1. Explain the key features of python that make it a popular choice for programming

ANS-

Python is a high-level, general-purpose, interpreted, object-oriented programming language with dynamic semantics. It was developed by Guido van Rossum and released in 1991. The name "Python" is a reference to the British comedy group Monty Python, and was chosen to be both fun and easy to use

1. It is widely used in the industry
2. It is free and open source
3. Data industry
4. A lot of libraries
5. Frontend and backend development
6. Automation and image processing

2. Describe the role of predefined keywords in python and provide example of how they are used in a program.

ANS-

A keyword refers to a predefined word that python reserves for working programs that have a specific meaning, You can't use a keyword anywhere else. Python Identifiers are the different values that a programmer can use to define various variables, integers, functions, and classes.

```
name = input("Enter your name: ")
print("Name: ", name)
```

3. Compare and contrast mutable and immutable objects in python with examples

ANS-

Mutable	Immutable
The objects can be modified after the creation as well.	Objects cannot be modified after the creation of the objects.
Classes that are mutable are not considered final.	Classes that are immutable are considered final.
Thread unsafe.	Thread-safe.
Classes are not made final for the mutable objects.	Classes are made final for the immutable objects.
Example: Lists, Dicts, Sets, User-Defined Classes, Dictionaries, etc.	Example: int, float, bool, string, Unicode, tuple, Numbers, etc.

Mutable example

list_cont = [1,2,3,4,5,4.5,True,'AJAY']

list_cont

```
output – [1, 2, 3, 4, 5, 4.5, True, 'Adhinath']
```

list_cont [7] = 'Mahesh'

list_cont
[1, 2, 3, 4, 5, 4.5, True, 'Mahesh']

Immutable example-

object/container whose state or value cannot be changed after they are created are called as immutable objects or container

```
b ="Mahesh"
b
```

```
output -- 'Mahesh'
```

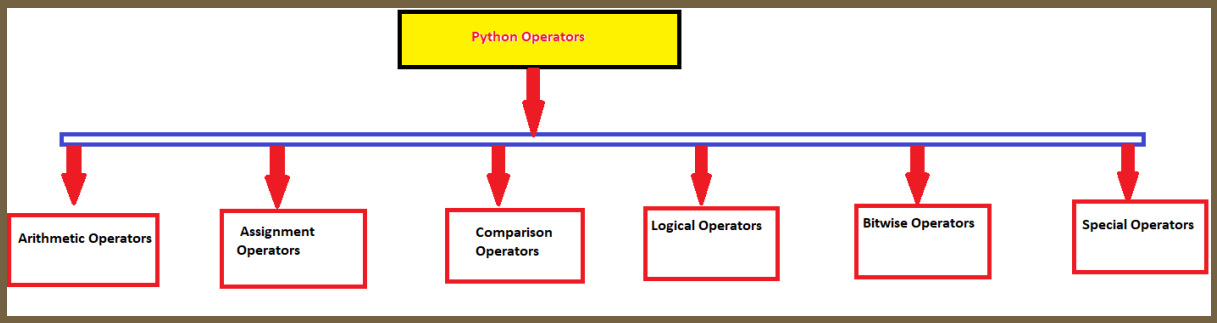
```
b[0] ="P"
```

```
---------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 b[0]="P"

TypeError: 'str' object does not support item assignment
```

4. Discuss the different types of operators in python and provide examples of how they are used.

ANS--



1. Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, Division, Floor Division, Modulo, Power.

For example

a=3

b=4

a+b output = 7

a-b output = -1

a*b output = 12

a/b output = 0.75

a%b output = 3

2**4 output = 81

2. Assignment Operators

Assignment operators are used to assign values to variables.

For example

```
# assign 10 to a
a = 10
# assign 5 to b
b = 5
# assign the sum of a and b to a
a += b      # a = a + b
print(a)
```

```
# Output: 15
```

3. Comparison Operators

Comparison operators compare two values/variables and return a boolean result: True or False.
For example,

```
a = 5
b = 2
# equal to operator
print('a == b =', a == b)

# not equal to operator
print('a != b =', a != b)

# greater than operator
print('a > b =', a > b)

# less than operator
print('a < b =', a < b)

# greater than or equal to operator
print('a >= b =', a >= b)

# less than or equal to operator
print('a <= b =', a <= b)
```

output-

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

```
a <= b = False
```

4. Logical Operators

Logical operators are used to check whether an expression is True or False. They are used in decision-making.
For example

```
x = 5
print(x > 0 and x < 10)

n = 25
print(n % 2 == 0 or n % 3 == 0)
```

output—

```
True
False
```

```
# logical NOT
print(not True)        # False
```

output—

False

5. Python Bitwise operators

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.
For example, 2 is 10 in binary, and 7 is 111.
In the table below: Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

```
bin (10)
```

output-- `'0b1010'`

6. Special Operators

Python language offers some special types of operators like the **identity** operator and the **membership** operator. They are described below with examples.

Identity operators

In Python, is and is not are used to check if two values are located at the same memory location.
For example

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

print(x1 is not y1)  # prints False

print(x2 is y2)  # prints True

print(x3 is y3)  # prints False
```

output—

```
False
True
False
```

Membership operators

In Python, in and not in are the membership operators. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

For example-

```
message = 'Hello world'
dict1 = {1:'a', 2:'b'}

# check if 'H' is present in message string
print('H' in message)  # prints True

# check if 'hello' is present in message string
print('hello' not in message)  # prints True

# check if '1' key is present in dict1
print(1 in dict1)  # prints True

# check if 'a' key is present in dict1
print('a' in dict1)  # prints False
```

output—

```
True
True
True
```

```
False
```

5. Explain the concept of type casting in python with examples.

ANS-

The process of changing the data type of a value or object OR

Type Casting is the method to convert the Python variable data type into a certain data type in order to perform the required operation.

a = "2"

b = 3

int(a) + b

output –

```
5
```

a = "2"

print(type(a))

print(type(int(a)))

output—

```
<class 'str'>
<class 'int'>
```

There can be two types of Type Casting in Python

1. Implicit

   Converts the data type into another data type automatically

For example-

```
# c to float as it is a float addition
a = 7
b = 3.0
c = a + b
print(c)
print(type(c))
```

output—

```
<class 'float'>
21.0
```

2. Explicit –

   convert data type using inbuilt function--str,float,int,bool

For example-

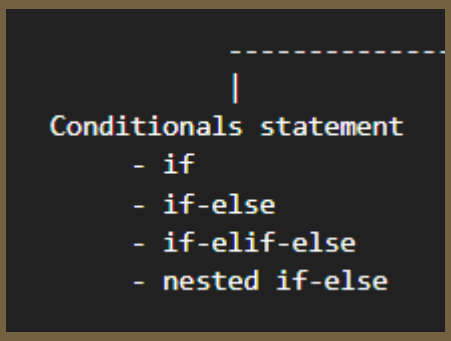
```
a =2
type(a)
int(a)
```

output—

```
2
```

6. How do conditional statements work in python? illustrate with examples

ANS –
Conditional Statements are statements in Python that provide a choice for the control flow based on a condition. It means that the control flow of the Python program will be decided based on the outcome of the condition



If statement-
If the simple code of block is to be performed if the condition holds then the if statement is used. Here the condition mentioned holds then the code of the block runs otherwise not.
For example-

```
a = 100
if a > 0:
    print("the number is grather than 0")
```

output-

```
the number is grather than 0
```

if-else statement-
In a conditional if Statement the additional block of code is merged as an else statement which is performed when if condition is false
For example-

```
weather = "Rainy"
if weather == "Rainy":
    print("I will not pleay cricket ")
else:
    print("I will watch TV")
```

output—

```
I will not pleay cricket
```

If-elif-else statement-

The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final “else” statement will be executed.
For example-

```
a = 10
if a > 100:
    print("This block will executed if a is greather than 100")
elif a < 100:
    print("This block will executed if a is lesser than 100")
else:
    print("THe number is equals to 100")
```

output—

```
This block will executed if a is lesser than 100
```

Nested if-else statement-

Nested if..else means an if-else statement inside another if statement. Or in simple words first, there is an outer if statement, and inside it another if – else statement is present and such type of statement is known as nested if statement. We can use one if or else if statement inside another if or else if statements.
For example-

```
x = 1
y = 6
if x > 5:
    if x < 5:
        print("Both x and y is greather than 5")
    else:
        print("x is greather than 5 but y is less than 5")
else:
    print("x is not greather than 5")
```

output—

```
x is not greather than 5
```

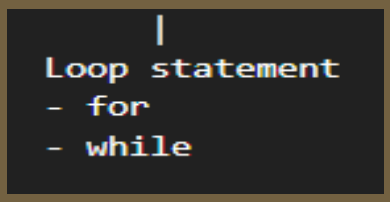
7. **Describe the different types of loops in python and their use case with examples.**

ANS—

Loop statement—

it allows you execute a block of code repeatedly.

Two types of loop statement.



1. While loop—

While loop is a control flow statement that allows you to execute a block of code repeatedly while a given condition is true.

For example—

```
n = 7
i = 1
while i < n:
    print(i)
    i = i + 1
```

output—

```
1
2
3
4
5
6
```

Second example—

```
n = 7
i = 1
while i < n:
    print(i)
    i = i + 1
else:
    print("This will be executed when the while is run successfully without any break")
```

output—

```
1
2
3
4
5
6
This will be executed when the while is run successfully without any break
```

2. For loop—

A for loop is a control flow statement in Python that allows you to iterate over a sequence of elements such as a list, tuple, or string.
On each iteration, the loop variable in Python will take on the value of the next item in the sequence.

For example—

```
# for loops-- iterate over a sequence of elements --strins,list
for i in "pwskills":
    print(i)
```

output—

```
p
w
s
k
i
l
l
s
```

LOOP CONTROL STATEMENT

1. Break

The break statement is used to quickly stop the loop regardless of whether the loop condition is still true. This signifies that the loop will be terminated and the next statement outside of the loop will be executed.
For example—

```
for i in range(10):
  if i == 5:
    break
  print(i)
```

Output

```
0
1
2
3
4
```

2. Continue

The continue statement is used to skip the current loop iteration and go to the next iteration. This implies that the code from the current iteration will not be performed, and the loop will continue to iterate until the loop condition is satisfied.

For example—

```
for i in range(10):
  if i % 2 == 0:
    continue
  print(i)
```

Output

```
1
3
5
7
9
```