Python Programming
Assignment - 5

PAGE No.
DATE / /

1) What are bytes in python ?
why are they immutable ?

→

What are bytes in python -

-the bytes data type represent an immutable sequence of bytes, usually used in file handelling and networking

- Immutable sequence of bytes
- values range from 0 to 255
- used in file handling and networking.

```
b = bytes([66, 67, 68])
print(b)                    # b'BCD'
print(type(b))              # <class 'bytes'>
```

Why byte are immutable in python -

i) **Data safty and integrity -**
- Binary data is often shared between systems, files or networks.
- If bytes where mutable, data could be changed accidently, leading to corruption.

ii) ~~Performance optimization~~ - **Hashability -**
- Because bytes are immutable, they can be hashed.
- -this allows byte to be used as:
  - dictionary keys
  - set elements

PAGE No.
DATE / /

iii) Performance optimization -
- Immutable object are faster and memory efficient
- python can reuse bytes objects safely without copying.

iv) Consistency with strings -
- bytes is the binary equivalent of str.
- since strings are immutable, bytes are also designed to be immutable for consistency and predictability.

2) Predict the output:

```
b = bytes([65, 66, 67])
print(b)
```

Explain how numbers are converted internally.

→

```
b = bytes([65, 66, 67])
print(b)                    # b'ABC'
```

predicted output -

b'ABC'

In this code:

```
b = bytes([65, 66, 67])
print(b)
```

- python creates a bytes objects from a list of numbers.

PAGE No.
DATE / /

- Each number represents a byte value (between 0 to 255).
- Python looks at each number and checks its ASCII meaning.

So internally:

- 65 becomes A
- 66 becomes B
- 67 becomes C

When we print the bytes objects, python shows readable characters for these values, and that's why we see:

b 'ABC'

- Even though it looks like text, python is actually storing raw binary numbers, not characters. The letters shown only to make it easy for humans to read.

3) What is the difference between bytes and bytearray?
mention the mutability and use cases.

→ bytes

- mutability - immutable, once created, you cannot change individual byte.
- use cases -
  - Representing fixed binary data, like

PAGE No.
DATE / /

Constants or read only file content.
- Network protocols where data integrity is crucial.
- As keys in dictionaries (since they are hashable).

bytearray -

- mutability - mutable, you can add, remove or change bytes after creation.

- Use cases -
  - Binary data processing where you need to modify data in place.
  - Reading from and writing to binary files.
  - Network programming where you build up data buffers.

4) Predict the output:

```
ba = bytearray([65, 66, 67])
ba[0] = 97
print(ba)
```

why this allowed?

→ The predicted output is -

bytearray(b'aBC')

why this allowed -
- Bytearray is mutable binary sequence.
- Each element is a number between 0 to 255.
- Because it is mutable python allows item assignment.

PAGE No.
DATE / /

5) What is None in python?
Is it the same as 0, false, or empty string?

→

None -

None is represent the absence of value and is commonly used as a place-holder or default return value.

- represents absence of value.
- used as default return value
- similar to NULL in Java.

Is it same as 0, false or empty string -

No, it is not same as 0, false or empty string.

```
None == 0      # false
None == False  # false
None == ""     # false
```

Even though all of this may behave as false in conditions, they are different objects.

6) predict the output -

```
x = None
print(type(x))
print(x == False)
```

Explain the result !

→

```
x = None
print(type(x))          # <class 'NoneType'>
print(x == False)       # False.
```

We declare x as None value.

x = None

+ None is the special value in python.
- It is the only object of type NoneType.

When we print(type(x))
then it gives us output <class 'NoneType'>

In this statement
print(x == False)
- None and False are different objects.
- they have different meanings:
  - None → No value
  - False → boolean false.

that is why this condition gives us output False.

4) What is the boolean data type in python? List all values that evaluate to false.

→

boolean data type is called bool.

bool - the bool data type represents truth values, mainly used in decision making and conditional statements.

PAGE No.
DATE / /

- Represents true or false
- Used in decision making and conditions.
- Internally stored as integers (True → 1, False → 0).

values that Evaluate to false -

i) False

```
bool(False)     # False
```

ii) None

```
bool(None)      # False
```

iii) Numeric zero -

```
0
0.0
0j
```

iv) Empty sequences and collections -

```
""      # empty string
[]      # empty list
()      # empty tuple
{}      # empty dict
set()   # empty set
```

v) Empty range -

```
range(0)        # false
```

Ex.

```
if []:
    print("Fralse")
else:
    print("False")      # False
```

8) What is set in python?

→ 1,

→ set-

A set is an unordered collection used to store unique elements only, mainly for removing ~~elements~~ duplicates.

- unordered collection
- storing unique elements only
- mutable
- used for removing duplicates:

```
s = {10, 20, 30, 40, 10}
print(s)                    # {10, 20, 30, 40}
print(type(s))              # <class 'set'>
```

9) Predict the output:

```
s = {10, 20, 30, 10, 20}
print(s)
```

why are some values missing?

→

```
s = {10, 20, 30, 10, 20}
print(s)                    # {10, 20, 30}
```

The output is → {10, 20, 30}

why some values missing?

- set stores only unique elements.
- That is why the duplicates are missing in the set.
- It mainly used for removing duplicates.

PAGE No.
DATE / /

10) What is dynamic typing in python ?

→

Dynamic typing means that in python, you do not need to declare the data type of a variable. The data type is decided at runtime, based on the value assigned to the variable.

Ex.

```
x = 10
print(type(x))        # int

x = "Hello"
print(type(x))        # str
```

- the type changes at runtime.
x first refers to an int
later, x refers to a str object.
- the type changes at runtime.