statement

2.if else statement

3.nested if statement

4.if el if ladder.

1.If statement: If statement is most simple decision making statement when the condition is true then curser enter in the block of code is executed other wise it does not executed.

Syntax: if (condition).

2.If else statement: If else statement is also decision making but it has when condition is true curser enter block of code is executed .while condition is false it will simply executed else block statement.

Syntax if(condition):

Print()

Else:

Condition is false

3.Nested if statement: netsted if statement means an if statement in side another if statement . we can place an if statement in side another if statement.

Syntax: if(conditon1):

If(conditon2):

Executed statement

Print(“pass”)

4.if elif- else: If elif-else will help in decide among multiple options the if statement are executed from the top down if a condition is true the statement associated with is true then the final element will be executed.

Syntax: if(condition):

Statement

Elif(condition):

Statement

Else:

Statement

# If statement.  
number = 10  
if number > 0:  
 print("The number is positive.")  
# if else  
number = 7  
  
if number % 2 == 0:  
 print("The number is even.")  
else:  
 print("The number is odd.")  
# Nested if statement  
number = 0  
  
if number >= 0:  
 if number == 0:  
 print("The number is zero.")  
 else:  
 print("The number is positive.")  
else:  
 print("The number is negative.")  
# if-elif-else  
a=25  
if a<20:  
 print("hari")  
elif a>21:  
 print("krishna")  
else:  
 print("default")

**output**

The number is positive.

The number is odd.

The number is zero.

krishna

Control statement: it is used to control the statement.

There are different types of control statement.

1. Continue.

2. Break

3. Pass

Continue: it is use to skip the current execution of statement and it will continue next statement iteration.

Break: it use break condition at least one condition is true loop will be terminated and pull the out from block of statement.

Pass: it is just pass the execution and start executing the statement after the pass statement.

**Loops**

Loops means the block of code execute multiple times iterating for met the specific condition.

Loops have different types

1.For loop: The for loop iterative over sequence and execute block of code for each element .

It has particular range is follow.

**Syntax: for element in sequence:**

**# Code block to execute**

3.While loop: the while loop executes a block of code as long as a condition is true.

If do not the range of values then it will use the while condition.

**Syntax: while condition:**

**Keywords**

Keyword are reserved words that have specific meanings and purpose .

It have 36 keywords in python.

1. false: Boolean literal.

2. None: special content.

3. True: Boolean literal.

4. and: logical operator.

5. as: alias and context management.

6. assert: Debugging aid.

7. async: Asynchronous programming.

8. await: Asynchronous programming.

9. break: loop control.

10. class: Class definition.

11. continue: looping control

12. def: function definition.

13. del: deletes a variable or object

14. elif: conditional branching

15. else: conditional branching.

16. except: Exceptional handing.

17. finally: Exception handing.

18. for: looping.

19. from: importing modules.

20. global: Declares global scope.

21. if: conditional branching.

22. import: Importing modules.

23. in: Membership testing or iteration

24. is: identity testing.

25. lambda: Anonymous function.

26. Nonlocal: Declares non-local scope.

27. not: logical operator.

28. or : logical operator.

29. pass: placeholeder.

30. raise: Exception raising.

31. return: return function.

32 .try: exception handling.

33. while: looping.

35. with: context management.

36. yield: generator function.

**Functions**

Function: A function is block of code to perform a specific task.

Type of function

There are two types of function

1.Standered liberary function: these are built-in function in python that are available.

2.User-defined function: we can create function our own function based on our requirement.

**Python function declaration:**

1.def-keyword used to declare a function.

2.Function name: any name give to the function name.

3.arguments: any value passed to function.

4.return(optional)- return value from a function.

Syntax:

Def function\_name(argument):

# function body

Return

Ex: def hello():

Print(‘hello’)

**Calling function**

To call a function use the function name followed by parenthysis.

EX: def hello():

Print(“hello world”)

Hello()

Print(“thank you”)

Output:

Hello world.

Thank you.

**Function with arguments**

Argument are specified after the function name inside the parentheses.

If you want many type of argument separated with comma.

Ex:

# with out argument.

Def add():

Print(“hari”)

Add()

Ex: function with argument

Def add(num1,num2):

Sum=(num1+num2)

Print(sum)

Add(5,6)

Output:11

**Function with argument:**

EX: def my(fname,lname):

Print(Fname+” ”+Lname)

My(“hari”,”Krishna”)

Output: hari Krishna

**arbitrary arguments, \*args**

if you do not know how many arguments that will be passed into your function, add a \* before the parameter name in the function definition.

**def my(\*kids):**

**print("The youngest child is " + kids[2])**

**my("Emil", "Tobias", "Linus")**

**output: The youngest child is** **Linus** The youngest child is Lin**Recursion:** function calling itself is called recursion.

There are two types

1.Base case-->terminating from recursion.

2.recursion case--> calling itself.

EX:

def factorial(n):  
 if (n==0 or n==1):  
 return 1  
 else:  
 return n\*factorial(n-1)  
n=int(input("enter n values:"))  
res=factorial(n)  
print(res)

output: enter n values:4

24

**Lambda functions:**

* A lambda function is a small anonymous function.
* Do not defined by def keyword.
* Return expression but not values.
* One-line function.
* Any number of argument.
* Can not access global variable.
* Defined by using “lambda” keyword.
* Function does not have any name.l

**syntax: Lambda argument: expression.**

sum=lambda x,y: x+y  
print("sum=",sum(2,3))

output: sum= 5

**Modules and Packages**

Before entering into the topic let’s have a brief idea on folders and files so in folders, we store subfolders and files in files we have content.

So, lets us assume modules as files and folders as packages.

**Modules:**

Module is nothing but a file that contains python code in that in form of functions classes and variables which we can use in other programs.

Modules are used to reuse code.

Any file with .py extension is called as a module.

So here assume we are developing a big project ABCD so we divide this into module A,B,C,D so we will be using this modules in another project if it is needed so code reusability is a very important thing in programming.

In modules we have 2 types of modules:

* User defined modules
* Built-in modules

**Built-in modules :**

The modules which are already predefined in python

So lets have a look on some built-in modules:

1.Math

2. Random

3. Datetime

These are few predefined modules in python let’s see with one simple examples

Import math

Print(math.pi) #output: 3.141592653589793

import random

print(random.randint(1, 10)) # Output: Random integer between 1 and 10

import datetime

now = datetime.datetime.now()

print(now) # Output: Current date and time

**Userdefined modules :**

The modules which are created by programmers is called userdefined modules

Here I have defined some functions in the module file

def add(a,b):  
 return a+b  
def sub(a,b):  
 return a-b  
def mul(a,b):  
 return a\*b  
def div(a,b):  
 return a/b

So I need to do some operations on add.sub,mul and div to do these operations no need to do the coding from the scratch so just I will import those modules.

from module import \*

now just I will do the operations which ever I needed.

a = 9  
b = 7  
c = add(a,b)  
print(c) # 16  
c = sub(a,b)  
print(c) # 2  
c= mul(a,b)  
print(c) # 63  
c = div(a,b)  
print(c) # 1.2857142857142858

**Package**

Package is the collection of modules and subpackages

Each package contains a special file called as \_\_init\_\_.py

When a directory contains a \_\_init\_\_.py file python treats the directory as a package allowing u to import modules and sub packages.

In python we have larger developer community so code which is written by the developer he will add that package to the Pypl

So knowing about the pypl(python package index) we call this pypl as a repository more than 2 lakh packages are present in this repository so we will install those packages from this repository with the help of pip we can say this as package manager we can install, uninstall, search , upgrade, list operations in with this pip.

**Strings**

String is a group of characters or collection of characters it given by a data type called str.

**Operations**

1.concatenation(+): it concatenate one string to another string.

2.Repetition(\*):

3.Indexing([]): it space is also a valid charater

Index starts from ‘o’ and ends with ‘n-1’.

Negative indexing from ‘1’ and ends with ‘n+1’.

Negative indexing ends with –(n).

4.Slicing([]):

Str\_ var[start:stop:step]

**String function:**

1.len(): it given length of sting(no of charaters in string)

Syntax: int len(str\_var)

Ex: s=’hari’

2.center (): it aligns the string in center in a larger width

Syntax: str\_var.center(width,delimiter character)

Strvar.center(width)

3.ljust(): it aligns the string in left in large width .

Strvar.ljust()

4.rjust():it is aligns the string in right in a large width

Strvar.rjust()

5.Zfill(): it is same as rjust() remaining are filled with space or zero.

6.upper(): it convert the lower case letters of the string to uppercase latters.

7.Lower(): it convert the upper case letters of the string to lowercase latters.

8.casefold():it converts the uppercase latters to lower case.

9.capitalize(): it convert the 0th index lower case alphabet to uppercase.

10.swapcase(): it reverse the case of the string charaters(lower to upper or upper tolower)

11.title() it converts every words starting alphabet to uppercase.

12.find():it gives the first occurance of the specified pattern in the given string find() works from left to right.

Searching pattern not available it return -1.

Syntax: strvar.find(pattern,startindex)

13.index():it gives the last ocurance of the specified pattern in the given string index() works from left to right.

If the searching pattern is not available it raises values error.

14.rfind:it given the last occurance of the specified pattern the given rfind works from the right to left. It return -1.

Syntax: strvar.rfind(pattern,startindex)

15.rindex(): it gives the last occurance rindex() work from right to left.

Syntax: starver. rindex (pattern, startindex)

16.count(): it gives the frequency of the specified pattern. In the given string.

Syntax: int strvar.count (pattern)

17.lstrip(): it use to remove the blank space.

Syntax: strvar.lstrip()

It use to remove spaces the before the string

18.Rstrip():it use to remove the black space right side of string.

Syntax:strvar.rstrip(pattern)

19.strip():it remove space both sides.

Syntax: strvar.strip()

20.split():it will split string into individual parts strings based on specified.

21.rsplit():works from right to left.it will split right side string.

22.splitline(): it works on ‘\n’or ‘\v’

23.isalpha(): it return ture if the string contains only alphabet otherwise it return false

Ex: a=input()

If a.isalpha():

Print(‘alphabet’)

Else:

Print(‘not alphabet’)

Output:

Abc kelmop

Alphabet

24.is alnum(): it return ture if the string contains only either alphabet or digit otherwise it return false.

A=input()

Print(a.isalnum())

Output:

Abcd12

True

25.isdentifiers(): it return true string contains a valid identifier otherwise.

Ex: a=’8d’

Print(a.isidentifier())

A=’-8d’

Print(a.isidentifier())

Output: false

True

26. isprintable(): 0-31,127 are non printable characters it return true if the string contains printable characters otherwise false .

For I in range(32):

Print(chr(i).isprintable(),end=’ ’)

Print()

For in range(32,127)

Print(chr(i.isprintable,end=’ ‘)

28.isascii(): it return true if the string characters with valid ascii values otherwise it returns false.

29.isdigit(): it return if the string contains digit or charater digit otherwise it return false.

30.isdecimal(): it return ture if the string contains digits otherwise it return false.

31.isnumeric():it return true if the string contains digits or Unicode digit otherwise it return false.

32.isupper(): it return false, if the string contains at least one uppercase letter otherwise it return true.

33.islower(): it return false if the string contains at least one uppercase letter otherwise it return true.

34.istitle():it returns ture if the string is in title case(every word starting alphabet is uppercase) other wise it returns false.

35.max():it return the maximum charaters(based on ascii values)

36.min(): it returns the minimum charaters(based upon ascii values)

37.ord(): it return Unicode code point

Ex; print(ord(‘A’))

38.startswith():it return true when the element specified word other wise return false.

Ex: s=”hello world”

Print(s.startswith(‘world’))

39.endswith(): it return true when element specified word to return other wise false.

Ex: s=”hello world”

Print(s.endswith(‘word’))

40.join(): it is use to join the element into single string.

Ex:hari=[”Krishna”,”hari”]

Print(‘ ‘.join(hari))

Output: krishnahari

**Examples of strings**

from os.path import split  
  
s = 'hari'  
print(len(s)) # Output: 4  
# center()  
s = 'hari'  
print(s.center(10,'\*'))  
# output :\*\*\*hari\*\*\*  
# ljust()  
s = 'hari'  
print(s.ljust(10, '-'))  
# Output: 'hari------'  
# rjust  
s = 'hari'  
print(s.rjust(10, '-'))  
# Output: '------hari'  
# Zfill()  
s = '42'  
print(s.zfill(5))  
# Output: '00042'  
# uppercase  
s = 'hari'  
print(s.upper())  
# Output: 'HARI'  
# lowercase  
s = 'HARI'  
print(s.lower())  
# Output: 'hari'  
# casefold()  
s = 'HARI'  
print(s.casefold())  
# Output: 'hari'  
# capitalize()  
s = 'hari'  
print(s.capitalize())  
# Output: 'Hari'  
# swapcase()  
s = 'HaRi'  
print(s.swapcase())  
# Output: 'hArI'  
# title()  
s = 'hello world'  
print(s.title())  
# Output: 'Hello World'  
  
# find()  
s = 'hello world'  
print(s.find('o'))  
# Output: 4  
  
# index()  
s = 'hello world'  
print(s.index('o'))  
# Output: 4  
  
# rfind()  
s = 'hello world'  
print(s.rfind('o'))  
# Output: 7  
# rindex()  
s = 'hello world'  
print(s.rindex('o'))  
# Output: 7  
# count()  
s = 'hello world'  
print(s.count('o'))  
# Output: 2  
# lstrip()  
s = ' hello'  
print(s.lstrip())  
# Output: 'hello'  
# rstrip()  
s = 'hello '  
print(s.rstrip())  
# Output: 'hello'  
# strip()  
s = ' hello '  
print(s.strip())  
# Output: 'hello'  
# split()  
s = 'hello world'  
print(s.split())  
# Output: ['hello', 'world']  
# splitline()  
s = 'hello\nworld'  
print(s.splitlines())  
# Output: ['hello', 'world']

**File operations**

File operation play main role when the data needs to be stored permanently into the file. A file is a named location on disk to store related information. We can access the stored information after the program termination.

File operation have different types

**1. Open a file.**

**2. Read Or write**

**3. Closed the file**

**1.Open a file:** it is a open() function that accepts two argument like file name and access mode in which file is access.

**Syntax:** open(file\_name,access\_mode)

Ex:open(“harifile.txt”,”r”)

For suppose we have file named is hari.txt with the following content.

If you want to open the by using open().

File1=open(“hari.txt”,”r”)

There are different types of modes in open()file.

**“r” read:**open file for reading,error file does not exist

**“a” append:** open file appending data from another file, if it does not exist.

**“w” write:** open a file for writing , create file if does not exist.

**“x” Create:** create specific file it return an error it the file exists.

**2.Reading file in python:** we open the file use read() method to read its data.

Ex:file1= open(“hello.txt”,”r”)

Print( file1.read())

Ex:file=open(“hello.text”, “r”)

Print(file.read(5))

Output:

Hello

**Read line:**

You can return only one line by using the reading () method.

By calling readline() you can read first line

By call readline() two times you can read the two first line.

Ex: f=open(“hari.txt”,”r”)

Print (f.readline())

Output:

Hello hari

Ex f=open(“hari.txt”,”r”)

Print (f.readline())

Print(f.readline())

Output:

Hello hari

Nice hari

**Write to an existing file:**

You have existing file you must add to parameters open() function.

Write: will overwrite any existing content.

Append: append content to the end of file.

Ex: f=open(“hari.txt”,”w”)

f.write(“ this is hari”)

f.close()

f=open(“hari.txt”,”r”)

print(f.read())

output: this is hari

ex: f=open(“hari.txt”,”a”)

f=open(“hari.txt”,”r”)

print(f.read())

output:

this is hari

nice ra hari

crate file by denoted by “x”

Ex f=open(“hari.txt”,”x”)

F=open(“hari.txt”,”w”)

Ex: f=open(“afile.txt”,”x”)

F=open(“afile.txt”,”w”)

f.write(“welcome to python”)

f.close()

f=open(“afile . txt”, ”r”)

print(f. read())

**advantages of filehandling**

1.varsatility: it will supports a wide variety of files formats including like text file, binary file, structured files.

2.Flexibility: python provides flexible file mod like read(),Write(a),binary(rb,wb) etc.

It is combine the read,write(W),append(a) use this case.

3.user: in python file has based on user friendly like simple syntax.

4.cross-platform: it will support like windows, linux and macOs

**Disadvantages:**

1.Error-prone: file operation are prone to errors during runtime due to various reasons like file not existing ,no access permissions, input/out put.

2.security Risks: it allowing user to access or modify files they should not.

It arbitrary file execution or reading sensitive file.

It can accepting unvalidated file execution name or paths from user input is particularly dangerous.

3.Complexity:

Managing multiple files and ensuring proper read/write operations can become complex.