**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **20-05-2020** | **Name:** | **Dhanya Shetty** |
| **Course:** | **TCSion** | **USN:** | **4AL17EC026** |
| **Topic:** | **Develop soft skills for the workplace** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **Dhanya Shetty\_026** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
|  |
| * C:\Users\Hp\Pictures\Camera Roll\20200520_151155.jpg  |  |  | | --- | --- | |  | | |  | | **C:\Users\Hp\Pictures\photos\20200520_151221.jpg** | | |
| |  |  |  | | --- | --- | --- | | **Date: 20-05-2020** |  | **Name: Dhanya Shetty** | | **Course: python operations with datatypes** |  | **USN:4AL17EC026** | | **Topic: Basics Data types** |  | **Semester & Section:6th A** | |

|  |
| --- |
|  |
|  |
|  |
| **AFTERNOON SESSION DETAILS** | |
| **Image of sessions**  **C:\Users\Hp\Pictures\python 1 20may.PNG** | |
| C:\Users\Hp\Pictures\python2 20may.PNGC:\Users\Hp\Pictures\python03may20.PNG  C:\Users\Hp\Pictures\python04may20.PNG  The principal **built-in types** are numerics, sequences, mappings, classes, instances and exceptions. Some collection classes are mutable. The methods that add, subtract, or rearrange their members in place, and don't return a specific item, never return the collection instance itself but None.  Variables can hold values of different data types. Python is a dynamically typed language hence we need not define the type of the variable while declaring it. The interpreter implicitly binds the value with its type. Python enables us to check the type of the variable used in the program. Python provides us the **type()** function which returns the type of the variable passed.  **Python has five standard Data Types:**  A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.   * [Numbers](https://www.javatpoint.com/python-data-types#numbers) * [String](https://www.javatpoint.com/python-data-types#string) * [List](https://www.javatpoint.com/python-data-types#list) * [Tuple](https://www.javatpoint.com/python-data-types#tuple) * [Dictionary](https://www.javatpoint.com/python-data-types#dictionary)   **Numbers**: Number stores numeric values. Python creates Number objects when a number is assigned to a variable.  For eg.  a = 3 , b = 5  #a and b are number objects  **String**: The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.  String handling in python is a straightforward task since there are various inbuilt functions and operators provided.  In the case of string handling, the operator + is used to concatenate two strings as the operation *"hello"+" python"* returns *"hello python"*.  The operator \* is known as repetition operator as the operation "Python " \*2 returns "Python Python ".  The following example illustrates the string handling in python.   1. str1 = 'hello javatpoint' #string str1 2. str2 = ' how are you' #string str2 3. **print** (str1[0:2]) #printing first two character using slice operator 4. **print** (str1[4]) #printing 4th character of the string 5. **print** (str1\*2) #printing the string twice 6. **print** (str1 + str2) #printing the concatenation of str1 and str2   **Output:**  he  o  hello javatpointhello javatpoint  hello javatpoint how are you  **List**: Lists are similar to arrays in C. However; the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (\*) works with the list in the same way as they were working with the strings.  Consider the following example.   1. l  = [1, "hi", "python", 2] 2. **print** (l[3:]); 3. **print** (l[0:2]); 4. **print** (l); 5. **print** (l + l); 6. **print** (l \* 3);   **Output:**  [2]  [1, 'hi']  [1, 'hi', 'python', 2]  [1, 'hi', 'python', 2, 1, 'hi', 'python', 2]  [1, 'hi', 'python', 2, 1, 'hi', 'python', 2, 1, 'hi', 'python', 2]  **Tuple:** A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.  Let's see a simple example of the tuple.   1. t  = ("hi", "python", 2) 2. **print** (t[1:]); 3. **print** (t[0:1]); 4. **print** (t); 5. **print** (t + t); 6. **print** (t \* 3); 7. **print** (type(t)) 8. t[2] = "hi";   **Output:**  ('python', 2)  ('hi',)  ('hi', 'python', 2)  ('hi', 'python', 2, 'hi', 'python', 2)  ('hi', 'python', 2, 'hi', 'python', 2, 'hi', 'python', 2)  <type 'tuple'>  Traceback (most recent call last):  File "main.py", line 8, in <module>  t[2] = "hi";  TypeError: 'tuple' object does not support item assignment  **Dictionary:** Dictionary is an ordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type whereas value is an arbitrary Python object.The items in the dictionary are separated with the comma and enclosed in the curly braces {}.  Consider the following example.   1. d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}; 2. **print**("1st name is "+d[1]); 3. **print**("2nd name is "+ d[4]); 4. **print** (d); 5. **print** (d.keys()); 6. **print** (d.values());   **Output:**  1st name is Jimmy  2nd name is mike  {1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}  [1, 2, 3, 4]  ['Jimmy', 'Alex', 'john', 'mike']  **Summary:**  **Python** is a high-level, interpreted, interactive and object-oriented scripting language. **Python** is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.  Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.  You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.  Python supports Object-Oriented style or technique of programming that encapsulates code within objects.  Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games. | |