Print function

PYTH

output main print karna

- Several ways to use print( )

1. Basic `print()` Function

- **Syntax**: `print(object(s), sep=' ', end='\n', file=sys.stdout, flush=False)`
- **Description**: The basic `print()` function is the most common way to output text in Python. It can take multiple objects to print, with optional separators, end characters, and output stream redirection.
- **Example**:

python Copy code

```python
print("Hello, World!")  # Output: Hello, World!
print("Hello", "World", sep=", ")  # Output: Hello, World
```

Print function methods

```python
print("My name is Aammar")
print("Hello, Codanics")
print("Hello", "Codanics")
print("Hello", "Codanics", sep=",")
print("Hello", "Codanics", sep=":", end="\n")
```

[12] ✓ 0.0s

```
My name is Aammar
Hello, Codanics
Hello Codanics
Hello,Codanics
Hello:Codanics
```

output main print karna

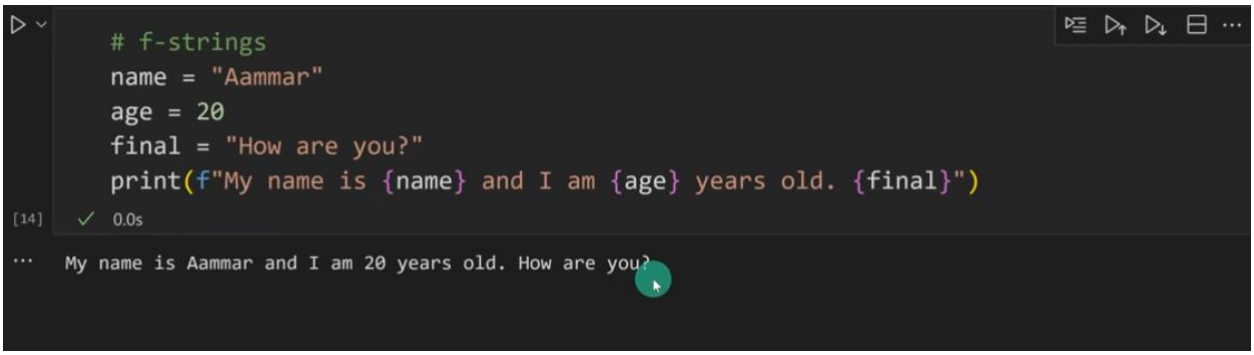
2. Formatted String Literals (f-strings)

- **Description**: Introduced in Python 3.6, f-strings allow for more readable and concise formatting by embedding expressions inside string literals prefixed by `f`.
- **Example**:

python Copy code

```python
name = "Alice"
age = 25
print(f"My name is {name} and I am {age} years old.")  # Output: My name is Ali
```

```
# Output: My name is Alice and I am 25 years old.
```



Print function

PYTHON KA C

output main print karna

3. `str.format()` Method

- **Description**: The `str.format()` method allows you to substitute values into a string, using placeholders (`{}`) for positional or keyword arguments.
- **Example**:

python Copy code

```
print("Hello, {}!".format("World"))  # Output: Hello, World!
print("My name is {0} and I am {1} years old.".format("Alice", 25))  # Output:
```

```
# Output: My name is Alice and I am 25 years old.
```

4. Percent (%) Formatting

- **Description**: An older way to format strings using the `%` operator, similar to C-style string formatting.
- **Example**:

python Copy code

```python
name = "Alice"
age = 25
print("My name is %s and I am %d years old." % (name, age))  # Output: My name
```

```
# Output: My name is Alice and I am 25 years old.
```

```python
# % formatting
name = "Aammar"
age = 20
print("My name is %s and I am %d years old." % (name, age))
#%s for string, %d for integer
```

✓ 0.0s

```
My name is Aammar and I am 20 years old.
```

output main print karna

5. Using `sys.stdout.write()`

- **Description**: This method writes to the console without adding a newline, unlike the `print()` function. It gives more control over the output format.
- **Example**:

```python
import sys
sys.stdout.write("Hello, World!")  # Output: Hello, World!
```

6. Printing Without Newline (`end=''` Parameter)

- **Description**: You can control the end character in the `print()` function using the `end` parameter to avoid newlines or use other symbols.
- **Example**:

```python
print("Hello", end='')  # Output: Hello (no newline)
print("World!")  # Output: World!
```

7. Multi-line Printing Using Triple Quotes

- **Description**: You can use triple quotes to print multi-line strings easily.
- **Example**:

```python
print("""This is a
multi-line string""")
```

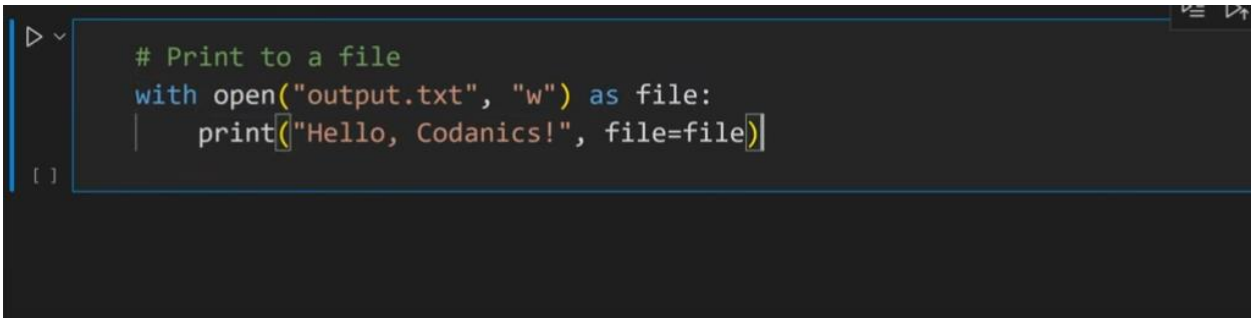
output main print karna

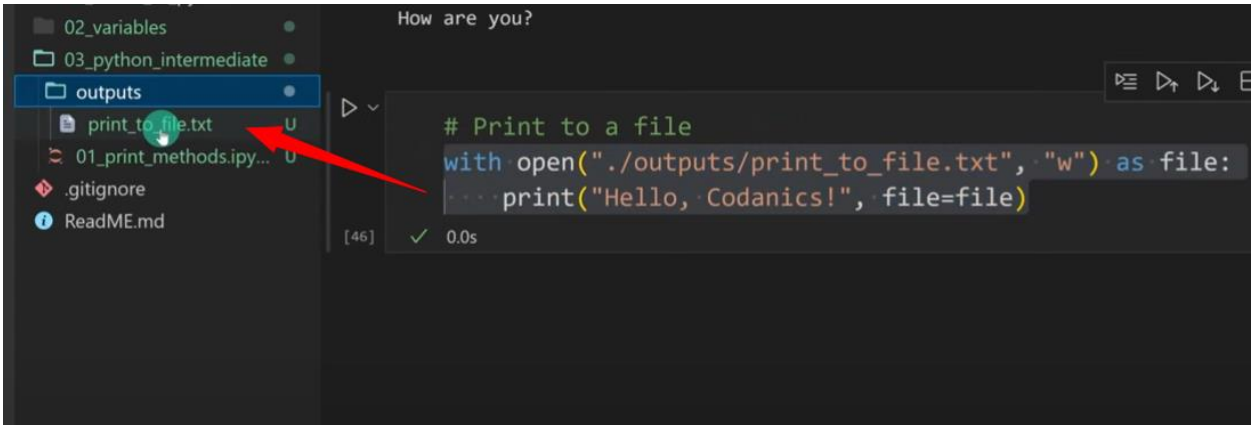
8. Printing to a File

- **Description**: You can direct the output of the `print()` function to a file using the `file` parameter.
- **Example**:

python Copy code

```
with open("output.txt", "w") as f:
    print("Hello, World!", file=f)
```





Print function All methods

PYTHOI

```python
# 1. Basic print() function
print("My name is Aammar and I am 25 years old.")

# 2. Using f-strings (Formatted String Literals)
name = "Aammar"
age = 25
print(f"My name is {name} and I am {age} years old.")

# 3. Using str.format() method
print("My name is {} and I am {} years old.".format(name, age))

# 4. Using percent (%) formatting
print("My name is %s and I am %d years old." % (name, age))

# 5. Using sys.stdout.write() (no newline at the end by default)
import sys
sys.stdout.write("My name is Aammar and I am 25 years old.\n")
```

output main print karna

9. Printing Objects with Custom `__str__` or `__repr__` Methods

- **Description**: When printing objects, Python will use the `__str__()` or `__repr__()` methods of the object to determine what to display.
- **Example**:

python Copy code

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return f"{self.name}, {self.age} years old"

p = Person("Alice", 25)
print(p)  # Output: Alice, 25 years old
```

Print function All methods

PYTHON KA CHILLA

Explanation:

1. **Basic print():** The simplest way to print a string.
2. **f-strings**: Introduced in Python 3.6, allowing variables to be embedded directly within strings.
3. **str.format():** Allows inserting values into placeholders within a string.
4. **Percent (%) formatting:** An older method that works similar to printf-style formatting in C.
5. **sys.stdout.write():** Provides finer control over the output (no newline added automatically).
6. **end=" in print():** Controls what is printed at the end (default is a newline, but can be changed).
7. **Triple quotes:** Allows multi-line strings.
8. Printing to a file: Redirects output to a file instead of the console.
9. **Custom __str__ method:** Defines how an object of the Person class should be represented when printed.