Start coding or generate with AI.

Name: Shivanshu Singh Parihar

Batch : September 2024

Assignment Date: 03 Oct 2024

Assignment : Python Basics Assignment

Assignment -1

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

Ans: Python is a very popular and open source programming language, it also known for its simplicity and readability. python is popular for several reasons:

Easy to learn and use: It is very easy to understand, also we can say it is a beginners friendly.

Simplicity and Readability: It has a clear and simple syntax, making it easy for beginners to learn and for experienced programmers to read and write code efficiently.

Versatile: Python can be used for various applications developments, also includes web development, data analysis, artificial intelligence, and many more fields.

Large Community and Libraries: Python has a very vast community compared to other programming language, which means a lot's of libraries and frameworks are available in this language. which makes it easier to find support and resources.

Dynamic Typing: In the Python, we don't have to declare the type of a variable it is when we create or perform any task, which makes coding quicker and very easier for coders.

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

Ans: Predefined keywords in Python are also known as reserved words that have specific meanings and functions in the language. These keywords form the syntax which dictating how the code is structured and executed. Also we can say that predefined keywords has already assigned instruction for the various task or duirng coding a project. Some Predefined Keywords are if, elif, else, for, while etc. just we take example of if and for, that we used in Loop :

if: It is Used to make conditional statements. for: It is Used for looping through sequences.

Double-click (or enter) to edit

```
num = 10

if num > 5:
    print("num is greater than 5")

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

```
num is greater than 5
0
1
2
3
4
```

```
Start coding or generate with AI.
```

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

Ans : **Mutable Objects:** These are objects that can be changed after they are created or made. It means we can modify their content without creating a new object or removing it.

- It Can be changed after creation.
- Modifications do not create a new object.
- Examples include Lists, Dictionaries, and Sets.
- Modifications happen in place, updating the same object in memory.
- Generally faster for changes since no new object creation is involved.
- Useful for collections that require frequent modifications.
- Not hashable means it cannot be used as keys in dictionaries or elements in sets.

Immutable Objects: These are objects that cannot be changed after they are created or made. Any modification in objects that results in the creation of a new object.

- Cannot be changed after creation; any modification creates a new object.
- Examples include Strings, Tuples, and Frozensets.
- Any change results in the creation of a new object in memory.
- Slower for modifications due to the need for new object creation.
- Often used for fixed collections or data that should not change.
- Hashable, it means can be used as keys in dictionaries or elements in sets.

```
my_Hobbies = ["Learning", "playing", "writing"]
my_Hobbies.append("traveling") #Now my_list is ["Learning", "playing", "writing" ,"travel
```

```
my_Hobbies = ("Learning", "playing", "writing")
print(my_Hobbies)



my_Hobbies[3] = "traveling"  #This would raise an error and it cannot  make any changes b
```

```
print(my_Hobbies)
```

```
('Learning', 'playing', 'writing')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-59-a4b091d155b6> in <cell line: 5>()
      3 
      4 
----> 5 my_Hobbies[3] = "traveling"  #This would raise an error and it cannot  make
any changes because Tuple is Immutable. This example of Immutable Objects.
      6 print(my_Hobbies)

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

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

Ans: In Python, operators are special symbols that perform operations on variables and values. The different types of operators in Python are:

(i). Arithmetic Operators: These operators perform basic mathematical operations.

- : Addition
- : Subtraction
- : Multiplication / : Division etc.

```
a = 10
b = 15
result = a + b
print(result)

result = a - b
print(result)

result = a * b
print(result)

result = a / b
print(result)
```

```
25
-5
150
0.6666666666666666
```

(ii). Comparison Operators: These operators compare two values and its return a Boolean result.

- == : Equal to
- != : Not equal to
- : Greater than

- < : Less than
- = : Greater than or equal to
- <= : Less than or equal to

```
a = 10
b = 15
result = (a == b)
print(result)
result = (a != b)
print(result)
result = (a > b)
print(result)
result = (a < b)
print(result)
result = (a >= b)
print(result)
result = (a <= b)
print(result)
```

```
False
True
False
True
False
True
```

(iii). Logical Operators: These operators are used to combine conditional statements.

- and
- or
- not

```
a = 10
b = 15
result = (a > 0) and (b > 0)
print(result)
```

```
True
```

```
result = (a > 0) or (b < 0)
print(result)
```

```
True
```

```
result = not(a > b)
print(result)
```

```
True
```

(iv). Not Operators : In Python, the not operator is a logical operator that negates a boolean value. It returns True if the value is False, and vice versa.

- Not

```
x = 10

if not x < 5:
    print("x is not less than 5")
```

```
x is not less than 5
```

(v). Assignment Operators: These operators assign values to variables.

- = : Assigns a value
- += : Adds and assigns
- -= : Subtracts and assigns
- *= : Multiplies and assigns
- /= : Divides and assigns

```
x = 10
print(x)
```

```
10
```

```
x += 5
print(x)
```

```
15
```

```
 x -= 3
 print(x)
```

```
12
```

```
x *= 2
print(x)
```

```
24
```

(vi). Membership Operators: These operators check for membership in sequences like lists, tuples, or strings.

- in
- not in

```
result = 4 not in [1, 2, 3]
print(result)
```

```
True
```

```
result = 2 in [1, 2, 3]
print(result)
```

```
True
```

(vii). Identity Operators: These operators are used to compare the memory locations of two objects.

- Is (is)
- is not

```
a = [1, 2, 3]
b = a
result = (a is b)
print(result)
```

```
True
```

```
c = a[:]
result = (a is not c)
print(result)
```

```
True
```

Double-click (or enter) to edit

(viii). bitwise: They are used primarily for low-level programming tasks, like manipulating binary data, flags, and optimizing certain algorithms.

- AND(&)
- OR (|)
- XOR(^)
- NOT(~)
- Right Shift (>>)
- Left Shift (<<)

```
a = 10  # In binary: 1010
b = 4   # In binary: 0100
result = a & b  # Result: 0 (In binary: 0000)
print(result)
```

```
0
```

```
result = a | b  # Result: 14 (In binary: 1110)
print(result)
```

```
14
```

```
result = a ^ b  # result is 6 (binary: 0101 ^ 0011 = 0110)
print(result)
```

```
14
```

```
result = ~a
print(result)
```

```
-11
```

```
result = a >> 1  # result is 2 (binary: 0101 >> 1 = 0010)
print(result)
```

```
5
```

```
result = a << 1  # result is 10 (binary: 0101 << 1 = 1010)
print(result)
```

```
20
```

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

Ans: Type casting in Python refers to converting a variable from one data type to another. This can be useful when you want to perform operations that require specific data types or when you need to manipulate data in a certain way. Python provides several built-in functions to perform type casting. Type casting is a fundamental concept in Python that allows for flexibility in data manipulation. By understanding how to convert between different types, you can ensure that your operations are valid and produce the desired results.

- int(): Converts a value to an integer.
- float(): Converts a value to a float.
- str(): Converts a value to a string.
- list(): Converts a value to a list.
- tuple(): Converts a value to a tuple.
- set(): Converts a value to a set.
- dict(): Converts a sequence of key-value pairs to a dictionary

```
# From int to float
a = 5
b = float(a)
```

```
print(b)
```

```
5.0
```

```
# From string to float
s = "3.14"
f = float(s)
print(f)
```

```
3.14
```

```
a = 5
b = 2.0
result = a + b
print(result)
```

```
7.0
```

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

Ans: Conditional statements in Python allow you to execute certain blocks of code based on specific conditions. In this condition can be one or more. The most commonly used conditional statements are if, elif, and else. Where we can say:

- if : This checks a condition and executes a block of code if the condition is true.
- elif: Short for "else if," this allows you to check multiple conditions.
- else: This executes a block of code if none of the preceding conditions are true.

```
temperature = 75
humidity = 30

if temperature > 70 and humidity < 40:
    print("It's a warm and dry day.")
elif temperature > 70 and humidity >= 40:
    print("It's a warm and humid day.")
else:
    print("It's a cool day.")
```

```
It's a warm and dry day.
```

```
num = 15

if num > 0:
    print("Positive number")
    if num % 2 == 0:
        print("Even number")
    else:
        print("Odd number")
else:
    print("Negative number or zero")
```

```
Positive number
Odd number
```

```
score = 85

if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: D")
```

```
Grade: B
```

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

Ans: In Python, loops are used to execute a block of code multiple times. The two primary types of loops are for loops and while loops. For loops are typically used when you know the number of iterations beforehand or when iterating over a sequence. While loops are used when the number of iterations is not known beforehand and depend on a condition being true.

```
#print a right traiangle with *
row =1
while row <=4:
    col=1
    while col <= row:
        print("*", end = "")
        col = col+1
    print()
    row=row+1
```

```
*
**
***
****
```

```
a='pwskills'
for i in a:
    print(i)
```

```
p
w
s
k
i
l
l
s
```

```
n=7
i=1
while i <0:
    print(i)
    i=i+1
else:
    print("This will be executed when whil loop")
```

```
This will be executed when whil loop
```

```
n=7
i=1
while i <n:
    print(i)
    i=i+1
    if i==3:
        continue
    print(1)
```

```
1
1
2
3
1
4
1
5
1
6
1
```

```
count =5
while count >0:
    print(count)
    count = count -1
```

```
5
4
3
2
1
```

```
l=[1,2,"Shivanshu", 'Data_analytics']
for i in l:
    print(i)
```

```
1
2
Shivanshu
Data_analytics
```