assignment

November 6, 2024

[ ]:

QUS - Explain the key features of Python that make it a popular choice for programming? ANS - Python's popularity stems from its:

Key Features:

1. Easy to Learn: Simple syntax, readable code, and forgiving nature.
2. Versatile: General-purpose programming language for web development, data analysis, AI, automation, and more.
3. Cross-Platform: Runs on Windows, macOS, Linux, and other operating systems.
4. Large Community: Extensive libraries, frameworks, and resources.
5. Open-Source: Free to use, modify, and distribute.
6. Dynamic Typing: No need to declare variable types before use.
7. Object-Oriented: Supports encapsulation, inheritance, and polymorphism.
8. Extensive Libraries: NumPy, pandas, matplotlib, scikit-learn, Django, Flask, and more.

QUS - Describe the role of predefined keywords in Python and provide examples of how they are used in a program?

ANS - Predefined Keywords in Python

Predefined keywords, also known as reserved words, are words that have special meanings in Python. They are used to define the structure and logic of a program.

Role of Predefined Keywords:

1. Define program flow (e.g., if, else, for)
2. Declare variables and data types (e.g., class, def)
3. Handle exceptions (e.g., try, except)
4. Control access (e.g., public, private)
5. Define functions and modules (e.g., import, from)

Examples of Predefined Keywords:

[ ]:
```
# PROGRAM FLOW
# IF : CONDITIONAL STSTEMENT
```

[ ]:
```
x = 5
if x < 10:
  print("x is greater than 10")
```

```
x is greater than 10
```

[ ]:
```
# FOR : LOOPING STSTEMENT
```

[ ]:
```
fruits = ("apple","banana","cherry")
for fruit in fruits:
  print(fruit)
```

```
apple
banana
cherry
```

[ ]:
```
# WHILE : LOOPING STSTEMENT
```

[ ]:
```
x = 0
while x < 5:
  print(x)
  x +=1
```

```
0
1
2
3
4
```

[ ]:
```

```

QUS - Compare and contrast mutable and immutable objects in Python with examples? ANS - Mutable vs Immutable Objects in Python

In Python, objects can be either mutable or immutable.

Mutable Objects:

- Can be modified after creation
- Changes affect the original object

Examples:

1. Lists (list)
2. Dictionaries (dict)
3. Sets (set)
4. User-defined classes (class)

Immutable Objects:

- Cannot be modified after creation

- Changes create a new object

Examples:

1. Integers (int)
2. Floats (float)
3. Strings (str)
4. Tuples (tuple)
5. Boolean (bool)
6. NoneType (None)

Comparison:

	Mutable	Immutable
Modifiable	Yes	No
Changes	Affect original	Create new object
Examples	Lists, Dictionaries	Integers, Strings

QUS - Discuss the different types of operators in Python and provide examples of how they are used ? ANS - Types of Operators in Python

Python has various types of operators that perform specific operations on variables and values. Here are the main categories:

1. Arithmetic Operators

Perform mathematical operations.

Operator	Description	Example
+	Addition	a = 2 + 3
-	Subtraction	a = 5 - 2
	Multiplication	a = 4 * 5
/	Division	a = 10 / 2
	Exponentiation	a = 2 ** 3
%	Modulus	a = 10 % 3
//	Floor Division	a = 10 // 3

2. Comparison Operators

Compare values.

Operator	Description	Example
==	Equal	a = 5; b = 5; a == b
!=	Not Equal	a = 5; b = 3; a != b
>	Greater Than	a = 5; b = 3; a > b
<	Less Than	a = 3; b = 5; a < b
>=	Greater Than or Equal	a = 5; b = 5; a >= b
<=	Less Than or Equal	a = 3; b = 5; a <= b

3. Logical Operators

Perform logical operations.

Operator	Description	Example
and	Logical And	a = True; b = True; a and b
or	Logical Or	a = True; b = False; a or b
not	Logical Not	a = True; not a

4. Assignment Operators

Assign values.

Operator	Description	Example
=	Assign	a = 5
+=	Add and Assign	a = 5; a += 3
-=	Subtract and Assign	a = 5; a -= 2
=	Multiply and Assign	a = 5; a *= 3
/=	Divide and Assign	a = 10; a /= 2

5. Bitwise Operators

Perform binary operations.

Operator	Description	Example
&	Bitwise And	a = 5; b = 3; a & b
		Bitwise Or
^	Bitwise Xor	a = 5; b = 3; a ^ b
~	Bitwise Not	a = 5; ~a
<<	Left Shift	a = 5; a << 2
>>	Right Shift	a = 20; a >> 2

6. Membership Operators

Check membership.

Operator	Description	Example
in	Member	a = [1, 2, 3]; 2 in a
not in	Not Member	a = [1, 2, 3]; 4 not in a

7. Identity Operators

Check identity.

Operator	Description	Example
is	Same Object	a = [1, 2, 3]; b = a; a is b

Operator	Description	Example
is not	Not Same Object	a = [1, 2, 3]; b = [1, 2, 3]; a is not b

These operators are used extensively in Python programming.

QUS - Explain the concept of type casting in Python with examples? ANSZ - Type Casting in Python

Type casting is the process of converting a value from one data type to another. Python supports two types of type casting:

1. Implicit Type Casting

Python automatically converts the data type without explicit conversion.

2. Explicit Type Casting

Python requires explicit conversion using built-in functions.

Implicit Type Casting Examples:

1. Integer to Float:

a = 5 b = 2.0 result = a + b # result becomes float (7.0)

1. String to Integer (in arithmetic operations):

a = “5” b = 2 result = int(a) + b # result becomes integer (7)

Explicit Type Casting Examples:

1. int() - Convert to Integer:

a = 3.14 b = int(a) # b becomes integer (3)

1. float() - Convert to Float:

a = 5 b = float(a) # b becomes float (5.0)

1. str() - Convert to String:

a = 5 b = str(a) # b becomes string (“5”)

1. tuple() - Convert to Tuple:

a = [1, 2, 3] b = tuple(a) # b becomes tuple (1, 2, 3)

1. list() - Convert to List:

a = (1, 2, 3) b = list(a) # b becomes list [1, 2, 3]

1. dict() - Convert to Dictionary:

a = [(“name”, “John”), (“age”, 30)] b = dict(a) # b becomes dictionary {“name”: “John”, “age”: 30}

1. set() - Convert to Set:

a = [1, 2, 2, 3] b = set(a) # b becomes set {1, 2, 3}

1. bool() - Convert to Boolean:

a = 5 b = bool(a) # b becomes boolean True

QUS - How do conditional statements work in Python? Illustrate with examples? ANS - Conditional Statements in Python

Conditional statements execute specific code blocks based on conditions or decisions.

Types of Conditional Statements:

1. if statement
2. if-else statement
3. if-elif-else statement
4. nested if statement
5. if statement

Executes code if the condition is true.

x = 5 if x > 10: print("x is greater than 10")

2. if-else statement

Executes one code block if the condition is true, another if false.

x = 5 if x > 10: print("x is greater than 10") else: print("x is less than or equal to 10")

3. if-elif-else statement

Executes different code blocks based on multiple conditions.

x = 5 if x > 10: print("x is greater than 10") elif x == 5: print("x is equal to 5") else: print("x is less than 5")

4. nested if statement

Executes code inside another if statement.

x = 5 y = 3 if x > 10: if y > 2: print("x is greater than 10 and y is greater than 2")

Conditional Operators:

1. == (equal)
2. != (not equal)
3. (greater than)
4. < (less than)
5. = (greater than or equal)
6. <= (less than or equal)
7. in (membership)
8. not in (non-membership)

Logical Operators:

1. and (logical and)
2. or (logical or)
3. not (logical not)

Example with Logical Operators:

x = 5 y = 3 if x > 10 and y > 2: print("x is greater than 10 and y is greater than 2")

QUS - Describe the different types of loops in Python and their use cases with examples ? ANS - Loops in Python

Loops are control structures that execute a block of code repeatedly.

Types of Loops:

1. For Loop: Iterates over a sequence (list, tuple, string, etc.).
2. While Loop: Continues execution while a condition is true.
3. Nested Loop: Loop inside another loop.
4. Infinite Loop: Loop with no termination condition.
5. For Loop

Iterates over a sequence.

fruits = ['apple', 'banana', 'cherry'] for fruit in fruits: print(fruit)

Use cases:

- Iterate over collections (lists, tuples, dictionaries).
- Execute code for each item in a sequence.

2. While Loop

Continues execution while a condition is true.

x = 0 while x < 5: print(x) x += 1

Use cases:

- Repeat code while a condition is met.
- Implement algorithms with unknown iteration counts.

3. Nested Loop

Loop inside another loop.

colors = ['red', 'green', 'blue'] shapes = ['circle', 'square', 'triangle'] for color in colors: for shape in shapes: print(f"{color} {shape}")

Use cases:

- Iterate over multiple sequences.
- Generate combinations of values.

4. Infinite Loop

Loop with no termination condition.

while True: print(“Hello, World!”)

Use cases:

- Implement event-driven programming.
- Create games or simulations.

Loop Control Statements:

1. break: Exit the loop immediately.
2. continue: Skip to the next iteration.
3. pass: Do nothing (placeholder).

Examples:

1 Break

for i in range(5): if i == 3: break print(i)

2 Continue

for i in range(5): if i == 3: continue print(i)

3 Pass

for i in range(5): if i == 3: pass print(i)

[ ]: