**NOTES:**

* **Part 1: use Python’s IDLE interactive tool. Write your answer beside each command in this sheet in bold.**
* **Part 2: upload a separate .py file that contains your script/code.**

**PART 1:**

1. **Lists in Python**: Given the following two lists:

L1 = [3, 'word', 4.5, (3,4,5), [(00,'x'), 11, (22,'y')], 'Windsor', 'BCD', 75, 'Toronto']

L2 = ['grass', 'green', 20, 5.00, 'sky', 'tree']

***Work with list indexing, slicing, striding***:

Indicate the results if you type the following at the Python prompt in IDLE interactive mode:

1. L1[2][1]
2. L1[3][0]
3. L1[4][2][1]
4. len(L1)
5. L1[14]
6. L1[-4:-1]
7. L1[2:14]
8. L1[-2:0:2]
9. L2+L1
10. L2\*2
11. L1[4][1] = 4
12. del L2[-3]
13. **TypeError: 'float' object is not subscriptable**
14. **3**
15. **y**
16. **9**
17. **IndexError: list index out of range as there are no elements.**
18. **['Windsor', 'BCD', 75]**
19. **[4.5, (3, 4, 5), [(0, 'x'), 11, (22, 'y')], 'Windsor', 'BCD', 75, 'Toronto']**
20. **[]**
21. **['grass', 'green', 20, 5.0, 'sky', 'tree', 3, 'word', 4.5, (3, 4, 5),**

**[(0, 'x'), 11, (22, 'y')], 'Windsor', 'BCD', 75, 'Toronto']**

1. **['grass', 'green', 20, 'sky', 'tree', [12, 15], 'grass', 'green', 20, 'sky', 'tree', [12, 15]]**
2. **Value is assigned, as Lists are mutable**
3. **Element deleted**

***Work with list methods***:

Type python commands to do the following:

1. append the string 'greetings' to L1 - **(L1.append('greetings'))**
2. remove the last element of L2 - **L2.remove(L2[-1])**
3. insert the item 3.22 at index 2 in L1 - **L1.insert(2,3.22)**
4. add the integers in the list [12, 15] at the end of L2 - **L2.append( [12, 15] )**

***Create list of numbers with conditions***:

Generate a list containing all numbers that are divisible by 2 but not multiple of 5 where the values are square of numbers between 100 and 200 (both included). Can you do that in one line of code? **YES**

**x = [(i) for i in range(pow(100,2)-1,pow(200,2)+1) if(i%2==0 and i%5!=0)]**

1. **Strings in Python:** Given the following two strings:

s1 = "Python is a powerful language!"

s2 = 'Why learn web development in Django?'

***Work with string indexing, slicing, striding, assignment, concatenation***: Indicate the results if you type the following at the Python prompt in IDLE interactive mode. Indicate the type of error if the command is wrong:

1. s1[:6]
2. s2[-1:-4]
3. s1[1] = ' '
4. s2[-1:]
5. s2[0:20:2]
6. s1+" "+s2
7. **Python**
8. **Empty [Index should start from a lesser number]**
9. **Not possible as strings are immutable**
10. **?**
11. **Wylanwbdvl**
12. **Python is a powerful language! Why learn web development in Django?**

***Work with string methods***: Use **str** methods to do the following and indicate the corresponding results:

1. Check if the string s2 ends with the word 'Django?' - **False**
2. Determine leftmost position of ‘pow’ in s1- **12**
3. Return a list of words from s2 - **6**
4. Convert s1 and s2 to all uppercase letters

**PYTHON IS A POWERFUL LANGUAGE!** **WHY LEARN WEB DEVELOPMENT IN DJANGO?**

1. Replace the string 'learn' of s2 with empty string

**‘Why web development in Django?'**

1. Count the number of times ‘p’ occurs in s1- **1**
2. **Dictionary in Python:** Define the following *dicts*:

*#dictionary literals*

d1={"name": "John", "age": 25, (3, 'm'):['a', 'b', 'c'], 5: "Ontario", 20: 96, 12:27}

*#dictionary using sequences*

d2 = dict([("name","Alice"), ('age', 24), ((1,2), ['u', 'v', 'w']), (0, 'blue'), (86, 20)])

*#dictionary using keywords*

d3 = dict(id=123, name='Willis', siblings=['Alex', 'Bob', 'Cindy'])

*#dictionary using zip( ) function*

d4 = dict(zip(("id", "name", "quantity"), (1234, "Disk Drive", 3)))

***Work with dict methods***: Type the following commands at the Python prompt in IDLE interactive mode and indicate the result of each command:

1. d1.keys()
2. d2.values()
3. d3.get('id')
4. d2.get('age')
5. d3.get('age')
6. d3.get('name', 'Tim')
7. d2.items()
8. d3['siblings']
9. d2['siblings']
10. d2.update(d3)
11. d2[0]
12. d1.get((1,2))
13. d2['siblings']**\***
14. d2['name']**\***
15. d1 == d2
16. len(d2)
17. for key in d1.keys():

print(key)

1. for key in d2.keys():

print(d2[key])

**\****After* update.

1. **dict\_keys(['name', 'age', (3, 'm'), 5, 20, 12])**
2. **dict\_values(['Alice', 24, ['u', 'v', 'w'], 'blue', 20])**
3. **123**
4. **24**
5. **None**
6. **Willis**
7. **dict\_items([('name', 'Alice'), ('age', 24), ((1, 2), ['u', 'v', 'w']), (0, 'blue'), (86, 20)])**
8. **['Alex', 'Bob', 'Cindy']**
9. **Error As there is no value name sibblings**
10. **None**
11. **blue**
12. **None**
13. **['Alex', 'Bob', 'Cindy']**
14. **Willis**
15. **False**
16. **7**
17. **name**

**age**

**(3, 'm')**

**5**

**20**

**12**

1. **Willis**

**24**

**['u', 'v', 'w']**

**blue**

**20**

**123**

**['Alex', 'Bob', 'Cindy']**

**4)** **datetimemodule in Python**:

Import the *datetime* module. Write Python commands to do following: (Check <https://docs.python.org/3/library/datetime.html?highlight=datetime#module-datetime>)

1. Display today’s year, month, and day each in separate line.
2. Display today’s datetime in this format Date is dd/mm/yyyy, Time is H:M:S
3. Calculate the duration, represented by the difference in days, between today’s date and the end of year date.
4. **import datetime**

**now = datetime.datetime.now()**

**print (now.year,"\n",now.month,"\n",now.day)**

**Output: 2020**

**5**

**25**

1. **from datetime import datetime,date**

**today = date.today()**

**now = datetime.now()**

**d1 = today.strftime("%d/%m/%Y")**

**current\_time = now.strftime("%H:%M:%S")**

**print("Date is",d1," Time is", current\_time)**

**Output: Date is 25/05/2020 Time is 16:23:51**

1. **a=date.today()**

**b = date(date.today().year, 12, 31)**

**print(b-a)**

**Output: 220 days, 0:00:00**

**PART 2: Write a complete Python script, with comments, to do the following**:

1. Open a text file called “*catalog.txt*”, attached with this lab, for reading. The file contains the items available in a fitness studio, the items categories/classes, and their quantities.
2. Define a list of strings called ***fit\_items***. The list should contain at least 10 strings and each string represent a specific fitness item, e.g., treadmill, lifting bars, weights, etc.
3. Loop over each element in ***fit\_items*** and check if that element matches any of the products in the file.

**Hint**: use the function readline() to read a new line from the file and compare that line with the elements in the list of strings.

1. If there is a match, save the category and the quantity corresponding to that item in some variables.
2. Create a dict ***d1*** with entries ***item:category*** where ***item* (key)** is the item (string) found in *catalog.txt* and ***category*** (**value**) is the category of that item. Add the item and its category to ***d1*** as {***item:category***}. Create another dict ***d2*** with entries ***item:quantity*** and add the item found and its quantity to ***d2***.

**Hint**: use the function update() on ***d1*** and ***d2*** to add the item found and its category and quantity to the dictionaries.

1. Next the program should ask the user to enter a string ***s, representing a fitness item,*** as an input and retrieve the *category* of ***s*** from ***d1*** and the *quantity* from ***d2*** .

* After displaying the category and quantity corresponding to item ***s***, the program asks if the user would like to do another search with (***yes***/***no***) options.
* If the user enters ***yes***, another category and quantity retrieval should be done for another item.
* If the answer is ***no***, the program should exit.

1. If the item’s name entered by the user does not correspond to a valid key, the program should catch an exception. When the exception occurs, display an appropriate error message then prompt the user to input another item’s name.