**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **18/5/20** | **Name:** | **Abhishek Vasudev Mahendrakar** |
| **Course:** | **TCS-ION** | **USN:** | **4AL17EC003** |
| **Topic:** | **Day 1- Communicate to Impress** | **Semester & Section:** | **6th-‘A’** |
| **Github Repository:** | **--------------** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report – Report can be typed or hand written for up to two pages.**   * **Communication is an art of sending and receiving information.** * **Communication has mainly 3 parts- Sender, Channel, Receiver.** * **Communication can be classified into verbal and non-verbal.** * **Non-verbal communication contains eye-contact, paralanguage, facial expressions, gestures, postures and appearance.** * **Verbal communication consists of face-to-face, written and telephonic communication.** * **Effective communication happens if there is balance between Verbal and Non-verbal communication.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **18/5/2020** | **Name:** | **Abhishek Vasudev Mahendrakar** |
| **Course:** | **UDEMY-The Python Mega Course: Build 10 real world applications** | **USN:** | **4AL17EC003** |
| **Topic:** | **Section-1, 2, 3, 4, 5** | **Semester & Section:** | **6th-‘A’** |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| Report – Report can be typed or hand written for up to two pages.  Python**is not the program you download from**python.org and that you install on your computer. That's just a desktop program written in C language. That program is designed to interpret Python.  Python itself is a language, or to put it differently, a set of rules defined to enable the communication between us and computers. So, when we say print(1 + 2) that code is a rule the Python developers have agreed upon with computers to ask computers to print out the sum of 1 and 2.  These rules reside as Python documentation and are officially laid out in the Python documentation at https://docs.python.org/3/  So, the program we install is called a Python interpreter. When we supply that program with Python code, the program first takes a look to see if the syntax is correct (parsing phase). Then, it translates the Python code to byte-code (compilation phase), and lastly, it interprets that bytecode (Interpreting phase) and it returns the output. Byte-code is another language that is very hard to read from humans but runs very fast on computers. That's why we write the code in a human-friendly language such as Python but execute it as byte-code.   * Python 3 and the Visual Studio Code IDE is used in the videos, but you can use any IDE. * The Python interactive shell (shown with >>>) is a quick way to execute Python code to see how it works. * Python programs are written in .py files. * You can make a program that shows the current date and time using these lines of code:  1. import datetime 2. x = datetime.datetime.now() 3. print(x)  * **Integers** are for representing whole numbers:  1. rank = 10 2. eggs = 12 3. people = 3  * **Floats** represent continuous values:  1. temperature = 10.2 2. rainfall = 5.98 3. elevation = 1031.88  * **Strings** represent any text:  1. message = "Welcome to our online shop!" 2. name = "John" 3. serial = "R001991981SW"  * **Lists** represent arrays of values that may change during the course of the program:  1. members = ["Sim Soony", "Marry Roundknee", "Jack Corridor"] 2. pixel\_values = [252, 251, 251, 253, 250, 248, 247]  * **Dictionaries** represent pairs of keys and values:  1. phone\_numbers = {"John Smith": "+37682929928", "Marry Simpons": "+423998200919"} 2. volcano\_elevations = {"Glacier Peak": 3213.9, "Rainer": 4392.1}  * **Keys** of a dictionary can be extracted with:  1. phone\_numbers.keys()  * **Values** of a dictionary can be extracted with:  1. phone\_numbers.values()  * **Tuples** represent arrays of values that are not to be changed during the course of the program:  1. vowels = ('a', 'e', 'i', 'o', 'u') 2. one\_digits = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)  * To find out what **attributes** a type has:  1. dir(str) 2. dir(list) 3. dir(dict)  * **To find out what Python builtin functions there are:**  1. dir(\_\_builtins\_\_)  * **Documentation** for a Python command can be found with:  1. help(str) 2. help(str.replace) 3. help(dict.values)  * **Lists, strings, and tuples have a positive index system:**  1. ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. 0 1 2 3 4 5 6  * **And a negative index system:**  1. ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. -7 -6 -5 -4 -3 -2 -1  * **In a list, the 2nd, 3rd, and 4th items can be accessed with:**  1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. days[1:4] 3. Output: ['Tue', 'Wed', 'Thu']  * **First three items of a list**:  1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. days[:3] 3. Output:['Mon', 'Tue', 'Wed']  * **Last three items of a list**:  1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. days[-3:] 3. Output: ['Fri', 'Sat', 'Sun']  * **Everything but the last**:  1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. days[:-1] 3. Output: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']  * **Everything but the last two**:  1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] 2. days[:-2] 3. Output: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']  * **A single in a dictionary can be accessed using its key:**  1. phone\_numbers = {"John Smith":"+37682929928","Marry Simpons":"+423998200919"} 2. phone\_numbers["Marry Simpsons"] 3. Output: '+423998200919'   **Using "and" and "or" in a Conditional**   1. x = 1 3. if x == 1: 4. print("Yes") 5. else: 6. print("No")   **Check if two conditions are met at the same time using an and operator:**   1. x = 1 2. y = 1 4. if x == 1 and y==1: 5. print("Yes") 6. else: 7. print("No")   That will return Yes since x == 1 and y ==1 are both True.  **Check if one of two conditions are met using an or operator:**   1. x = 1 2. y = 1 4. if x == 1 or y==2: 5. print("Yes") 6. else: 7. print("No")   That will return Yes since at least one of the conditions is True. In this case x == 1 is True.   * **Define a function:**  1. def cube\_volume(a): 2. return a \* a \* a  * **Write a conditional block:**  1. message = "hello there" 3. if "hello" in message: 4. print("hi") 5. else: 6. print("I don't understand")  * **Write a conditional block of multiple conditions:**  1. message = "hello there" 3. if "hello" in message: 4. print("hi") 5. elif "hi" in message: 6. print("hi") 7. elif "hey" in message: 8. print("hi") 9. else: 10. print("I don't understand")  * **Use the and operator to check if both conditions are True at the same time:**  1. x = 1 2. y = 1 4. if x == 1 and y==1: 5. print("Yes") 6. else: 7. print("No")   Output is Yes since both x and y are 1.   * **Use the or operator to check if at least one condition is True:**  1. x = 1 2. y = 2 4. if x == 1 or y==2: 5. print("Yes") 6. else: 7. print("No")   Output is Yes since x is 1.   * Check if a value is of a certain **type** with:  1. isinstance("abc", str) 2. isinstance([1, 2, 3], list)   or   1. type("abc") == str 2. type([1, 2, 3]) == lst  * **A Python program can get user input via the input function:** * The **input** **function** halts the execution of the program and gets text input from the user**:**  1. name = input("Enter your name: ")  * The input function converts any **input to a string**, but you can convert it back to int or float:  1. experience\_months = input("Enter your experience in months: ") 2. experience\_years = int(experience\_months) / 12  * You can **format strings** with (works both on Python 2 and 3):  1. name = "Sim" 2. experience\_years = 1.5 3. print("Hi %s, you have %s years of experience." % (name, experience\_years))   Output: Hi Sim, you have 1.5 years of experience.   * **You can also format strings with (Python 3 only):**  1. name = "Sim" 2. experience\_years = 1.5 3. print("Hi {}, you have {} years of experience".format(name, experience\_years))   Output: Hi Sim, you have 1.5 years of experience.   * **For loops** are useful for executing a command over a large number of items. * **You can create a for loop like so:**  1. for letter in 'abc': 2. print(letter.upper())   Output:  A B C   * The name after for (e.g. letter) is just a variable name * **You can loop over dictionary keys:**  1. phone\_numbers = {"John Smith":"+37682929928","Marry Simpons":"+423998200919"} 2. for value in phone\_numbers.keys(): 3. print(value)   Output:  John Smith Marry Simpsons   * **You can loop over dictionary values:**  1. phone\_numbers = {"John Smith":"+37682929928","Marry Simpons":"+423998200919"} 2. for value in phone\_numbers.values(): 3. print(value)   Output:  +37682929928 +423998200919   * **You can loop over dictionary items:**   1. phone\_numbers = {"John Smith":"+37682929928","Marry Simpons":"+423998200919"}   2. for key, value in phone\_numbers.items():   3. print(key, value)   Output:  ('John Smith', '+37682929928')  ('Marry Simpons', '+423998200919')   * **While loops** will run as long as a condition is true:   1. while datetime.datetime.now() < datetime.datetime(2090, 8, 20, 19, 30, 20):   2. print("It's not yet 19:30:20 of 2090.8.20")   The loop above will print out the string inside print() over and over again until the 20th of August, 2090. | | | |