NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

MACHINE LEARNING (LAB)

Lab Report: 01

Submitted to
Ms. Qurat Raja

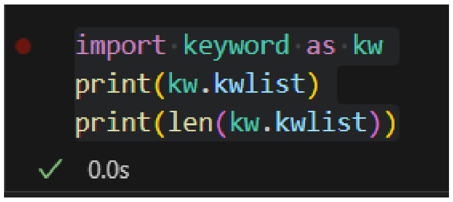
Submitted By
Junaid Asif
(BSAI-144)

Submission Date: October 01, 2024

Importing and Listing Python Keywords:

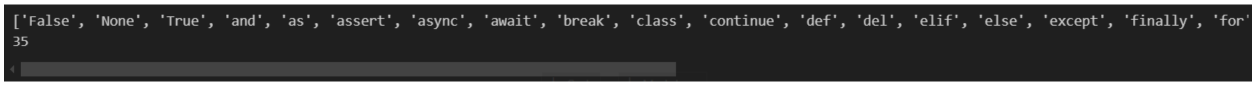
In this section, Python's keyword module is imported, and the list of all reserved keywords is displayed using 'kw.kwlist'.

Code Snippet:



```
import keyword as kw
print(kw.kwlist)
print(len(kw.kwlist))
✓ 0.0s
```

Output:

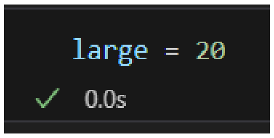


```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for'
35
```

Valid Variable Declaration:

Here, a variable 'large' is declared and assigned a value of 20.

Code Snippet:

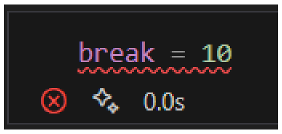


```
large = 20
✓ 0.0s
```

Invalid Use of Python Keywords:

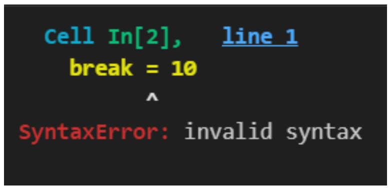
The attempt to declare a variable named 'break' results in a syntax error.

Code Snippet:



```
break = 10
0.0s
```

Output:

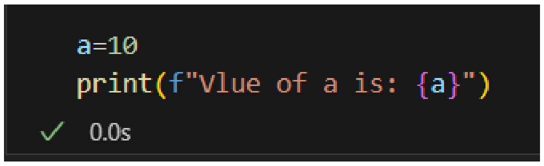


```
Cell In[2],   line 1
  break = 10
        ^
SyntaxError: invalid syntax
```

- **Printing a Variable:**

  This section corrects the keyword mistake and prints the value of the variable ‘a’.

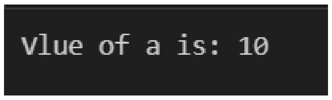
  Code Snippet:



```
a=10
print(f"Vlue of a is: {a}")
```

✓ 0.0s

  Output:

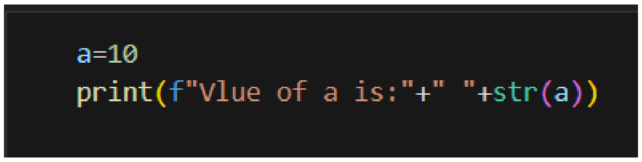


```
Vlue of a is: 10
```

- **String Concatenation in Print Statement:**

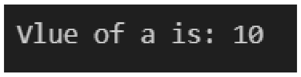
  This section converts an integer ‘a’ into a string to concatenate it with another string during printing.

  Code Snippet:



```
a=10
print(f"Vlue of a is:"+" "+str(a))
```

  Output:

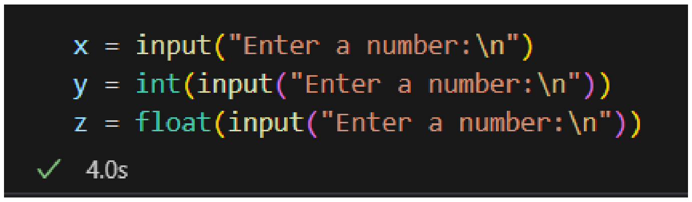


```
Vlue of a is: 10
```

- **Taking Input and Data Type Conversion:**

  Here, user input is taken in three forms: string, integer, and float. The ‘input()’ function is used, and explicit type conversion is applied to ‘y’ and ‘z’.

  Code Snippet:



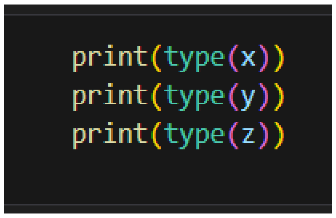
```
x = input("Enter a number:\n")
y = int(input("Enter a number:\n"))
z = float(input("Enter a number:\n"))
```

✓ 4.0s

- ## Checking Variable Types:

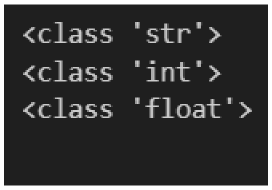
The types of the variables ‘x’, ‘y’, and ‘z’ are printed to confirm their data types.

Code Snippet:



```
print(type(x))
print(type(y))
print(type(z))
```

Output:

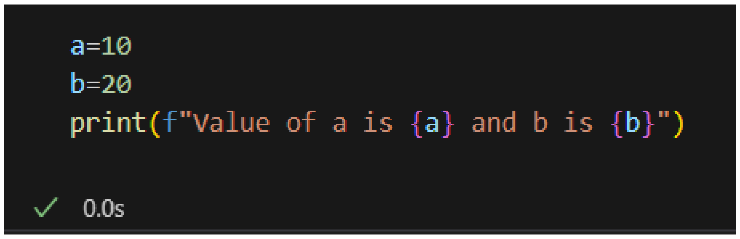


```
<class 'str'>
<class 'int'>
<class 'float'>
```

- ## Using f-Strings for Print Formatting:

In this part, ‘a’ and ‘b’ are printed using f-string formatting. The ‘f-string’ allows for easier and more readable insertion of variables within a string, producing the output.

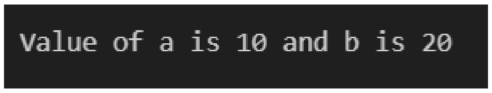
Code Snippet:



```
a=10
b=20
print(f"Value of a is {a} and b is {b}")
```

✓ 0.0s

Output:

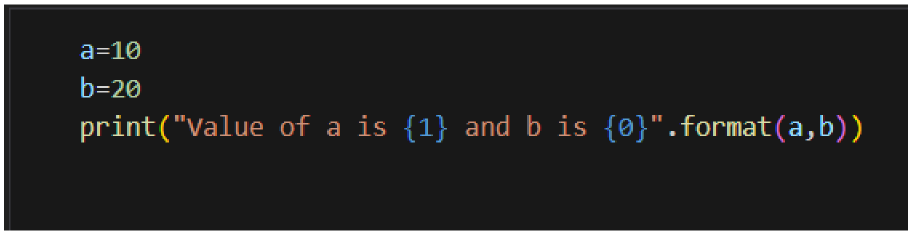


```
Value of a is 10 and b is 20
```

- ## Using .format() for String Formatting:

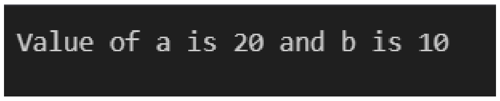
This section demonstrates how to format strings using the .format() method by inserting values in the specified placeholders {}. The numbers {1} and {0} reference the positions of a and b in the .format(a, b) argument list.

Code Snippet:



```
a=10
b=20
print("Value of a is {1} and b is {0}".format(a,b))
```

Output:

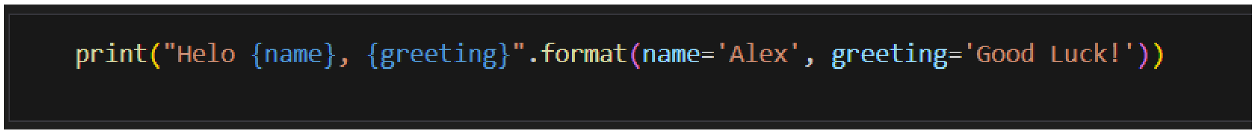


```
Value of a is 20 and b is 10
```

- ## Named Placeholders in .format():

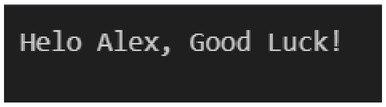
Named placeholders like {name} and {greeting} are replaced by the corresponding keyword arguments in the .format() function.

Code Snippet:



```
print("Helo {name}, {greeting}".format(name='Alex', greeting='Good Luck!'))
```

Output:

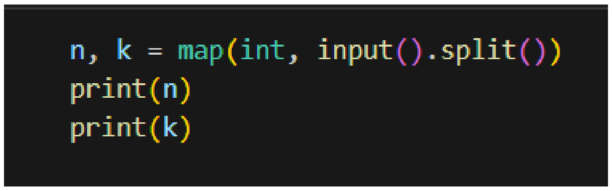


```
Helo Alex, Good Luck!
```

- ## Using map() with split() to Parse Input:

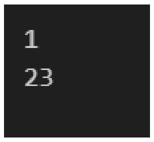
input() takes input from the user. split() breaks the input by spaces (default separator) into a list of strings. map(int, ...) converts each element in the list to an integer. In the example, the user input is two space-separated values like 1 23.

Code Snippet:



```
n, k = map(int, input().split())
print(n)
print(k)
```

Output:

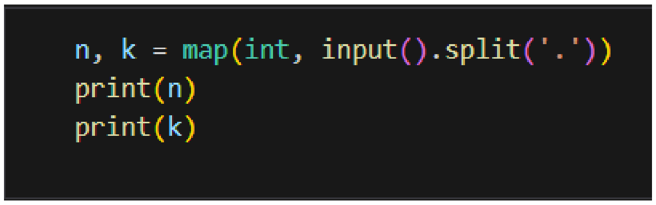


```
1
23
```

- ## Using map() with a Custom Split Character:

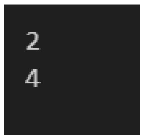
Instead of splitting the input by spaces, the input is split by periods (.). For example, if the input is 2.4.

Code Snippet:



```
n, k = map(int, input().split('.'))
print(n)
print(k)
```

Output:

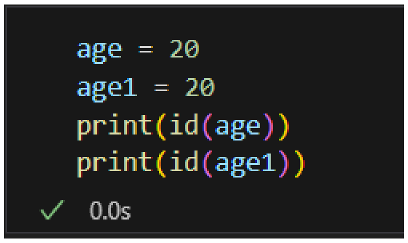


```
2
4
```

- ## Object Identity with id():

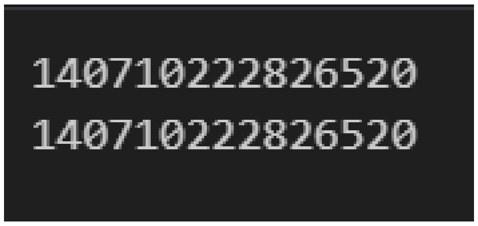
The id() function returns the identity (or memory address) of an object. Here, since both age and age1 hold the same integer value (20), Python optimizes memory usage by pointing both variables to the same memory location. Thus, the output shows identical IDs.

Code Snippet:



```
age = 20
age1 = 20
print(id(age))
print(id(age1))
✓ 0.0s
```

Output:

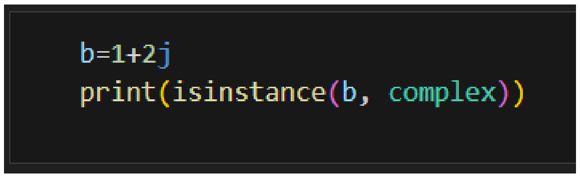


```
140710222826520
140710222826520
```

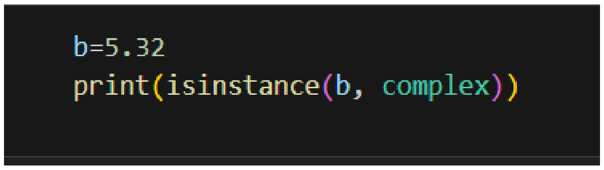
- ## Type Checking with isinstance():

Here, the isinstance() function is used to check the type of a variable. The isinstance() function checks if ‘b’ is of the complex type or not.

Code Snippet:

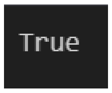


```
b=1+2j
print(isinstance(b, complex))
```

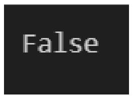


```
b=5.32
print(isinstance(b, complex))
```

Output:



```
True
```

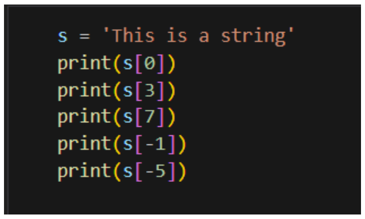


```
False
```

- ## String Indexing:

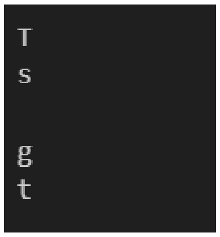
This section demonstrates accessing characters from a string using positive and negative indexing.

Code Snippet:



```
s = 'This is a string'
print(s[0])
print(s[3])
print(s[7])
print(s[-1])
print(s[-5])
```

Output:



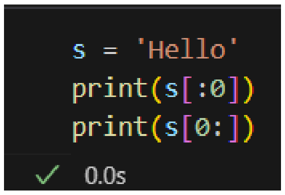
```
T
s

g
t
```

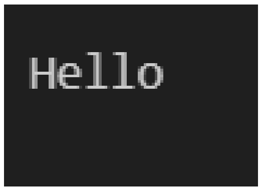
- **String Slicing:**

  Demonstrates how to slice a string and retrieve specific portions of it.

  Code Snippet: **Output:**



```
s = 'Hello'
print(s[:0])
print(s[0:])
```
0.0s

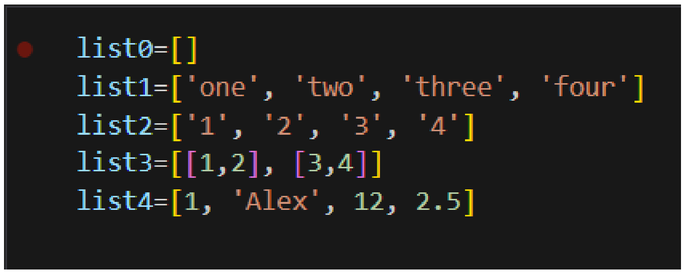


```
Hello
```

- **List Initialization:**

  Shows how to create different types of lists, including empty lists and lists with mixed data types.

  Code Snippet:

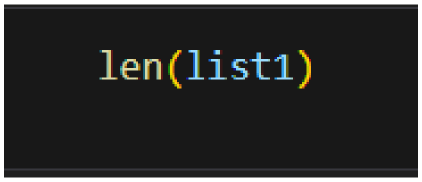


```
list0=[]
list1=['one', 'two', 'three', 'four']
list2=['1', '2', '3', '4']
list3=[[1,2], [3,4]]
list4=[1, 'Alex', 12, 2.5]
```

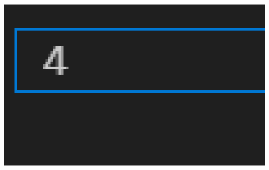
- **Length of a List:**

  Uses the len() function to determine the number of elements in a list.

  Code Snippet: **Output:**



```
len(list1)
```

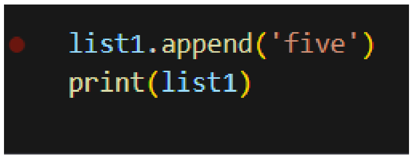


```
4
```

- **Appending an Element to a List:**

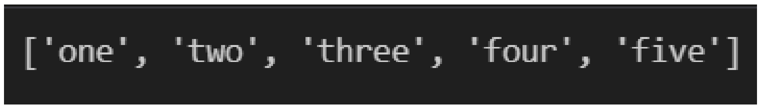
  Adds a new element ('five') to the end of an existing list.

Code Snippet:



```
list1.append('five')
print(list1)
```

Output:

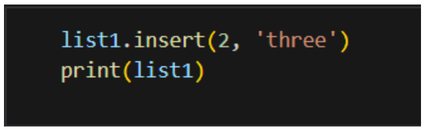


```
['one', 'two', 'three', 'four', 'five']
```

- ## Inserting an Element at a Specific Position:

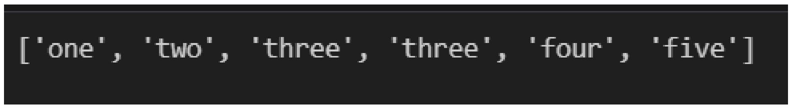
Inserts an element 'three' at the third position of a list, shifting the remaining elements.

Code Snippet:



```
list1.insert(2, 'three')
print(list1)
```

Output:

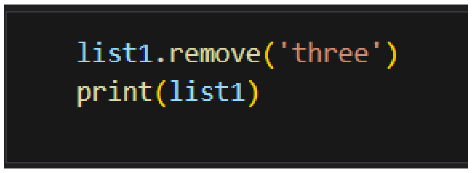


```
['one', 'two', 'three', 'three', 'four', 'five']
```

- ## Removing an Element from a List:

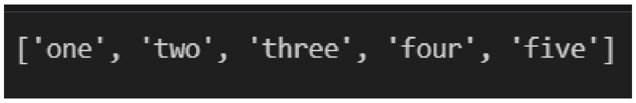
Demonstrates how to remove a specific element 'three' from a list.

Code Snippet:



```
list1.remove('three')
print(list1)
```

Output:

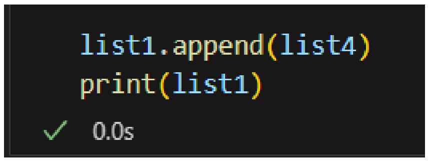


```
['one', 'two', 'three', 'four', 'five']
```

- ## Appending a List to Another List:

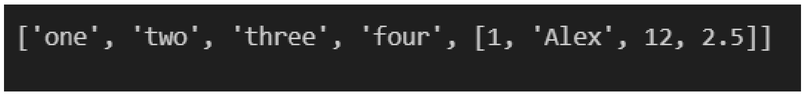
Shows how to append an entire list (list4) as a single element to another list ‘list1’.

Code Snippet:



```
list1.append(list4)
print(list1)
✓ 0.0s
```

Output:

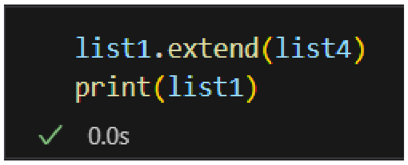


```
['one', 'two', 'three', 'four', [1, 'Alex', 12, 2.5]]
```

- **Extending a List with Another List:**

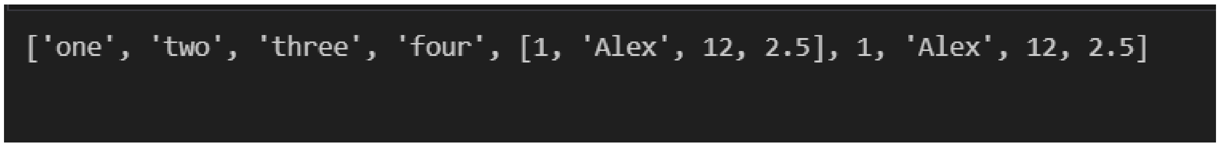
  Illustrates how to extend a list by adding the individual elements of another list ‘list4’ to it.

Code Snippet:



```
list1.extend(list4)
print(list1)
✓ 0.0s
```

Output:



```
['one', 'two', 'three', 'four', [1, 'Alex', 12, 2.5], 1, 'Alex', 12, 2.5]
```