DAILY ASSESSMENT FORMAT

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 20/05/2020 | Name: | Prajna |
| Course: | TCS ion | USN: | 4AL16EC047 |
| Topic: |  | Semester & Section: | 8 “A” |
| Github Repository: | prajna\_salian |  |  |

|  |
| --- |
| FORENOON SESSION DETAILS |
| Image of session        **Ace corporate interviews:**  What id an interview?   * Interview is a skill which requires practice and presentation. * It provides direct informations about a candidate,his/her skills,background and personality. * Interview process includes four P’s:  1. Preparation 2. Practice 3. Prsentation 4. Participate   Preparation for an effective interview include:   1. A good assessment of yourself 2. Researching the organization 3. Updating your resume 4. Understanding the venue details  * Do’s before the interview  1. Dress appropriately as per the corparate setting 2. Take care of pesonal grooming and cleanliness 3. Reach 10-15 minutes early  * Don’ts before an interview  1. Don’t stay up late at night 2. Don’t feel nervous 3. Don’t forget to be courteous to everybody  * During the interview Do’s  1. Ask for clarification if you don’t understand question 2. Be brief and concise in your response  * During the interview Don’ts  1. Don’t take a seat until you are offered one 2. don’t slouch and fidget   Rules:   * Mens interview attire * Women interview attire   Most frequent questions asked:   * Tell something about yourself * Why shoud we hire you? * What are your strenght? * What are your weaknesses? * What are your achievements? * What is your career objective?   **Write effective emails**   * Describe the structure of a email * Develop an effective subject line and text * Utilize a few opening and closing phrases * State the do’s and dont’s of email writing * Draft an email using the pointers taught in the session * Do’s of email etiquette   1.Use strong subject line  2.Keep your email short  3.Type the correct email id  4.Reply within a reasonable time   * Don’ts of email etiquette  1. don’t use all upper case or all lower case 2. don’t use on word responses 3. don’t call as soon as you send the message |

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 20/05/2020 | Name: | Prajna |
| Course: | PYTHON | USN: | 4AL16EC047 |
| Topic: | List Comprehensions  File Processing | Semester & Section: | 8 “A” |
| AFTERNOON SESSION DETAILS | | | |
| **List Comprehensions:**  Python is popular for permitting you to compose code that is exquisite, simple to compose, and nearly as simple to peruse as plain English. One of the language's most particular highlights is the list comprehensions, which you can use to make amazing usefulness inside a solitary line of code. Be that as it may, numerous engineers battle to completely use the further developed highlights of a list comprehensions in Python.  List comprehensions are a third method of making lists. With this elegant approach, you could rewrite the for loop from the first example in just a single line of code:  Every list comprehension in Python includes three elements:   1. **expression** is the member itself, a call to a method, or any other valid expression that returns a value. In the example above, the expression i \* i is the square of the member value. 2. **member** is the object or value in the list or iterable. In the example above, the member value is i. 3. **iterable** is a list, [set](https://realpython.com/python-sets/), sequence, [generator](https://realpython.com/introduction-to-python-generators/), or any other object that can return its elements one at a time. In the example above, the iterable is range(10).  * A list comprehension is an expression that creates a list by iterating over another container. * A basic list comprehension: * [i\*2 for i in [1, 5, 10]]   Output: [2, 10, 20]   * List comprehension with if condition:   [i\*2 for i in [1, -2, 10] if i>0]  Output: [2, 20]   * List comprehension with an if and else condition:   [i\*2 if i>0 else 0 for i in [1, -2, 10]]  Output: [2, 0, 2  If –Else condition comprehension  obj = ["Even" if i%2==0 else "Odd" for i in range(10)]  print (obj)  **Output -**['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']  ['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd'] Benefits of Using List Comprehensions: List comprehensions are regularly portrayed as being more Pythonic than loops or map(). But instead than indiscriminately tolerating that evaluation, it's justified, despite all the trouble to comprehend the advantages of utilizing a list comprehension in Python when contrasted with the other options. Later on, you'll find out around a couple of situations where the choices are a superior decision.One main benefit of using a list comprehension in Python is that it’s a single tool that you can use in many different situations. In addition to standard [list creation](https://realpython.com/python-lists-tuples/), list comprehensions can also be used for mapping and filtering. You don’t have to use a different approach for each scenario.  This is the main reason why list comprehensions are considered Pythonic, as Python embraces simple, powerful tools that you can use in a wide variety of situations. As an added side benefit, whenever you use a list comprehension in Python, you won’t need to remember the proper order of arguments like you would when you call map().  List comprehensions are also more declarative than loops, which means they're simpler to peruse and comprehend. loops expect you to concentrate on how the list is made. You need to physically make an empty list, loop over the components, and add each of them to the end of the list. With a list comprehension in Python, you can rather concentrate on what you need to go in the list and trust that Python will deal with how the list construction happens.  **File processing:**  File is a contiguous set of bytes used to store data. This data is organized in a specific format and can be anything as simple as a text file or as complicated as a program executable. In the end, these byte files are then translated into binary 1 and 0 for easier processing by the computer.  Files on most modern file systems are composed of three main parts:   1. Header: metadata about the contents of the file (file name, size, type, and so on) 2. Data: contents of the file as written by the creator or editor 3. End of file (EOF): special character that indicates the end of the file   We can **read** an existing file with Python:   1. with open("file.txt")as file: 2. content =file.read()   We can **create** a new file with Python and **write** some text on it:   1. with open("file.txt","w")as file: 2. content =file.write("Sample text")  * We can **append** text to an existing file without overwriting it:  1. with open("file.txt","a")as file: 2. content =file.write("More sample text")  * We can both **append and read** a file with:  Opening and Closing a File in Python When you want to work with a file, the first thing to do is to open it. This is done by invoking the open() built-in function. open() has a single required argument that is the path to the file. open() has a single return, the file object:  file=open('dog\_breeds.txt') Text File Types A text file is the most common file that you’ll encounter. Here are some examples of how these files are opened:  open('abc.txt')  open('abc.txt','r')  open('abc.txt','w')  With these types of files, open() will return a TextIOWrapper file object:  >>>  >>>file=open('dog\_breeds.txt')  >>>type(file)  <class '\_io.TextIOWrapper'>  This is the default file object returned by open(). Buffered Binary File Types A buffered binary file type is used for reading and writing binary files. Here are some examples of how these files are opened:  open('abc.txt','rb')  open('abc.txt','wb')  With these types of files, open() will return either a BufferedReader or BufferedWriter file object:  >>>  >>>file=open('dog\_breeds.txt','rb')  >>>type(file)  <class '\_io.BufferedReader'>  >>>file=open('dog\_breeds.txt','wb')  >>>type(file)  <class '\_io.BufferedWriter'>   1. with open("file.txt","a+")as file: 2. content =file.write("Even more sample text") 3. file.seek(0) 4. content =file.read() | | | |