Chapter 1: Introduction

**1. What are different features of python?**

1) Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

2) Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type print("Hello World"). It will take only one line to execute, while Java or C takes multiple lines.

3) Interpreted Language

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

5) Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code

8) Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

9) GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

10) Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C,C++ Java. It makes easy to debug the code.

11. Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

12. Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

**2. Differentiate between Java and Python**

|  |  |
| --- | --- |
| **Java** | **Python** |
| It is object-oriented programming language | It is functional as well as object-oriented programming language |
| Java programs are verbose | Python programs are concise and compact |
| Java has do-while statement | Python do not have do-while statement |
| Java has switch statement | Python do not have switch statement |
| Java has 1D and 2D arrays | Python only supports 1D array, to support multidimensional array it uses numPy |
| Indentation of statements is not necessary | Indentation of statements is necessary |
| A semicolon is used to terminate statement | New line indicated end of the statement |

**3. Explain different sequence data types**

1. String (str)

Strings are amongst the most popular types in Python.

We can create them simply by enclosing characters in quotes.

Example: s1=‘Welcome to Python’

In Python, Strings are arrays of bytes

Python does not have a character data type; a single character is simply a string with a length of 1.

Square brackets [ ] can be used to access elements of the string.

2. List

Lists are used to store multiple items in a single variable.

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Since lists are indexed, lists can have items with the duplicate value

Example:

a = ["apple", "banana", "cherry"]

print(a) #['apple', 'banana', 'cherry']

3. Tuple

Tuples are used to store multiple items in a single variable.

When we say that tuples are ordered, it means that the items have a defined order, and that order will not change

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Since tuple are indexed, tuples can have items with the duplicate value

Example:

a = ("apple", "banana", "cherry")

print(a) #('apple', 'banana', 'cherry')

4. Set

Sets are used to store multiple items in a single variable.

Sets items are unordered, which means that the items in a set do not have a defined order.

Sets are unchangeable, meaning that we cannot change the items after the set has been created.

Sets cannot have two items with the same value, so no duplication

Example:

a={"apple", "banana", "cherry", "apple"}

print(a) #{'banana', 'cherry', 'apple'}

5. Dictionary

Dictionaries are used to store data values in key:value pairs.

When we say that dictionaries are unordered, it means that the items does not have a defined order, you cannot refer to an item by using an index.

Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

Dictionary cannot have two items with the same value, so no duplication

Example:

a={"brand": "Ford", "model": "Mustang", "year": 1964}

print(a) #{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **List** | **Tuple** | **Set** | **Dictionary** |
| **Ordered** | √ | √ | -- | -- |
| **Changeable** | √ | -- | -- | √ |
| **Duplicates** | √ | √ | -- | -- |
| **Brackets** | [ ] | ( ) | { } | { } |
| **Example** | A=[10,20,30] | A=(10,20.5,30) | A={‘SSS’,20,30} | A={1:’AA’, ‘bb’:3} |

**4. Explain different string handling functions**

|  |  |
| --- | --- |
| Methods | Description |
| s1.isalnum() | Returns True if all characters in the string are alphanumeric |
| s1.isalpha() | Returns True if all characters in the string are in the alphabet |
| s1.isdigit() | Returns True if all characters in the string are digits |
| s1.islower() | Returns True if all characters in the string are lower case |
| s1.isupper() | Returns True if all characters in the string are upper case |
| s1.isnumeric() | Returns True if all characters in the string are numeric |
| s1.isspace() | Returns True if all characters in the string are whitespaces |
| s1.upper() | Converts a string into upper case |
| s1.lower() | Converts a string into lower case |
| s1.capitalize() | Converts the first character to upper case |
| s1.replace() | Returns a string where a specified value is replaced with a specified value |
| s1.split() | Splits the string at the specified separator, and returns a list |
| s1.swapcase() | Swaps cases, lower case becomes upper case and vice versa |
| s1.title() | Converts the first character of each word to upper case |
| s1.replace() | Returns a string where a specified value is replaced with a specified value |
| len(s1) | Returns length of a string |
| s1.find() | Searches the string for a specified value and returns the position of where it was found |
| s1.index() | Searches the string for a specified value and returns the position of where it was found |
| s1.strip() | Returns a trimmed version of the string |
| s1.lstrip() | Returns a left trim version of the string |
| s1.rstrip() | Returns a right trim version of the string |

Example:

>>> s='a'

>>> type(s)

<class 'str'>

>>> s='Welcome'

>>> type(s)

<class 'str'>

>>> print(s)

Welcome

>>> s[0]

'W'

>>> s[3:6]

'com'

>>> s[0:4]

'Welc'

>>> s[3:]

'come'

>>> s[-1]

'e'

>>> s

'Welcome'

>>> s[1:6:2]

'ecm'

>>> s[::-1]

'emocleW'

>>> s='Welcome to VCP'

>>> print(s)

Welcome to VCP

>>> len(s)

14

>>> s.isalnum()

False

>>> s.isalpha()

False

>>> s='1234'

>>> s.isdigit()

True

>>> s.isnumeric()

True

>>> s='welcome'

>>> s

'welcome'

>>> s.islower()

True

>>> s.isupper()

False

>>> s.isspace()

False

>>> s=' '

>>> s.isspace()

True

>>> s='welcome to python'

>>> s

'welcome to python'

>>> s1=s.upper()

>>> s1

'WELCOME TO PYTHON'

>>> s

'welcome to python'

>>> s1='WELcome'

>>> s1

'WELcome'

>>> s2=s1.lower()

>>> s2

'welcome'

>>> s1='welcome to vcp'

>>> s1.capitalize()

'Welcome to vcp'

>>> s1

'welcome to vcp'

>>> s2=s1.title()

>>> s2

'Welcome To Vcp'

>>> s2

'Welcome To Vcp'

>>> s3=s2.swapcase()

>>> s3

'wELCOME tO vCP'

>>> a='I like apple'

>>> a

'I like apple'

>>> b=a.replace('apple','cherry')

>>> b

'I like cherry'

>>> a

'I like apple'

>>> b=a.split()

>>> b

['I', 'like', 'apple']

>>> s='welcome'

>>> s[6:1:-2]

'eol'

>>> s='Welcome to python'

>>> s

'Welcome to python'

>>> s.find('to')

8

>>> s.index('python')

11

>>> s.index('thon')

13

>>> s.index('t')

8

>>> s=' SSS '

>>> s

' SSS '

>>> s.strip()

'SSS'

>>> s=' sss sss '

>>> s

' sss sss '

>>> s.strip()

'sss sss'

>>> a=' sss '

>>> a.lstrip()

'sss '

>>> a.rstrip()

' sss'

**5. Explain functions in python with example**

A function is a block of code which only runs when it is called

We can pass data known as parameters or arguments to a function

A function can return data as result

Example:

def f1( ):

print(‘Hello’)

f1()

Every parameter in python is passed by reference

If we do not know how many parameters passed to a function, add \* before parameter name in function call, it is called as Variable Arguments

Python uses recursion, when a function calls itself it is called as recursion

Example:

def max(x,y):

if x>y:

return x

else:

return y

a,b,c,d=map(int,input('Enter 4 numbers').split())

e=max(a,b) #e=max(max(a,b),max(c,d))

e=max(e,c)

e=max(e,d)

print('Largest Number=',e)

**6. Explain arrays in python with example**

import array as arr

# creating an array with integer type

a = arr.array('i', [1, 2, 3])

# printing original array

print ("The new created array is : ", end =" ")

for i in range (0, 3):

print (a[i], end =" ")

print()

# creating an array with float type

b = arr.array('d', [2.5, 3.2, 3.3])

# printing original array

print ("The new created array is : ", end =" ")

for i in range (0, 3):

print (b[i], end =" ")

Example:

import array

a=array.array('i',[])

n=int(input('Enter size of array='))

for i in range(0,n):

ele=int(input('Enter number='))

a.append(ele)

sum=0

for i in range(n):

sum=sum+a[i]

print('Sum of array elements=',sum)

import array

a=array.array('i',[])

n=int(input('Enter size of array='))

for i in range(0,n):

ele=int(input('Enter number='))

a.append(ele)

i=0

j=len(a)-1

while i<j:

temp=a[i]

a[i]=a[j]

a[j]=temp

i=i+1

j=j-1

for i in a:

print(i)

**7. Explain different math handling functions**

|  |  |
| --- | --- |
| Methods | Description |
| math.floor(n) | Rounds a number down to the nearest integer |
| math.ceil(n) | Rounds a number up to the nearest integer |
| math.degrees(n) | Converts an angle from radians to degrees |
| math.radians(n) | Converts a degree value into radians |
| math.fabs(n) | Returns the absolute value of a number |
| math.factorial(n) | Returns the factorial of a number |
| math.gcd(m,n) | Returns the greatest common divisor of two integers |
| math.log(n) | Returns the logarithmic value of n |
| math.log10(n) | Returns the base-10 logarithm of n |
| math.log2(n) | Returns the base-2 logarithm of n |
| math.sqrt(n) | Returns the square root of a number |
| math.pow(x,y) | Returns the value of x to the power of y |
| math.trunc(n) | Returns the truncated integer parts of a number |
| math.sin(n) | Returns the sine of a number |
| math.sinh(n) | Returns the hyperbolic sine of a number |
| math.asin(n) | Returns the arc sine of a number |

Example:

>>> import math

>>> math.ceil(5.8)

6

>>> math.ceil(-2.5)

-2

>>> math.floor(6.1)

6

>>> math.floor(-1.8)

-2

>>> math.trunc(6.7)

6

>>> math.degrees(8.90)

509.9324376664327

>>> math.radians(509.9324376664327)

8.9

>>> math.fabs(-9)

9.0

>>> math.factorial(4)

24

>>> math.gcd(10,20)

10

>>> math.sqrt(2)

1.4142135623730951

>>> math.log(10)

2.302585092994046

>>> math.log10(20)

1.3010299956639813

>>> math.log2(30)

4.906890595608519

>>> math.sin(30)

-0.9880316240928618

>>> math.asin(-0.9880316240928618)

-1.4159265358979325

>>> math.sinh(10)

11013.232874703393

>>>