Python Assignment

Assignment Solution

Q1.Explain the key features of Python that make it a popular choice for programming.

Ans. <u>Key Features of Python</u> -:

1. **<u>Free and Open Source-:</u>** Python language is freely available at the official website and you can download it . .It is open-source, this means that source code is also available to the public .
2. **<u>Easy to code -:</u>** Python is very easy to learn the language as compared to other languages like C,C++,Java etc. It is also a developer - friendly languages.
3. **<u>Object-Oriented Language-:</u>** Python supports object-oriented languages and concepts of classes , object encapsulation, etc.
4. **<u>GUI Programming Support -:</u>** PyQt5 is the most popular option for creating graphical apps with Python.
5. **<u>High - Level Language-:</u>** When we write programs in Python, we do not need to remember the system architecture , nor do we need to manage the memory.
6. **<u>Easy to Debug-:</u>** Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces.
7. **<u>Python is a Portable Language-:</u>** If we have Python code for Windows and if we want to run this code on other platforms.

Q2. Describe the role of predefined keywords in python and provide examples of how they are used in a program.

Ans. Python Keywords are some predefined and reserved words in Python that have special meanings . Keywords are used to define the syntax of the coding.The Keywords cannot be used as variable , identifier, function . All the keywords in Python are written in lower case except True and False. There are keywords in Python**.** .

Program -:

```
a=5
if  a>20  :
    print("Greater than 20")
else :
    print("less than 20")
```

Q3. Compare and contrast mutable and immutable objects in python with examples.

Ans. **Mutable : -**

1. These are data types those value can be change after creation
2. Retain the same memory location even after the content modified
3. It Is memory efficient ,as no new object can be created for frequents Change
4. Not inherently thread-safe
5. Ex. list,dictionaries,Set

```
list=["Apple","ram","mango",12,34.4,True]
print(list)
list[0]="Orange"
print(list)
```

Immutable : -

1. These are data types those value can not be change after creation
2. Any modify result in a new object and new memory location
3. It might be faster in some scenario as there's no need track change
4. They are inherently thread -safe due to their unchangeable nature
5. Ex. String ,Tuples

```
str="I am vaishnavi"
print(str)
str[5:14]="Kartik"

TypeError: 'str' object does not support item assignment
```

Q4. Discuss the different types of operators in python and provide examples of how they are used.

Ans. In Python, operators are **special symbols or keywords t**hat carry out operations on values and python variables. They serve as a basis for expressions, which are used to modify data and execute computation

Types of Python Operators

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators

- Identity Operators

1.Arithmetic Operator

- Mathematical operations including addition, subtraction, multiplication, and division are commonly carried out using Python arithmetic operators.
- They are compatible with integers, variables, and expressions.
- In addition to the standard arithmetic operators, there are operators for modulus, exponentiation, and floor division.

Example

```
a=int(input("enter the first number : "))

b=int(input("enter the second number : "))

print(f"{a} + {b} = {a+b}")

print(f"{a} - {b} = {a-b}")

print(f"{a} * {b} = {a*b} ")

print(f" {a} / {b} = {a/b} ")

print(f"{a} % {b} = {a%b}")

print(f"{a} ** {b} = {a**b}")

print(f"{a} // {b} = {a//b}")
```

2.Comparison (Relational) Operators ➖:

- To compare two values, Python comparison operators are needed.
- Based on the comparison, they produce a Boolean value (True or False).

Example

```
 a=int(input("enter value of a :"))

b=int(input("enter value of b :"))

if(a==b):

    print(f"{a} is  equal to {b} ")

elif(a>b):

    print(f"{a} is greater than {b}")

elif(a>=b):
    print(f"{a} is greater than equal to {b}")
elif(a<b):
    print(f"{a} ia less than {b} ")
elif(a<=b):
    print(f"{a} is less than equal to {b} ")
else:
    print(f"{a} is not equal to {b} ")
```

3. <u>Assignment Operators :-</u>

- Python assignment operators are used to assign values to variables in Python.
- The single equal symbol (=) is the most fundamental assignment operator.
- It assigns the value on the operator's right side to the variable on the operator's left side.

  Example

```
a=int(input("enter the first number :"))
b=int(input("enter the second number : "))
a+=b
print(a)
a-=4
print(a)
a*=b
print(a)
a/=b
print(a)
```

<u>4.Logical Operators :-</u>

- Python logical operators are used to compose Boolean expressions and evaluate their truth values.
- They are required for the creation of conditional statements as well as for managing the flow of execution in programs.
- Python has three basic logical operators: AND, OR, and NOT.

  Example

and Logical AND	**If both of the operands are true then the condition becomes true.**	**(a and b) is true.**
or Logical OR	**If any of the two operands is non-zero then the condition becomes true.**	**(a or b) is true.**
not Logical NOT	**Used to reverse the logical state of its operand**	**Not(a and b) is false.**

5.<u>Bitwise Operators</u>:

- Python bitwise operators execute operations on individual bits of binary integers.
- They work with integer binary representations, performing logical operations on each bit location.
- Python includes various bitwise operators, such as AND (&), OR (|), NOT (), XOR (), left shift (), and right shift (>>).

&	**Binary AND**	**Sets each bit to 1 if both bits are 1**
|	**Binary OR**	**Sets each bit to 1 if one of the two bits is 1**
^	**Binary XOR**	**Sets each bit to 1 if only one of two bits is 1**
~	**Binary Ones Complement**	**Inverts all the bits**
~	**Binary Ones Complement**	**Inverts all the bits**

<<	**Binary Left Shift**	**Shift left by pushing zeros in from the right and let the leftmost bits fall off**
>>	**Binary Right Shift**	**Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off**

6.Membership Operators-:

- Python membership operators are used to determine whether or not a certain value occurs within a sequence.
- They make it simple to determine the membership of elements in various Python data structures such as lists, tuples, sets, and strings.
- Python has two primary membership operators: the in and not in operators

in	**Evaluates to true if it finds a variable in the specified sequence and false otherwise.**	**x in y, here in results in a 1 if x is a member of sequence y.**
not in	**Evaluates to true if it does not find a variable in the specified sequence and false otherwise.**	**x not in y, here not in results in a 1 if x is not a member of sequence y.**

7.Identity Operators :-

- Python identity operators are used to compare two objects' memory addresses rather than their values.
- If the two objects refer to the same memory address, they evaluate to True; otherwise, they evaluate to False.
- Python includes two identity operators: the is and is not operators.

is	**Evaluates to true if the variables on either side of the operator point to the same object and false otherwise**	**x is y, here are results in 1 if id(x) equals id(y)**
is no t	**Evaluates to false if the variables on either side of the operator point to the same object and true otherwise**	**x is not y, there are no results in 1 if id(x) is not equal to id(y).**

Q5. Explain the concept of type casting in python with examples.
Ans. Type Casting is the method to convert the Python variable **datatype** into a certain data type in order to perform the required operation by users.

Two types of Type Casting in Python:

1. Python Implicit Type Conversion
2. Python Explicit Type Conversion

Python Implicit Type Conversion:- Python converts the datatype into another datatype automatically. Users don't have to involve in this process.

Example:-

```
a=(9)

print(a)
```

Python Explicit Type Conversion-: Python needs user involvement to convert the variable data type into the required data type.

Example

```
a=(9)

print(float(a))
```

Q6. How do conditional statements work in python ? illustrate with examples.

Ans. Conditional Statements are statements in Python that provide a choice for the control flow based on a condition. It means that the control flow of the Python program will be decided based on the outcome of the condition.

Types of Conditional Statements in Python

1. If Conditional Statement in Python

2. If else Conditional Statement in Python

3. Nested if..else Conditional Statement in Python

4. If-elif-else Conditional Statement in Python

5. Ternary Expression Conditional Statement in Python

1. If Conditional Statement in Python :- If the simple code of block is to be performed if the condition holds then the if statement is used. Here the condition mentioned holds then the code of the block runs otherwise not.

Syntax : **if(condition) :**

print("statement")

Example :

```
if(10 == 5):
        print("both are equal !")
    print("program terminate")
```

2 . **If else Conditional Statement in Python:-** In a conditional if Statement the additional block of code is merged as an else statement which is performed when if condition is false.

Syntax :

if(condition):

#this block is executed if condition is true

else:

# this block is executed if condition is false

Example :

```
if(10 == 5):
  print("both are equal !")
else:
  print("both are not equal !")
```

3. Nested if..else Conditional Statement in Python :- Nested if..else means an if-else statement inside another if statement.

Syntax :

if(condition) :

if(condition):

#executed this block

else:

#executed this clock

else:

#executed this block

Example :

```
x=5

y=8

if(x!=y):



  if(x<y):

    print("y is greater than x !")

  else:

    print("y is less than x")

else:

  print("not equal !")
```

4. If-elif-else Conditional Statement in Python:- The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final "else" statement will be executed.

Syntax:

```
if(condition):

    # block are executed

elif(condition):

   #block are executed

else:

  #block are executed
```

Example

```
y=50

if(20<y<40):

  print("y is greater than 40")

elif(40<y<100):

  print("y is greater than 100")

else:
  print("y is less than 20")
```

5. Ternary Expression Conditional Statement in Python :- The Python ternary Expression determines if a condition is true or false and then returns the appropriate value in accordance with the result. The ternary Expression is useful in cases where we need to assign a value to a variable based on a simple condition, and we want to keep our code more concise — all in just one line of code.

Example :

```
a,b=40,70

print("both are variable " if a == b else "a is greater
than b " if a>b else "b is greater than a")
```

Q7. Describe the different types of loops in Python and their use cases with examples.

Ans. It allow you to executed a block of code repeatedly.

While Loop in Python :-

a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.

Syntax :

```
while(condition) :
    Statement
```

Example :

```
count = 0
while (count < 3):
    count = count + 1
    print("while loop")
```

For Loop in Python :-

For loops are used for sequential traversal. For example: traversing a list or string or array etc.

Syntax:

For var in sequence :

    statement(s)

Example:

```
a = "vaishnavi"
for i in a:
  print(i)
```