python 1st assignment

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

- The key features of python that make it a popular choice for programming are:-
  Here are the key features of Python that make it popular:
  a). **Easy to Learn & Use** – Simple syntax, similar to English.
  b). **Interpreted Language** – Runs code line-by-line, great for debugging.
  c). **Versatile** – Used in web development, data science, automation, AI, etc.
  d). **Large Standard Library** – Built-in modules for various tasks.
  e). **Community Support** – Active community with plenty of tutorials and libraries.
  f). **Cross-Platform** – Runs on Windows, macOS, Linux, etc.
  g). **Object-Oriented & Functional** – Supports multiple programming paradigms.
  h). **Extensible & Embeddable** – Can integrate with C/C++ and other languages.

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 as identifiers They define the syntax and structure of the Python language.

Example:

a.) if, else, elif – used for conditional statements:

---

[12]

0s

```
x = 10



if x > 0:

    print("Positive")
```

```
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

```
Positive
```

---

b.) for, while – used for loops:

---

[5]

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

---

c.)def – defines a function:

---

[15]

```
def greet(name):
    print("Hello", name)
```

---

d.)class – defines a class:

---

[11]

```
class Dog:
    pass
```

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

- Mutable vs Immutable Objects in Python:-
  (Mutable)

Definition: Can be changed after creation.

Examples: list, dict, set, bytearray.

Memory Behavior: Changes in place (same ID).

Use Case: When data needs to change.

(Immutable)

Definition: Cannot be changed after creation.

Examples: int, float, str, tuple, frozenset.

Memory Behavior: New object created on change.

Use Case: When data must remain constant

Example (Mutable):

---

[16]

0s

```
a = [1, 2, 3]

a[0] = 100

print(a)  # [100, 2, 3]
```

```
[100, 2, 3]
```

Example (Immutable):

---

[17]

0s

```
s = "hello"

s = s.replace("h", "y")

print(s)  # "yello" (new string created)
```

```
yello
```

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

- In Python, operators are special symbols or keywords used to perform operations on variables and values. There are several types of operators, each serving a different purpose.

a.)Arithmetic Operators:

Perform basic math operations.

---

[18]

0s

```
a = 10; b = 3

print(a + b)  # 13

print(a % b)  # 1
```

```
13
1
```

---

b.) Comparison Operators:

Compare values and return True or False.

---

[19]

0s

```
print(5 > 3)    # True

print(5 == 5)   # True
```

```
True
True
```

---

c.)Assignment Operators:

Assign and update variable values.

---

[20]

0s

```
x = 5

x += 2  # x = x + 2

print(x)  # 7
```

```
7
```

---

d.)Logical Operators:

Combine multiple conditions.

---

[21]

1s

```
print(True and False)  # False

print(not True)       # False
```

```
False
False
```

---

e.)Bitwise Operators:

Operate on bits (binary values).

---

[22]

0s

```
print(5 & 3)   # 1

print(5 << 1)  # 10
```

```
1
10
```

---

f.) Membership Operators:

Check if a value exists in a sequence.

---

[23]

0s

```
print('a' in 'apple')     # True

print(3 not in [1, 2, 3])  # False
```

```
True
False
```

---

g.) Identity Operators:

Check if variables refer to the same object

---

[24]

0s

```
a = [1, 2]; b = a; c = [1, 2]
```

```
print(a is b)   # True

print(a is c)   # False
```

```
True
False
```

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

- Type Casting in Python is the process of converting the data type of a variable into another type. It's useful when you need to perform operations involving different data types.

🔸 Types of Type Casting:-

a). Implicit Type Casting:

- Python automatically converts one data type to another during an operation when it's safe to do so.

example:

---

[26]

```
x = 5

y = 2.0

z = x + y

print(z)
```

```
7.0
```

---

b.) Explicit Type Casting:

- You manually convert one data type into another using functions like:
  int()
  float()
  str()
  bool()

---

[27]

```
a = "10"

b = int(a)

print(b + 5)



c = 3

d = float(c)

print(d)
```

```
15
3.0
```

6.)How do conditional statements work in Python? Illustrate with examples.

- Conditional statements in Python are used to execute certain blocks of code based on whether a condition is true or false. The main conditional statements in Python are:

a.)if Statement:

Executes a block of code only if the condition is true.

---

[28]

0s

```
x = 10


if x > 5:
    print("x is greater than 5")
```

```
x is greater than 5
```

---

b.)if-else Statement:

Provides an alternative block of code to run if the condition is false.

---

[29]

0s

```
x = 3


if x > 5:
    print("x is greater than 5")
else:
    print("x is 5 or less")
```

```
x is 5 or less
```

---

c.)if-elif-else Statement:

Allows checking multiple conditions

---

[30]

0s

```
x = 7


if x > 10:
    print("x is greater than 10")
elif x == 7:
    print("x is exactly 7")
else:
    print("x is 10 or less but not 7")
```

```
x is exactly 7
```

7.)Describe the different types of loops in Python and their use cases with examples.

- In Python, loops are used to execute a block of code repeatedly. The main types of loops in Python are:-

a.)for Loop:

The for loop is used for iterating over a sequence (like a list, tuple, dictionary, string, or range)

uses case:

- Iterating through a list of items
- Processing items in a string or file
- Looping a fixed number of times

examle:

---

[31]

0s

```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

    print(fruit)
```

```
apple
banana
cherry
```

---

b.) while Loop:

The while loop runs as long as a condition is True.

uses case:

- When the number of iterations is not known in advance
- Waiting for a condition to become False

example:

---

[33]

0s

```
count = 0
```

```
while count < 5:
    print("Count:", count)
    count += 1
```

```
Count: 0
Count: 1
Count: 2
Count: 3
Count: 4
```

---

c.)nested Loops:

Loops inside loops are called nested loops. You can use a for loop inside a while loop, or vice versa.

uses case:

- Iterating through a 2D array or matrix
- Creating patterns

example:

---

[34]

0s

```
for i in range(3):
    for j in range(2):
        print(f"i={i}, j={j}")
```

```
i=0, j=0
i=0, j=1
i=1, j=0
```

```
i=1, j=1
i=2, j=0
i=2, j=1
```

---

d.) Loop Control Statements:

These are not loops themselves, but they help control the flow of loops

- break

Used to exit the loop prematurely.

---

[35]

0s

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

```
0
1
2
3
4
```

- continue

Skips the rest of the code inside the loop for the current iteration.

---

[36]

0s

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

```
0
1
3
4
```

- else in loops

You can add an else block to loops that executes if the loop ends normally (not via break).

---

[37]

0s

```
for i in range(3):
    print(i)
else:
    print("Loop finished without break.")
```

```
0
1
2
Loop finished without break.
```

Double-click (or enter) to edit

Connected to Python 3 Google Compute Engine backend

fiber_manual_record

close