Assignment:

Python:

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

- Simple and Readable Syntax:

Python's syntax is clear and easy to read, which makes it accessible to beginners and allows experienced programmers to write code efficiently. The use of indentation to define code blocks enhances readability and reduces the chances of errors.

- Versatility:

Python is a general-purpose language, meaning it can be used for a wide range of applications, from web development and data analysis to machine learning and automation. Its versatility is one of the reasons it's so widely adopted.

- Large Standard Library:

Python comes with a comprehensive standard library that provides modules and functions for various tasks, such as file handling, regular expressions, and even internet protocols. This reduces the need to write code from scratch for common tasks.

- Extensive Ecosystem of Third-Party Libraries:

Beyond the standard library, Python has a vast ecosystem of third-party libraries and frameworks, such as NumPy, Pandas, TensorFlow, Django, and Flask. These libraries extend Python's capabilities and allow developers to work more efficiently in specialized areas.

- Cross-Platform Compatibility:

Python is platform-independent, meaning that Python code can run on various operating systems like Windows, macOS, and Linux without modification. This makes it easier to develop and deploy applications across different environments.

- Dynamic Typing:

Python uses dynamic typing, meaning that the type of a variable is determined at runtime, which allows for more flexibility in writing code. Developers don't need to declare variable types explicitly, which can speed up the development process.

In [ ]:

```

```

2. Describe the role of predefined keywords in Python and provide examples of how they are used in a program:

- predefined keywords in python are reserved words that have special meaning and cannot be used for any other purpose. They help in defining the structure and syntax of the python language. examples:
- if, elif, else: Used for conditional statements.
- for, while: used for loops.
- True, False
- break, continue. etc.

In [2]:

```python
#example for predefined keywords
a = 50
b = 50
if a < b:
    print("b is greater than a")
elif a > b:
    print("a is greater than b")
else:
    print("both are equal")
```

```
both are equal
```

In [3]:

```python
#example for True, False:
a == b
```

Out[3]:

```
True
```

In [ ]:

```

```

3.Mutable and immutable objects.

- Mutable objects: Objects that can be changed after their creation.
- immutable objects: Objects that can be changed once they are created.

```
In [4]: #example for mutable objects:
        c = [ 1, 2, 3, 3.5, "akif", True]
        c[0] = 4
        c

Out[4]: [4, 2, 3, 3.5, 'akif', True]
```

```
In [5]: #example for immutable objects:
        d = "MD AKIF NAWAB"
        d[0] = W

        ---------------------------------------------------------------------------
        NameError                                 Traceback (most recent call last)
        Cell In[5], line 3
              1 #example for immutable objects:
              2 d = "MD AKIF NAWAB"
        ----> 3 d[0] = W

        NameError: name 'W' is not defined
```

```
In [ ]:
```

4. Discuss the different types of operators in Python and provide examples of howthey are used.

- In Python, operators are special symbols or keywords used to perform operations on variables and values. They are the foundation of most operations you perform in your code, ranging from simple arithmetic to more complex logic and manipulation.
- Arithmetic Operators: '+', '-', '', '/', '%', '*', '//'
- Comparison Operators: '==', '!=', '>', '<', '>=', '<='
- Logical Operators: 'and', 'or', 'not'
- Bitwise Operators: '&', '|', '^', '~', '<<', '>>'
- Assignment Operators: '=', '+=', '-=', '=', '/=', '%=', '*=', '//='
- Identity Operators: 'is', 'is not'
- Membership Operators: 'in', 'not in'

```
In [6]: # example for arithemetic operators:
        e = 5
        f = 10
        print(e+f)

        15
```

```
In [7]: True or False

Out[7]: True
```

```
In [8]: # example for logical operator:
        print(True|False)

        True
```

```
In [9]: #example for assignment operator:
        x = 5
        x+=5
        print(x)

        10
```

```
In [ ]:
```

5. type casting in python

- Type casting in Python refers to the process of converting a variable from one data type to another. It allows you to change the type of data stored in a variable to suit the needs of a particular operation or function.

```
In [10]: #int to string:
         num = 10
         num = str(num)
         print(num)
         type(num)

         10
Out[10]: str
```

```
In [11]: #string to int:
         name = "9"
         name = int(name)
         print(name)
         type(name)

         9
Out[11]: int
```

```
In [ ]:
```

6. Conditional Statements in Python:

- Conditional statements in Python allow you to execute specific blocks of code depending on whether certain conditions are met. These statements control the flow of execution in a program by using conditions that evaluate to either True or False. Key Conditional Statements.

1. 'if' Statement
2. 'elif' (else if) Statement
3. 'else' Statement

```
In [13]: #example for conditional statement:
         weather = "cool"
         if weather == "rainy":
             print("i will not go to ground")
         elif weather == "sunny":
             print("i will go to ground")
         else:
             print("i will stay in home")

         i will stay in home
```

```
In [ ]:
```

7. types of loops with examples

- loops in Python allow you to execute a block of code repeatedly, either for a specific number of times or until a certain condition is met. There are two primary types of loops in Python:

- 'for' Loop

- 'while' Loop

- for Loop:

- The for loop is used to iterate over a sequence (such as a list, tuple, dictionary, set, or string) or other iterable objects. It allows you to execute a block of code once for each item in the sequence.

- while loop:

A while loop in Python is used to repeatedly execute a block of code as long as a specified condition remains True. It is a control flow statement that allows for iterative execution based on a condition. The condition is checked before each iteration, and if it evaluates to True, the code block inside the loop is executed. If the condition becomes False, the loop terminates, and the program continues with the next statement following the loop.

```
In [14]: #example for "for" loop:
         for i in range(5):
             print(i)

         0
         1
         2
         3
         4
```

In [15]:

```
y = 6
p = 10
while p > y:
    print(y)
    y = y + 1
```

```
6
7
8
9
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js