In [ ]:

In [ ]:
```
#Q1. Explain the key features of Python that make it a popular choice for programming
```

Ans-Python's popularity stems from its simplicity, readability, and ease of learning, making it ideal for beginners and experts alike. Its extensive standard library supports a wide range of tasks, while frameworks like Django and Flask enable efficient web development. Python is versatile, with applications in data analysis, machine learning, automation, and more. It also boasts strong community support and cross-platform compatibility. Its dynamic typing and interpreted nature allow quick development and debugging. Python's integration capabilities with other languages and tools further enhance its usability across various industries.

In [ ]:

In [ ]:
```
#2 Q. Describe the role of predefined keywords in Python and provide examples of how they are used in a program
```

Ans - Python keywords are special reserved words that have specific meanings and purposes and can’t be used for anything but those specific purposes. These keywords are always available—you’ll never have to import them into your code.

Example print is only used to print a statement . it cannot be useed as identifier

In [ ]:
```
 Example print is only used to print a statement . it cannot be useed as identifier.
```

In [ ]:
```
print ("hello ananya")
```

```
hello ananya
```

In [ ]:
```
#3 Q. Compare and contrast mutable and immutable objects in Python with examples
```

Ans Python, being a dynamically-typed language, offers a variety of mutable and immutable objects. Lists, dictionaries, and sets are instances of mutable objects, while numbers, strings, and tuples are instances of immutable objects.

In [ ]:
```
#ex
#list
list = [1,2,3]
list[0] = 'a'
list
```

Out[ ]:
```
['a', 2, 3]
```

In [ ]:
```
#dictionary
dict = {'name' : 'ananya', 'age' : 24}
dict['age'] = 25
dict
```

Out[ ]:
```
{'name': 'ananya', 'age': 25}
```

In [ ]:
```
#set
set = {1,2,3}
set.add(4)
set
```

Out[ ]:
```
{1, 2, 3, 4}
```

In [ ]:
```
#Q4 Discuss the different types of operators in Python and provide examples of how they are used
```

Types of Python Operators

Python language supports various types of operators, which are: 1 Arithmetic Operators 2 Comparison (Relational) Operators 3 Assignment Operators 4 Logical Operators 5 Bitwise Operators 6 Membership Operators 7 Identity Operators

In [ ]:
```
#Example of Python Arithmetic Operators
a = 21
```

```
b = 10
# Addition
print ("a + b : ", a + b)
# Subtraction
print ("a - b : ", a - b)
# Multiplication
print ("a * b : ", a * b)
# Division
print ("a / b : ", a / b)
# Modulus
print ("a % b : ", a % b)
# Exponent
print ("a ** b : ", a ** b)
# Floor Division
print ("a // b : ", a // b)
```

```
a + b :  31
a - b :  11
a * b :  210
a / b :  2.1
a % b :  1
a ** b :  16679880978201
a // b :  2
```

In [ ]:
```
# Python Comparison Operators
a = 4
b = 5
# Equal
print ("a == b : ", a == b)
# Not Equal
print ("a != b : ", a != b)
# Greater Than
print ("a > b : ", a > b)
# Less Than
print ("a < b : ", a < b)
# Greater Than or Equal to
print ("a >= b : ", a >= b)
# Less Than or Equal to
print ("a <= b : ", a <= b)
```

```
a == b :  False
a != b :  True
a > b :  False
a < b :  True
a >= b :  False
a <= b :  True
```

In [ ]:
```
#Python Assignment Operators
# Assignment Operator
a = 10
# Addition Assignment
a += 5
print ("a += 5 : ", a)
# Subtraction Assignment
a -= 5
print ("a -= 5 : ", a)
# Multiplication Assignment
a *= 5
print ("a *= 5 : ", a)
# Division Assignment
a /= 5
print ("a /= 5 : ",a)
# Remainder Assignment
a %= 3
print ("a %= 3 : ", a)
# Exponent Assignment
a **= 2
print ("a **= 2 : ", a)
# Floor Division Assignment
a //= 3
print ("a //= 3 : ", a)
```

```
a += 5 :  15
a -= 5 :  10
a *= 5 :  50
a /= 5 :  10.0
a %= 3 :  1.0
a **= 2 :  1.0
a //= 3 :  0.0
```

In [ ]:
```
#Python Bitwise Operators
a = 60 # 60 = 0011 1100
b = 13 # 13 = 0000 1101
# Binary AND
c = a & b # 12 = 0000 1100
```

```
print ("a & b : ", c)
# Binary OR
c = a | b # 61 = 0011 1101
print ("a | b : ", c)
# Binary XOR
c = a ^ b # 49 = 0011 0001
print ("a ^ b : ", c)
# Binary Ones Complement
c = ~a; # -61 = 1100 0011
print ("~a : ", c)
# Binary Left Shift
c = a << 2; # 240 = 1111 0000
print ("a << 2 : ", c)
# Binary Right Shift
c = a >> 2; # 15 = 0000 1111
print ("a >> 2 : ", c)
```

```
a & b :  12
a | b :  61
a ^ b :  49
~a :  -61
a << 2 :  240
a >> 2 :  15
```

In [ ]:
```
# Python Logical Operators
x = 5
y = 10
if x > 3 and y < 15:
  print("Both x and y are within the specified range")
```

```
Both x and y are within the specified range
```

In [ ]:
```
#Python Membership Operators
fruits = ["apple", "banana", "cherry"]
if "banana" in fruits:
    print("Yes, banana is a fruit!")
else:
    print("No, banana is not a fruit!")
```

```
Yes, banana is a fruit!
```

In [ ]:
```
# Python Identity Operators
x = 10
y = 5
if x is y:
    print("x and y are the same object")
else:
    print("x and y are not the same object")
```

```
x and y are not the same object
```

In [ ]:
```
#Q5.  Explain the concept of type casting in Python with example
```

Ans- Type Casting is the method to convert the Python variable datatype into a certain data type in order to perform the required operation by users. In this article, we will see the various techniques for typecasting. There can be two types of Type Casting in Python:

Python Implicit Type Conversion

Python Explicit Type Conversion

In [ ]:
```
# Python program to demonstrate
# implicit type Casting

# Python automatically converts
# a to int
a = 7
print(type(a))

# Python automatically converts
# b to float
b = 3.0
print(type(b))
```

```
# Python automatically converts
# c to float as it is a float addition
c = a + b
print(c)
print(type(c))

# Python automatically converts
# d to float as it is a float multiplication
d = a * b
print(d)
print(type(d))
```

```
<class 'int'>
<class 'float'>
10.0
<class 'float'>
21.0
<class 'float'>
```

In [ ]:
```
#Q6.  How do conditional statements work in Python? Illustrate with examples
```

In [ ]:
```
num = 5

if num > 0:
    print("The number is positive.")
```

```
The number is positive.
```

In [ ]:
```
num = 5

if num > 0:
    print("The number is positive.")
```

```
The number is positive.
```

In [ ]:
```
score = 85

if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"

print("Your grade is:", grade)
```

```
Your grade is: B
```

In [ ]:
```
#Q7.  Describe the different types of loops in Python and their use cases with examples.
```

Ans- A loop is a control flow statement in Python that allows you to execute a piece of code repeatedly until a specific condition is met. Loops are necessary for operations that require repetitive execution, such as iterating through a Python list of items or doing calculations many times.

In [ ]:
```
# python while loop
count = 0
while (count < 10):
 print ('The count is:', count)
 count = count + 1
print ("Good bye!")
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
The count is: 9
Good bye!
```

In [ ]:
```
#python for loop
```

```
for letter in 'Python': # First Example
 print ('Current Letter :', letter)
fruits = ['guava', 'apple', 'mango']
for fruit in fruits: # Second Example
 print ('Current fruit :', fruit)
print ("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : guava
Current fruit : apple
Current fruit : mango
Good bye!
```

In [ ]:

```
#python nested loop
i = 2
while(i < 100):
 j = 2
 while(j <= (i/j)):
   if not(i%j): break
   j = j + 1
 if (j > i/j) : print (i, " is prime")
 i = i + 1
print ("Good bye!")
```

```
2  is prime
3  is prime
5  is prime
7  is prime
11  is prime
13  is prime
17  is prime
19  is prime
23  is prime
29  is prime
31  is prime
37  is prime
41  is prime
43  is prime
47  is prime
53  is prime
59  is prime
61  is prime
67  is prime
71  is prime
73  is prime
79  is prime
83  is prime
89  is prime
97  is prime
Good bye!
```