Python

September 23, 2024

1 PYTHON BASICS AND INTERVIEW QUESTIONS with Python DSA Questions

2 BY PRATHAM M

Print Function

[1]:
```
print("Hi I am pratham")
print("welcome to Learning2Infinity" )
```

```
Hi I am pratham
welcome to Learning2Infinity
```

#Variables in Python1

[2]:
```
name="Pratham"
print(name)

a="youtube channel"
print(a)
```

```
Pratham
youtube channel
```

Once a varible is defined. the data inside it can be accesseed when ever we want just by using varibale name

[3]:
```
print(a)
```

```
youtube channel
```

Accessing Varibales in any of statement

[4]:
```
print(name+'is a youtuber')
print(name+'is teaching python')
print(name+'is a proud Kannadiga')
```

```
Prathamis a youtuber
Prathamis teaching python
Prathamis a proud Kannadiga
```

##################Strings in Python

[5]:
```
print('Hello. I am Pratham')
print('Hello.\nI am Pratham') #print in next line use \n
```

```
Hello. I am Pratham
Hello.
I am Pratham
```

#Indexing In python for string

Indexing is the process of accessing individual elements in a sequence data type, such as strings, lists, tuples, and arrays. Indexing in Python is zero-based, meaning the first element has an index of 0, the second has an index of 1, and so on

[6]:
```
print(a[0])
print(a[2])
print(a[10])
```

```
y
u
a
```

TO change to Uppercase and Lowercase

[7]:
```
print(name)
b=name.isupper()
print(b)
c=name.upper()
print(c)


d=name.islower()
print(d)
e=name.lower()
print(e)
```

```
Pratham
False
PRATHAM
False
pratham
```

to find the length

[8]:
```
name
len(name)
```

[8]:
```
7
```

How to get index number

```
[9]: print(name)
     n1=name.index('P')
     print(n1)

     n2=name.index('a')
     print(n2)
     n3=name.index('m')
     print(n3)
```

```
Pratham
0
2
6
```

REPLACING THE LETTER

```
[10]: print(name)
      print(name.replace('m','M'))
```

```
Pratham
PrathaM
```

Numbers in Python

```
[11]: print(30)
      number=16
      print(number)


      #Basic Airthmetic operations
      a=30
      b=16
      print('Airthemetic Operations')
      print(a+b)
      print(a-b)
      print(a*b)
      print(a/b)
      print(a%b)
      print(a**b)
```

```
30
16
Airthemetic Operations
46
14
480
1.875
14
43046721000000000000000
```

How to Concatenate

```
[12]: num1=30
num2=str(num1)
#print('number is'+num1)   return error
print('The number is '+num2)
```

```
The number is 30
```

absolute abs function

```
[13]: print(-5)
print(abs(-5))
```

```
-5
5
```

Max Function

```
[14]: a=30
b=16
c=27
d=max(a,b,c)
print('the max number is',d)
```

```
the max number is 30
```

Round Function

```
[15]: print(round(3.5))
print(round(2.333333))
a1=2334.6667
print(round(a1))
```

```
4
2
2335
```

bin Function- convert into binary

```
[16]: a=30
b=16
print(bin(a))
print(bin(b))
```

```
0b11110
0b10000
```

```
[17]: from math import *
print(sqrt(100))
print(sqrt(144))
```

```
10.0
12.0
```

Get user input

```
[18]: name1=input('Input ur name:')
      age=input('Input the age:')
      print(name1)
      print(int(age))
      print("The name is "+name1+" The age is"+age)
```

```
pratham
21
The name is pratham The age is21
```

Replace Function

```
[19]: string=input('Enter the Sentence:')
      print('You sentence is'+string)
      word1=input('enter to replace')
      word2=input('enter the word to replace it with')
      print(string.replace(word1,word2))
```

```
You sentence isHi how are ur day ?
Hi how is ur day ?
```

List In Python

```
[42]: states=['Karnataka','TN','Andra','Telangana','kerala']
      print(states)
      print(states[0])
      print(states[0][2])
      print(states[0:])
      print(states[1:])
      print(states[2:])
      print(states[1:3])
      print(states[:-2])
      print(states[-1:])
      print(states[-2:-3])
      print(states[-3:])
      print(states[-4:-1])

      print(type(states))
      print(len(states))

      state1=['karntaka',30,True,'Andra']
      print(state1)
      print(type(state1[0]))
      print(type(state1[1]))
      print(type(state1[2]))
print(type(state1[3]))
```

```
['Karnataka', 'TN', 'Andra', 'Telangana', 'kerala']
Karnataka
r
['Karnataka', 'TN', 'Andra', 'Telangana', 'kerala']
['TN', 'Andra', 'Telangana', 'kerala']
['Andra', 'Telangana', 'kerala']
['TN', 'Andra']
['Karnataka', 'TN', 'Andra']
['kerala']
[]
['Andra', 'Telangana', 'kerala']
['TN', 'Andra', 'Telangana']
<class 'list'>
5
['karntaka', 30, True, 'Andra']
<class 'str'>
<class 'int'>
<class 'bool'>
<class 'str'>
```

Change the word in list

[31]:
```
print(states)
states[1]='TamilNadu'
print(states)
```

```
['Karnataka', 'TN', 'Andra', 'Telangana', 'kerala']
['Karnataka', 'TamilNadu', 'Andra', 'Telangana', 'kerala']
```

[ ]:
```
#COMPLETED VIDEO TILL 1hr
```

[1]:
```
def print_square_pattern(n):
    for i in range(n):
        print('* ' * n)

def print_triangle_pattern(n):
    for i in range(1, n + 1):
        print('* ' * i)

def print_reverse_triangle_pattern(n):
    for i in range(n, 0, -1):
        print('* ' * i)

def print_right_angle_triangle_pattern(n):
    for i in range(1, n + 1):
        print(' ' * (n - i) + '* ' * i)
```

```python
def print_pyramid_pattern(n):
    for i in range(1, n + 1):
        print(' ' * (n - i) + '* ' * (2 * i - 1))

def print_reverse_pyramid_pattern(n):
    for i in range(n, 0, -1):
        print(' ' * (n - i) + '* ' * (2 * i - 1))

# Example usage:
print("Square pattern:")
print_square_pattern(5)
print("\nTriangle pattern:")
print_triangle_pattern(5)
print("\nReverse Triangle pattern:")
print_reverse_triangle_pattern(5)
print("\nRight Angle Triangle pattern:")
print_right_angle_triangle_pattern(5)
print("\nPyramid pattern:")
print_pyramid_pattern(5)
print("\nReverse Pyramid pattern:")
print_reverse_pyramid_pattern(5)
```

```
Square pattern:
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

Triangle pattern:
*
* *
* * *
* * * *
* * * * *

Reverse Triangle pattern:
* * * * *
* * * *
* * *
* *
*

Right Angle Triangle pattern:
    *
   * *
  * * *
 * * * *
* * * * *

Pyramid pattern:
    *
   * * *
  * * * * *
 * * * * * * *
* * * * * * * * *

Reverse Pyramid pattern:
* * * * * * * * *
 * * * * * * *
  * * * * *
   * * *
    *
```

[1]:

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (2.11.0)
Requirement already satisfied: keras in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (2.11.0)
Requirement already satisfied: numpy in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (1.21.6)
Collecting matplotlib
  Downloading matplotlib-3.5.3-cp37-cp37m-win_amd64.whl.metadata (6.7 kB)
Requirement already satisfied: tensorflow-intel==2.11.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow) (2.11.0)
Requirement already satisfied: absl-py>=1.0.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (3.8.0)
```

```
Requirement already satisfied: libclang>=13.0.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (16.0.6)
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (3.3.0)
Requirement already satisfied: packaging in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (23.2)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (3.19.6)
Requirement already satisfied: setuptools in c:\program files\python37\lib\site-
packages (from tensorflow-intel==2.11.0->tensorflow) (47.1.0)
Requirement already satisfied: six>=1.12.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (4.7.1)
Requirement already satisfied: wrapt>=1.11.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (1.62.0)
Requirement already satisfied: tensorboard<2.12,>=2.11 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (2.11.2)
Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (2.11.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorflow-intel==2.11.0->tensorflow) (0.31.0)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.11.0-py3-none-any.whl.metadata (785 bytes)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.38.0-py3-none-any.whl.metadata (138 kB)
     ---------------------------------------- 138.5/138.5 kB 2.8 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1 (from matplotlib)
  Downloading kiwisolver-1.4.5-cp37-cp37m-win_amd64.whl.metadata (6.5 kB)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
matplotlib) (9.5.0)
```

```
Collecting pyparsing>=2.2.1 (from matplotlib)
  Downloading pyparsing-3.1.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
astunparse>=1.6.0->tensorflow-intel==2.11.0->tensorflow) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (2.28.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (3.4.4)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (2.2.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
(5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
(0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
(4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
```

```
(6.7.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
werkzeug>=1.0.1->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
(2.1.5)
Requirement already satisfied: zipp>=0.5 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (3.15.0)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from
pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
(0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in
c:\users\pratham.m\appdata\roaming\python\python37\site-packages (from requests-
oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow) (3.2.2)
Downloading matplotlib-3.5.3-cp37-cp37m-win_amd64.whl (7.2 MB)
   ---------------------------------------- 7.2/7.2 MB 5.5 MB/s eta 0:00:00
Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Downloading fonttools-4.38.0-py3-none-any.whl (965 kB)
   ---------------------------------------- 965.4/965.4 kB 4.7 MB/s eta 0:00:00
Downloading kiwisolver-1.4.5-cp37-cp37m-win_amd64.whl (55 kB)
   ---------------------------------------- 55.8/55.8 kB 2.8 MB/s eta 0:00:00
Downloading pyparsing-3.1.2-py3-none-any.whl (103 kB)
   ---------------------------------------- 103.2/103.2 kB 2.0 MB/s eta 0:00:00
Installing collected packages: pyparsing, kiwisolver, fonttools, cycler,
matplotlib
Successfully installed cycler-0.11.0 fonttools-4.38.0 kiwisolver-1.4.5
matplotlib-3.5.3 pyparsing-3.1.2
```

```
Note: you may need to restart the kernel to use updated packages.
```

3 Operator in Python

3.1 1.Airthmetic Operator

[4]:
```
a=50
b=45

print("adds",a+b)
print("subs",a-b)
print("Mult",a*b)
print("Div",a/b)
print("Floor div",a//b)
print("Mod for getting remainder",a%b)
print(a%3)
```

```
95
5
2250
1.1111111111111112
1
5
2
```

3.2 2.Assignment operators

[14]:
```
x=5



x+=5
print(x)

x-=5
print(x)

x=+5
print(x)

x=-5
print(x)
```

```
10
5
5
-5
```

3.3 Logical Operator

[19]:
```
x1=10
y1= 20

print(x1==10 and x1<y1 and x1==y1)
```

```
False
```

3.4 Membership Operator

[21]:
```
str1="Hello"
print('h' in str1 )
print('H' in str1 )
```

```
False
True
```

[24]:
```
l1=[10,20,30,40]
print(50 in l1)

print(50 not in l1)
```

```
False
True
```

3.5 Identity operators

[26]:
```
z1=10
z2=10
print(z1 is z2,z1==z2)
print(z1 is not z2,z1!=z2)
```

```
True True
False False
```

3.6 Bitwise Operator

&=AND
|=OR
^=XOR

[27]:
```
XX=10
YY=9
print(bin(XX))
```

```
0b1010
```

[28]:
```
print(bin(YY))
```

```
0b1001
```

[35]:
```
print("AND OPERATION",XX &YY,bin(XX&YY))

print("OR OPERATION",XX &YY,bin(XX|YY))

print("XOR OPERATION",XX^YY,bin(XX^YY))
```

```
AND OPERATION 8 0b1000
OR OPERATION 8 0b1011
XOR OPERATION 3 0b11
```

4 Data Types In Python

Mutable objects can chnage its state or contents and Immutable objects cannot

Mutable Data types: List,Dictionary,Byte array

Immutable Dta Types: Int,Float,Complex,String,Tuple,Set

5 For Loop with range

range(5) means: it starts from 0, condition< 5 and increment 1

range(1,6) means: it starts from 1, condition less than 6, increment 1

range(1,6,2) means: it starts from 1, condition less than 6, increment 2, o/p:1,3,5

[41]:
```
for n in range(5):
    print(n)
```

```
0
1
2
3
4
```

[42]:
```
for n in range(1,5):
    print(n)
```

```
1
2
3
4
```

[43]:
```
for n in range(1,6,2):
    print(n)
```

```
1
3
5
```

```
[44]: for a in range(1,11):
          print("2*",a,"=",2*a)
```

```
2* 1 = 2
2* 2 = 4
2* 3 = 6
2* 4 = 8
2* 5 = 10
2* 6 = 12
2* 7 = 14
2* 8 = 16
2* 9 = 18
2* 10 = 20
```

5.0.1 Range in reverse

```
[47]: for b in range(10,0,-1):
          print(b)
```

```
10
9
8
7
6
5
4
3
2
1
```

```
[48]:
```

```
2*b 2
```

6 While Statement

```
[52]: i=1
      while i<=10:
          print(i,"Hi")
          i=i+1
```

```
1 Hi
2 Hi
3 Hi
4 Hi
5 Hi
6 Hi
7 Hi
8 Hi
```

```
9 Hi
10 Hi
```

[53]:
```
a="pratham Hi HOw aRe You"
print(a.lower())
print(a.upper())
print(a.title())
print(a.capitalize())
```

```
pratham hi how are you
PRATHAM HI HOW ARE YOU
Pratham Hi How Are You
Pratham hi how are you
```

use this as input “pratham Hi HOw aRe You” and output should be “Pratham Hi How ARE yOU”

[59]:
```
n = "pratham Hi HOw aRe You"
# Split the string into a list of words
words = n.split()
print(words)

# Modify the required words as per your example
words[0] = words[0].capitalize()  # 'Pratham'
words[2] = words[2].capitalize()  # 'How'
words[3] = words[3].upper()       # 'ARE'
words[4] = words[4].swapcase()    # 'yOU'

# Join the words back into a sentence
result = " ".join(words)
print(result)
```

```
['pratham', 'Hi', 'HOw', 'aRe', 'You']
Pratham Hi How ARE yOU
```

7 Chr() and Ord() Functions

[62]:
```
a=65
print(chr(a))

b=69
print(chr(b))
```

```
A
E
```

[66]:
```
m='P'
print(ord(m))

m='r'
```

```
print(ord(m))
```

```
80
114
```

ord() function:

`Purpose: Converts a single character to its corresponding ASCII or Unicode integer value.`

chr() function:

`Purpose: Converts an ASCII or Unicode integer value to its corresponding character.`

8 LIST

[70]:
```
L=[10,20,30,40,50,"hello"]
print(L[3],L[4])
print(L[0:2])
print(L[0::2])
print(L[3:])
print(L[-1::])
print(L[-1::-2])
```

```
40 50
[10, 20]
[10, 30, 50]
[40, 50, 'hello']
['hello']
['hello', 40, 20]
```

8.1 List Function

del pop() remove() clear() insert() append() extend()

[12]:
```
l1=[20,30,40,50]
del l1[1]
print(l1)
```

```
[20, 40, 50]
```

[13]:
```
l1.pop(2)
print(l1)
```

```
[20, 40]
```

[15]:
```
print(l1.pop(1))
print(l1)
```

```
40
[20]
```

[25]:
```
l2=[20,2,3,4,45,6,7]
l2.remove(45)
print(l2)

l2[0]=100
print(l2)
```

```
[20, 2, 3, 4, 6, 7]
[100, 2, 3, 4, 6, 7]
```

[23]:
```
l2.clear()
print(l2)
```

```
[]
```

[32]:
```
l3=[2,3,4,5,6,7,8,9]
l3.insert(1,33)
print(l3)

l3.append(l2)  #append() adds an element at the end of the list
print(l3)

l3.extend(l2)
print(l3)
```

```
[2, 33, 3, 4, 5, 6, 7, 8, 9]
[2, 33, 3, 4, 5, 6, 7, 8, 9, [100, 2, 3, 4, 6, 7]]
[2, 33, 3, 4, 5, 6, 7, 8, 9, [100, 2, 3, 4, 6, 7], 100, 2, 3, 4, 6, 7]
```

8.2 List Comprehension

[41]:
```
l=[]
for a in range(1,101):
    l.append(a)
print(l)

#Using comprehension
n=[m for m in range(1,101)]
print(n)

#printing only even number from 1 to 101
n=[m for m in range(1,101) if m%2==0]
print(n)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82,
84, 86, 88, 90, 92, 94, 96, 98, 100]
```

[42]:
```
s="Pratham"
d=[g for g in s]
print(d)
```

```
['P', 'r', 'a', 't', 'h', 'a', 'm']
```

[55]:
```
m=[10,20,30,40,50]
n=[2,3,4,5,6]
t=len(n)
for a,b in zip(m,n):
    print(a,b)

for h in range(t):
    print(m[h],n[h])
```

```
10 2
20 3
30 4
40 5
50 6
10 2
20 3
30 4
40 5
50 6
```

9 Program to convert string to list

[57]:
```
a=input()
b=a.split()
c=list(b)
print(c)
```

```
['Hi', 'hello', 'namskaes']
```

[60]:
```
#another method of converting sting to list
l=[]
for a in range(1,4):
    n=input("enter the value"+str(a))
    l.append(n)
print(l)
```

```
['hi', 'helllo', 'pratham']
```

10 Stack in Python

The stack is linear data structure

stores item in a Last-In First Out(LIFO) or First-In-Last-Out(FILO) manner

[3]:
```
st=[]
while True:
    c=int(input('''
            1 Push Elements
            2 Pop Elements
            3 Peek Element
            4 Display Stack
            5 Exit
            '''))
    if c==1:
        n=input("Enter the value");
        st.append(n)
        print(st)
    elif c==2:
        if len(st)==0:
            print("Stack is empty")
        else:
            p=st.pop()
            print(p)
            print(st)
    elif c==3:
        if len(st)==0:
            print("Stack is empty")
        else:
            print("Last value of stack",st[-1])
    elif c==4:
        print("Display stack",st)
    elif c==5:
        break;
```

```
Stack is empty
['200']
['200', '300']
Last value of stack 300
Last value of stack 300
Display stack ['200', '300']
```

11 Queue in Python

The queue is a linear Data Structure Stores item in First In First Out Manner

Queue Operations: Enqueue,Dequeue,Front,Rear

[5]:
```
st = []

while True:
    c = int(input('''
            1 Push Elements
            2 Pop Elements
            3 Front Element
            4 Last Element
            5 Display Queue
            6 Exit
            '''))

    if c == 1:
        n = input("Enter the value: ")
        st.append(n)
        print("Queue:", st)

    elif c == 2:
        if len(st) == 0:
            print("Queue is empty")
        else:
            popped_element = st.pop(0)  # Removing the front element
            print(f"Popped element: {popped_element}")
            print("Updated Queue:", st)

    elif c == 3:
        if len(st) == 0:
            print("Queue is empty")
        else:
            print("Front element of the queue:", st[0])

    elif c == 4:
        if len(st) == 0:
            print("Queue is empty")
        else:
            print("Last element of the queue:", st[-1])

    elif c == 5:
        print("Current Queue:", st)
    elif c == 6:
        print("Exiting...")
        break

    else:
        print("Invalid operation, please select a valid option.")
```

```
Queue: ['30']
Queue: ['30', '40']
Queue: ['30', '40', '50']
Popped element: 30
Updated Queue: ['40', '50']
Front element of the queue: 40
Last element of the queue: 50
Current Queue: ['40', '50']
Exiting…
```

[6]:
```
# completed till 6 hr 45mins
```

[5]:
```
nums=[1,2,3,4]
for i in range(len(nums)):
    for j in range(i,len(nums)):
        print(nums[j],end="")
```

```
1
2
3
4
2
3
4
3
4
4
```

12 Interview Questions

1.Find out common letters between two strings using python

[1]:
```
a=input("enter a string")
b=input("enter second string")
print("strings are",a,b)
c=set(a)
d=set(b)
lst=c & d
print(lst)
```

```
strings are pratham ranjith
{'t', 'r', 'a', 'h'}
```

2.Using SQLite3 with Python

Description: Learn how to create and manage SQLite databases using the sqlite3 module.

[5]:
```
import sqlite3

conn=sqlite3.connect('example2.db')
c=conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS users (name text , age integer, city␣
 ↪text)''')

c.execute("INSERT INTO users VALUES('Pratham',22,'Bengaluru')")
conn.commit()

c.execute("SELECT *FROM users")
print(c.fetchall())

conn.close()
```

```
[('Pratham', 22, 'Bengaluru')]
```

3.Count the frequency of words apperaring in python

[1]:
```
def freq():
    str=input("enter a string")
    str=str.lower()
    # str=str.translate(str.maketrans('','','.,!?:;'))
    li=str.split()
    d={}

    for i in li:
        if i not in d.keys():
            d[i]=0
        d[i]=d[i]+1
    print(d)
freq()
```

```
{'pratham': 2, 'name': 1, 'is': 2, 'he': 2, 'likes': 2, 'to': 1, 'code': 2,
'douring': 1, 'coding': 1, 'well': 1}
```

4.Find the missing number in array

[14]:
```
def get_missing_summation(a):
    # The last number in the array should be the highest number
    n = len(a) + 1  # Adjusting n to be the size of the full sequence
    total = n * (n + 1) // 2  # Sum of first n natural numbers
    sum1 = sum(a)  # Sum of the array
    print(f"Missing number (Summation method): {total - sum1}")

def get_missing_xor(a):
    n = len(a) + 1  # Adjusting n to be the size of the full sequence
    xor_a = a[0]
    for index in range(1, len(a)):
        xor_a ^= a[index]

    x2 = 0
    for index in range(1, n + 1):
        x2 ^= index

    print(f"Missing number (XOR method): {xor_a ^ x2}")

# Example array with a missing number
arr = [1, 2, 3, 5]  # Missing number is 4

get_missing_summation(arr)
get_missing_xor(arr)
```

```
Missing number (Summation method): 4
Missing number (XOR method): 4
```

[4]:
```
def get_missing_summation(a):
    # The last number in the array should be the highest number
    n = len(a) + 1  # Adjusting n to be the size of the full sequence
    total = n * (n + 1) // 2  # Sum of first n natural numbers
    sum1 = sum(a)  # Sum of the array
    print(f"Missing number (Summation method): {total - sum1}")

# def get_missing_xor(a):
#     n = len(a) + 1  # Adjusting n to be the size of the full sequence
#     xor_a = a[0]
#     for index in range(1, len(a)):
#         xor_a ^= a[index]

#     x2 = 0
#     for index in range(1, n + 1):
#         x2 ^= index

#     print(f"Missing number (XOR method): {xor_a ^ x2}")

# Example array with a missing number
arr = [1, 2, 3, 5]  # Missing number is 4

get_missing_summation(arr)
```

```
# get_missing_xor(arr)
```

```
Missing number (Summation method): 4
```

5.how to take an input and print in list format

[15]:
```
# Taking input from the user
user_input = input("Enter elements separated by spaces: ")

# Splitting the input string into a list
elements_list = user_input.split()

# Printing the list
print("List:", elements_list)
```

```
List: ['HI', 'HELLO', 'NAMSKARA']
```

6.Find Out Pairs with given sum in an array in python of time complexity O(n log n)- FACEBOOK,AMAZON

[17]:
```
def twosum(arr,sum):
    arr.sort()
    left=0
    right=len(arr)-1
    while left<=right:
        if arr[left]+arr[right]>sum:
            right=right-1
        elif arr[left]+arr[right]<sum:
            left=left+1
        elif arr[left]+arr[right]==sum:
            print("vales of pair are",arr[left],"&",arr[right])
            right=right-1
            left=left+1
arr=[5,7,4,3,9,8,19,21]
sum=16
twosum(arr,sum)
```

```
vales of pair are 7 & 9
vales of pair are 8 & 8
```

7.Height of binary tree(Max Depth)

[6]:
```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def max_depth(node):
    # Base case: If the node is null, the depth is 0
    if node is None:
        return 0
    else:
        # Recursively find the depth of the left and right subtrees
        left_depth = max_depth(node.left)
        right_depth = max_depth(node.right)

        # The height of the current node is the greater of the two subtrees'␣
↪heights plus 1
        return max(left_depth, right_depth) + 1

# Example usage:

# Create a sample binary tree:
#       1
#      / \
#     2   3
#    / \
#   4   5
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

# Find the maximum depth
height = max_depth(root)
print(f"Height of the binary tree: {height}")
```

```
Height of the binary tree: 3
```

Fibonacci Number

[8]:
```
def fibonacci_series(n: int) -> list:
    series = []
    a, b = 0, 1
    for _ in range(n):
        series.append(a)
        a, b = b, a + b
    return series

# Take input from the user
try:
    n = int(input("Enter the number of terms in the Fibonacci series: "))
    if n <= 0:
        print("Please enter a positive integer.")
    else:
        print(f"The Fibonacci series up to {n} terms is: {fibonacci_series(n)}")
```

```
except ValueError:
    print("Invalid input! Please enter an integer.")
```

```
The Fibonacci series up to 10 terms is: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

[11]:
```
def is_palindrome(s: str) -> bool:
    # Normalize the string: remove spaces and convert to lowercase
    s = s.replace(" ", "").lower()
    return s == s[::-1]

# Take input from the user
input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print(f'"{input_string}" is a palindrome.')
else:
    print(f'"{input_string}" is not a palindrome.')
```

```
"A man a plan a canal Panama" is a palindrome.
```

8.Convert a list of dictionary to JSON

[18]:
```
import json

# Example data as a list of dictionaries
data = [
    {"name": "Pratham", "age": 25, "city": "New York"},
    {"name": "John", "age": 30, "city": "San Francisco"},
    {"name": "Jane", "age": 28, "city": "Los Angeles"}
]

# Convert the list of dictionaries to a JSON string
json_data = json.dumps(data, indent=4)

# Print the JSON string
print(json_data)
```

```
[
    {
        "name": "Pratham",
        "age": 25,
        "city": "New York"
    },
    {
        "name": "John",
        "age": 30,
        "city": "San Francisco"
    },
    {
        "name": "Jane",
        "age": 28,
        "city": "Los Angeles"
    }
]
```

9.Longest Substring Without Repeating Characters

[26]:
```python
def longest_unique_substring(s: str) -> int:
    char_map = {}
    left = max_length = 0

    for right in range(len(s)):
        if s[right] in char_map:
            left = max(left, char_map[s[right]] + 1)
        char_map[s[right]] = right
        max_length = max(max_length, right - left + 1)

    return max_length

# Example usage:
input_string = "abcabcbb"
result = longest_unique_substring(input_string)
print(f"The length of the longest substring without repeating characters is:␣
 ↪{result}")
```

```
The length of the longest substring without repeating characters is: 3
```

10.Merged List

[28]:
```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def merge_two_lists(l1: ListNode, l2: ListNode) -> ListNode:
    dummy = ListNode()
    tail = dummy

    while l1 and l2:
        if l1.val < l2.val:
            tail.next = l1
            l1 = l1.next
        else:
            tail.next = l2
            l2 = l2.next
        tail = tail.next

    tail.next = l1 or l2
    return dummy.next
```

```python
def print_list(node: ListNode):
    while node:
        print(node.val, end=" -> ")
        node = node.next
    print("None")

# Example usage:
# Creating two sorted linked lists: 1 -> 2 -> 4 and 1 -> 3 -> 4
l1 = ListNode(1, ListNode(2, ListNode(4)))
l2 = ListNode(1, ListNode(3, ListNode(4)))

merged_list = merge_two_lists(l1, l2)
print("Merged List:")
print_list(merged_list)
```

```
Merged List:
1 -> 1 -> 2 -> 3 -> 4 -> 4 -> None
```

11.Reverse a list

[30]:
```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def reverse_list(head: ListNode) -> ListNode:
    prev = None
    current = head

    while current:
        next_node = current.next
        current.next = prev
        prev = current
        current = next_node

    return prev

def print_list(node: ListNode):
    while node:
        print(node.val, end=" -> ")
        node = node.next
    print("None")

# Example usage:
# Creating a linked list: 1 -> 2 -> 3 -> 4 -> 5
head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))
```

```
print("Original List:")
print_list(head)

# Reversing the linked list
reversed_head = reverse_list(head)
print("Reversed List:")
print_list(reversed_head)
```

```
Original List:
1 -> 2 -> 3 -> 4 -> 5 -> None
Reversed List:
5 -> 4 -> 3 -> 2 -> 1 -> None
```

12.Find the index of sum of two number is 9

[32]:
```
def two_sum(nums: list[int], target: int) -> list[int]:
    num_map = {}
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_map:
            return [num_map[complement], i]
        num_map[num] = i
    return []  # Return an empty list if no solution is found

# Example usage:
nums = [2, 7, 11, 15]
target = 9

result = two_sum(nums, target)
print(f"The indices of the two numbers that add up to {target} are: {result}")
```

```
The indices of the two numbers that add up to 9 are: [0, 1]
```

13.Rotate Function

[33]:
```
def rotate(nums: list[int], k: int) -> None:
    k %= len(nums)
    nums[:] = nums[-k:] + nums[:-k]

# Example usage:
nums = [1, 2, 3, 4, 5, 6, 7]
k = 3
rotate(nums, k)
print(f"The rotated list is: {nums}")
```

```
The rotated list is: [5, 6, 7, 1, 2, 3, 4]
```

14. Binary Search

[34]:
```
def binary_search(arr: list[int], target: int) -> int:
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = left + (right - left) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1

# Example usage:
arr = [1, 2, 3, 4, 5, 6, 7]
target = 5
index = binary_search(arr, target)
print(f"The index of {target} in the array is: {index}")
```

```
The index of 5 in the array is: 4
```

15 Problem: Climbing Stairs (Dynamic Programming)

Description: You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

[46]:
```
def climb_stairs(n):
    if n <= 2:
        return n
    dp = [0] * (n + 1)
    dp[1], dp[2] = 1, 2
    for i in range(3, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]

# Call the function and print the result
n = 5  # Example input
print(climb_stairs(n))  # This will display the output
```

```
8
```

16. Find the Maximum Subarray (Kadane's Algorithm)

[47]:
```
def max_subarray(nums):
    max_current = max_global = nums[0]

    for num in nums[1:]:
        max_current = max(num, max_current + num)
        if max_current > max_global:
            max_global = max_current
    return max_global

# Sample input
nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
result = max_subarray(nums)
print("Maximum subarray sum:", result)
```

```
Maximum subarray sum: 6
```

Group Anagrams

[48]:
```
def group_anagrams(strs):
    anagrams = {}

    for s in strs:
        key = ''.join(sorted(s))
        if key not in anagrams:
            anagrams[key] = []
        anagrams[key].append(s)

    return list(anagrams.values())

# Example usage
strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
result = group_anagrams(strs)
print("Grouped anagrams:", result)  # Output: [['eat', 'tea', 'ate'], ['tan',␣
 ↪'nat'], ['bat']]
```

```
Grouped anagrams: [['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
```

[49]:
```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def max_depth(root):
    if not root:
        return 0
    return max(max_depth(root.left), max_depth(root.right)) + 1

# Example usage
root = TreeNode(3)
root.left = TreeNode(9)
root.right = TreeNode(20, TreeNode(15), TreeNode(7))
result = max_depth(root)
print("Maximum depth of binary tree:", result)  # Output: 3
```

```
Maximum depth of binary tree: 3
```

Implement Queue Using Stacks

[50]:
```
class MyQueue:
    def __init__(self):
        self.stack1 = []
        self.stack2 = []

    def push(self, x):
        self.stack1.append(x)

    def pop(self):
        if not self.stack2:
            while self.stack1:
                self.stack2.append(self.stack1.pop())
        return self.stack2.pop()

    def peek(self):
        if self.stack2:
            return self.stack2[-1]
        return self.stack1[0]

    def empty(self):
        return not self.stack1 and not self.stack2

# Example usage
queue = MyQueue()
queue.push(1)
queue.push(2)
print("Peek:", queue.peek())  # Output: 1
print("Pop:", queue.pop())      # Output: 1
print("Is empty:", queue.empty())  # Output: False
```

```
Peek: 1
Pop: 1
Is empty: False
```

Kth Largest Element in an Array

[51]:
```
import heapq

def find_kth_largest(nums, k):
    return heapq.nlargest(k, nums)[-1]

# Example usage
nums = [3, 2, 1, 5, 6, 4]
k = 2
result = find_kth_largest(nums, k)
print("Kth largest element:", result)  # Output: 5
```

```
Kth largest element: 5
```

Rotate Image

[53]:
```
def rotate(matrix):
    n = len(matrix)
    for i in range(n):
        for j in range(i, n):
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]
    for i in range(n):
        matrix[i].reverse()

# Example usage
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
rotate(matrix)
print("Rotated image:")
for row in matrix:
    print(row)
```

```
Rotated image:
[7, 4, 1]
[8, 5, 2]
[9, 6, 3]
```

Deapth First Search

[5]:
```
def has_cycle(graph, node, visited, stack):
    visited.add(node)
    stack.add(node)
    for neighbor in graph[node]:
        if neighbor not in visited:
            if has_cycle(graph, neighbor, visited, stack):
                return True
        elif neighbor in stack:
            return True
    stack.remove(node)
    return False

# Example graph with a cycle (0 -> 1 -> 2 -> 0)
graph_with_cycle = {
    0: [1],
    1: [2],
    2: [0]
}
```

```python
# Example graph without a cycle (0 -> 1 -> 2)
graph_without_cycle = {
    0: [1],
    1: [2],
    2: []
}

# Function to check for a cycle in the entire graph
def detect_cycle_in_graph(graph):
    visited = set()
    stack = set()
    for node in graph:
        if node not in visited:
            if has_cycle(graph, node, visited, stack):
                return True
    return False

# Test both graphs
graph_with_cycle_result = detect_cycle_in_graph(graph_with_cycle)
graph_without_cycle_result = detect_cycle_in_graph(graph_without_cycle)

print(f"Graph with cycle has a cycle: {graph_with_cycle_result}")
print(f"Graph without cycle has a cycle: {graph_without_cycle_result}")
```

```
Graph with cycle has a cycle: True
Graph without cycle has a cycle: False
```

12.0.1 System Design: Design a URL shortening service (like TinyURL).

Components:

API Layer: Accept long URLs and return a shortened version.

Database: Store mappings of short URLs to original URLs. Use a key-value store like Redis for speed.

URL Encoding: Use a base-62 encoding for the shortened URLs (a combination of alphanumeric characters) to generate unique IDs.

Redirection Service: When a user visits a short URL, retrieve the original URL and redirect them.

Flow: User submits a long URL. The system generates a unique short URL. The mapping is stored in the database. When the short URL is accessed, the original URL is retrieved and used for redirection. Additional Features: Analytics, custom URL aliases, expiration times.

12.0.2 How would you scale a real-time chat application?

Use WebSockets for real-time, bidirectional communication. Load Balancing: Deploy multiple chat servers to handle requests and distribute traffic via a load balancer.

Message Queue: Use a message queue (e.g., RabbitMQ) to decouple message sending from real-time updates to reduce server load.

Database: Use a highly scalable database like Cassandra or DynamoDB to store chat history.

Sharding: Split users into shards based on geography or other criteria to reduce server load. Scaling WebSocket Servers: Deploy multiple WebSocket servers behind a load balancer, and store session states in Redis for consistency.

Maximum Depth of Binary Tree

Kadane's Algorithm is a popular algorithm used to find the maximum sum of a contiguous subarray within a one-dimensional array of numbers. The key idea is to iterate through the array while maintaining the maximum sum that ends at each position.

12.0.3 System Design Questions

How would you design a distributed system to handle millions of concurrent users?

Use load balancers to distribute traffic among multiple servers. Deploy microservices to break down functionality into independent services. Use horizontal scaling to add more servers as user demand grows. Implement caching (e.g., with Redis) to reduce load on databases. For storage, use scalable databases like Cassandra or DynamoDB.

12.0.4 Discuss the design of a load balancer. What considerations would you take?

The load balancer should distribute traffic evenly among servers to prevent overload. Health checks: The load balancer should regularly check server health. Algorithms: Round-robin, least connections, or IP hash can be used to distribute traffic. Consider failover mechanisms to redirect traffic in case a server goes down.

12.0.5 What are ACID properties in a database, and why are they important?

Atomicity: Transactions are all-or-nothing.

Consistency: The database remains in a valid state before and after a transaction.

Isolation: Transactions don't interfere with each other.

Durability: Once a transaction is committed, it remains so even in case of a failure.

12.0.6 Explain the differences between SQL and NoSQL databases. When would you use each?

SQL: Structured, relational, uses tables, supports ACID. Use when consistency and complex queries are needed.

NoSQL: Unstructured, document or key-value based, scalable, good for large datasets. Use when scalability is more important than consistency.

13 DATA STRUCTURES

Fundamental Data Structures

```
Arrays:
    Definition: Arrays are collections of elements (of the same data type) stored in contiguous
    Advantages: Direct access to elements using indices.
    Disadvantages: Fixed size, inefficient insertions/deletions.
    Time Complexities:
        Access: O(1)
        Insertion/Deletion: O(n) (for shifting elements)

Linked Lists:
    Definition: A sequence of elements where each element points to the next. A linked list is
    Advantages: Dynamic sizing, efficient insertions/deletions.
    Disadvantages: Slower access time (O(n)) since it requires traversal from the head.
    Time Complexities:
        Access: O(n)
        Insertion/Deletion: O(1) (if the position is known)

Stacks:
    Definition: A LIFO (Last In, First Out) data structure where elements are added and remove
    Operations:
        Push (add to the top): O(1)
        Pop (remove from the top): O(1)
    Usage: Undo functionality, recursion.

Queues:
    Definition: A FIFO (First In, First Out) data structure where elements are added at the re
    Operations:
        Enqueue (add to the rear): O(1)
        Dequeue (remove from the front): O(1)
    Usage: Task scheduling, buffering.

Trees:
    Definition: A hierarchical data structure with a root node and child nodes. Each node has
    Types:
        Binary Trees, Binary Search Trees (BST), AVL Trees, Heaps.
    Advantages: Efficient searching and sorting (especially with BST).
    Time Complexities (BST):
        Search: O(log n) (if balanced), O(n) (if unbalanced)
        Insertion/Deletion: O(log n) (if balanced), O(n) (if unbalanced)

Graphs:
    Definition: A collection of nodes (vertices) connected by edges. Can be directed or undire
    Types:
        Directed, undirected, weighted, unweighted.
    Usage: Representing networks, social connections, pathfinding.
    Time Complexities:
        BFS/DFS: O(V + E) where V is vertices and E is edges.
```

Sorting and Searching Quicksort

```
Algorithm:
    Pick a pivot element.
    Partition the array such that elements less than the pivot are on the left and elements gre
    Recursively apply quicksort on the left and right partitions.
Time Complexity:
    Best/Average case: O(n log n)
    Worst case: O(n²) (if the pivot selection is poor)
Space Complexity: O(log n) due to recursion stack.
```

Mergesort

```
Algorithm:
    Divide the array into two halves.
    Recursively sort both halves.
    Merge the two sorted halves.
Time Complexity: O(n log n) for all cases.
Space Complexity: O(n) due to the additional array needed for merging.
```

Binary Search

```
Algorithm:
    Start with the middle element.
    If the target is less than the middle, search the left half; otherwise, search the right ha
    Repeat until the target is found or the search space is exhausted.

Time Complexity: O(log n) for a sorted array.

Space Complexity: O(1) for iterative binary search, O(log n) for recursive binary search.
```

Graph Algorithms BFS (Breadth-First Search)

```
Algorithm: Traverse the graph layer by layer using a queue. Visit all nodes at the current dept
Time Complexity: O(V + E)
Usage: Finding shortest paths in unweighted graphs.
```

DFS (Depth-First Search)

```
Algorithm: Traverse the graph depth-wise using a stack or recursion. Visit as far down as possi
Time Complexity: O(V + E)
Usage: Detecting cycles, topological sorting.
```

Dijkstra's Algorithm

```
Algorithm: Finds the shortest path from a source node to all other nodes in a weighted graph.
Time Complexity: O((V + E) log V) using a priority queue (min-heap).
Usage: Shortest path finding in weighted graphs.
```

7. Explain the time and space complexity of common sorting algorithms:

Bubble Sort: Time Complexity: O(n^2) in the worst and average case, O(n) in the best case (if already sorted). Space Complexity: O(1) (in-place).

Quick Sort: Time Complexity: O(n log n) on average, O(n^2) in the worst case (when the pivot is poorly chosen). Space Complexity: O(log n) due to recursive stack calls (in-place).

Merge Sort: Time Complexity: O(n log n) in all cases (due to the divide-and-conquer approach). Space Complexity: O(n) because it requires additional space for merging.

Heap Sort: Time Complexity: O(n log n). Space Complexity: O(1) (in-place).

14 Data Types

1. Primitive Data Types

Primitive data types are the most basic data types provided by a programming language. They represent simple values and are usually predefined by the language. Here are common primitive data types found in most programming languages:

```
Integer (int): Used to represent whole numbers (e.g., 1, -5, 42).
    Example: int age = 25;

Floating-point (float, double): Used to represent numbers with decimal points (e.g., 3.14, -0.
    Example: float price = 19.99;

Character (char): Used to represent single characters (e.g., 'a', 'B', '$'). In some languages
    Example: char grade = 'A';

Boolean (bool): Represents true or false values (often used in conditional statements).
    Example: bool is_active = true;

String: Although not considered primitive in all languages, strings are often treated like a p
    Example: string name = "Alice";

Null: Represents the absence of a value. Some languages use null, None, or nil to represent th
    Example: Object obj = null;
```

15 OOPS IN PYTHON

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "objects," which can contain data and methods. The four fundamental concepts of OOP are Inheritance, Polymorphism, Encapsulation, and Abstraction.

15.1 1 Inheritance

Inheritance allows a class (child class) to inherit attributes and methods from another class (parent class). It promotes code reusability and creates a hierarchy between classes. Definition:

```
Parent Class (Base Class): The class whose properties are inherited.
Child Class (Derived Class): The class that inherits properties from the parent class.
```

[35]:
```python
# Parent Class
class Animal:
    def __init__(self, name):
        self.name = name
    def speak(self):
        return f"{self.name} makes a sound."

# Child Class inheriting from Animal
class Dog(Animal):
    def speak(self):
        return f"{self.name} barks."

# Child Class inheriting from Animal
class Cat(Animal):
    def speak(self):
        return f"{self.name} meows."

# Creating instances
dog = Dog("Buddy")
cat = Cat("Whiskers")

print(dog.speak())  # Output: Buddy barks.
print(cat.speak())  # Output: Whiskers meows.
```

```
Buddy barks.
Whiskers meows.
```

15.2 2. Polymorphism

Polymorphism means "many forms," and it allows objects to be treated as instances of their parent class, even when they are actually instances of different child classes. It is often implemented through method overriding. Definition:

```
Method Overriding: A child class can provide a specific implementation of a method that is alr
```

[36]:
```
# Parent Class
class Shape:
    def area(self):
        pass

# Child Class 1
class Square(Shape):
    def __init__(self, side):
        self.side = side

    def area(self):
        return self.side ** 2

# Child Class 2
class Circle(Shape):
    def __init__(self, radius):
```

```python
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2

# Function that uses polymorphism
def print_area(shape):
    print(f"Area: {shape.area()}")

square = Square(4)
circle = Circle(5)

print_area(square)  # Output: Area: 16
print_area(circle)  # Output: Area: 78.5
```

```
Area: 16
Area: 78.5
```

15.3 3. Encapsulation

Encapsulation refers to restricting direct access to some of an object's components. It is the bundling of data and methods that operate on the data within a single unit (a class), and it restricts access to internal variables via public methods.

Definition:

```
Private Attributes/Methods: Variables and methods that cannot be accessed directly from outsid
```

[38]:
```python
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # Private variable

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount

    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount
        else:
            print("Insufficient funds")

    def get_balance(self):
        return self.__balance

account = BankAccount(1000)
account.deposit(500)
account.withdraw(200)
print(account.get_balance())  # Output: 1300
```

```
1300
```

15.4 4. Abstraction

Abstraction involves hiding the internal details and showing only the essential features of an object. This is often achieved using abstract classes and methods.

Definition:

```
Abstract Class: A class that cannot be instantiated and often contains abstract methods.
Abstract Method: A method that is declared, but contains no implementation (forcing child clas
```

[39]:
```python
from abc import ABC, abstractmethod

# Abstract Class
class Vehicle(ABC):

    @abstractmethod
    def start_engine(self):
        pass

# Concrete Class implementing the abstract method
class Car(Vehicle):
    def start_engine(self):
        print("Car engine started.")

# Concrete Class implementing the abstract method
class Motorcycle(Vehicle):
    def start_engine(self):
        print("Motorcycle engine started.")

# Creating instances
car = Car()
motorcycle = Motorcycle()

car.start_engine()         # Output: Car engine started.
motorcycle.start_engine()  # Output: Motorcycle engine started.
```

```
Car engine started.
Motorcycle engine started.
```

16 Types of Inheritance

16.1 1. Single Inheritance

In single inheritance, a child class inherits from only one parent class. This is the most basic form of inheritance.

[40]:
```python
# Parent class
class Animal:
    def speak(self):
        return "Animal speaks"

# Child class inheriting from Animal
class Dog(Animal):
    def bark(self):
        return "Dog barks"

# Creating instance of Dog
dog = Dog()
print(dog.speak())  # Output: Animal speaks
print(dog.bark())   # Output: Dog barks
```

```
Animal speaks
Dog barks
```

16.2 2. Multiple Inheritance

In multiple inheritance, a child class inherits from more than one parent class. The child class gets access to attributes and methods of all the parent classes.

[41]:
```python
# Parent class 1
class Father:
    def work(self):
        return "Father works"

# Parent class 2
class Mother:
    def cook(self):
        return "Mother cooks"

# Child class inherits from both Father and Mother
class Child(Father, Mother):
    def play(self):
        return "Child plays"

# Creating instance of Child
child = Child()
print(child.work())  # Output: Father works
print(child.cook())  # Output: Mother cooks
print(child.play())  # Output: Child plays
```

```
Father works
Mother cooks
Child plays
```

16.3 3. Multilevel Inheritance

In multilevel inheritance, a class inherits from a derived class, creating a chain of inheritance. In other words, a class acts as both a child class and a parent class.

[42]:
```python
# Parent class
class Animal:
    def speak(self):
        return "Animal speaks"

# Child class inheriting from Animal
class Dog(Animal):
    def bark(self):
        return "Dog barks"

# Grandchild class inheriting from Dog
class Puppy(Dog):
    def weep(self):
        return "Puppy weeps"

# Creating instance of Puppy
puppy = Puppy()
print(puppy.speak())  # Output: Animal speaks
print(puppy.bark())   # Output: Dog barks
print(puppy.weep())   # Output: Puppy weeps
```

```
Animal speaks
Dog barks
Puppy weeps
```

16.4 4. Hierarchical Inheritance

In hierarchical inheritance, multiple child classes inherit from a single parent class. This structure forms a hierarchy with one parent and several child classes.

[43]:
```python
# Parent class
class Animal:
    def speak(self):
        return "Animal speaks"

# Child class 1 inheriting from Animal
class Dog(Animal):
    def bark(self):
        return "Dog barks"

# Child class 2 inheriting from Animal
class Cat(Animal):
    def meow(self):
        return "Cat meows"
```

```python
# Creating instances of Dog and Cat
dog = Dog()
cat = Cat()
print(dog.speak())  # Output: Animal speaks
print(dog.bark())   # Output: Dog barks
print(cat.speak())  # Output: Animal speaks
print(cat.meow())   # Output: Cat meows
```

```
Animal speaks
Dog barks
Animal speaks
Cat meows
```

16.5 5. Hybrid Inheritance

Hybrid inheritance is a combination of two or more types of inheritance. It usually involves a mix of single, multiple, hierarchical, or multilevel inheritance. Python supports hybrid inheritance, and it is often used in more complex scenarios.

[44]:
```python
# Parent class
class Animal:
    def speak(self):
        return "Animal speaks"

# Child class 1 inheriting from Animal
class Dog(Animal):
    def bark(self):
        return "Dog barks"

# Child class 2 inheriting from Animal
class Cat(Animal):
    def meow(self):
        return "Cat meows"

# Another class inheriting from Dog and Cat
class Hybrid(Dog, Cat):
    def growl(self):
        return "Hybrid growls"

# Creating instance of Hybrid
hybrid = Hybrid()
print(hybrid.speak())  # Output: Animal speaks
print(hybrid.bark())   # Output: Dog barks
print(hybrid.meow())   # Output: Cat meows
print(hybrid.growl())  # Output: Hybrid growls
```

```
Animal speaks
Dog barks
```

```
Cat meows
Hybrid growls
```

[ ]:

```

```