**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]: **TypeError: 'float' object is not subscriptable**
2. L1[3][0]: **3**
3. L1[4][2][1]: **y**
4. len(L1): **9**
5. L1[14]: **IndexError: list index out of range**
6. L1[-4:-1]: **['Windsor', 'BCD', 75]**
7. L1[2:14] : **[4.5, (3, 4, 5), [(0, 'x'), 11, (22, 'y')], 'Windsor', 'BCD', 75, 'Toronto']**
8. L1[-2:0:2]: **[]**
9. L2+L1: **['grass', 'green', 20, 5.0, 'sky', 'tree', 3, 'word', 4.5, (3, 4, 5), [(0, 'x'), 11, (22, 'y')], 'Windsor', 'BCD', 75, 'Toronto']**
10. L2\*2: **['grass', 'green', 20, 5.0, 'sky', 'tree', 'grass', 'green', 20, 5.0, 'sky', 'tree']**
11. L1[4][1] = 4: **replace 11 at index 1 in the sublist [(00,'x'), 11, (22,'y')] at index 4 => [(00,'x'), 14, (22,'y')]**
12. del L2[-3]: **delete item at index -3 which is 5.00**

***Work with list methods***:

Type python commands to do the following:

1. append the string 'greetings' to L1: **L1+[‘greeting’]**
2. remove the last element of L2: **del 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 += [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?

**L = [x\*\*2 for x in range(100,201) if (x\*x)%2==0 and (x\*x)%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]: **g**
2. s2[-1:-4]: **''**
3. s1[1] = ' ' : **TypeError: 'str' object does not support item assignment**
4. s2[-1:] **: ?**
5. s2[0:20:2]: **'Wylanwbdvl'**
6. s1+" "+s2: **'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?': **s2.endswith('Django?') => True**
2. Determine leftmost position of ‘pow’ in s1: **s1.index('pow') => 12**
3. Return a list of words from s2: **L = s2.split(' ')**
4. Convert s1 and s2 to all uppercase letters: **s1.upper(), s2.upper()**
5. Replace the string 'learn' of s2 with empty string: **s2.replace('learn','')**
6. Count the number of times ‘p’ occurs in s1: **s1.count('p')**
7. **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(): **dict\_keys(['name', 'age', (3, 'm'), 5, 20, 12])**
2. d2.values():**dict\_values(['Alice', 24, ['u', 'v', 'w'], 'blue', 20])**
3. d3.get('id'): **123**
4. d2.get('age'): **24**
5. d3.get('age'): **return empty**
6. d3.get('name', 'Tim'): **Willis**
7. d2.items():**dict\_items([('name', 'Alice'), ('age', 24), ((1, 2), ['u', 'v', 'w']), (0, 'blue'), (86, 20)])**
8. d3['siblings']: **['Alex', 'Bob', 'Cindy']**
9. d2['siblings']: **KeyError: 'sibling'**
10. d2.update(d3): d2 = **{'name': 'Willis', 'age': 24, (1, 2): ['u', 'v', 'w'], 0: 'blue', 86: 20, 'id': 123, 'siblings': ['Alex', 'Bob', 'Cindy']}**
11. d2[0]: **blue**
12. d1.get((1,2)): **return empty**
13. d2['siblings']**\*: ['Alex', 'Bob', 'Cindy']**
14. d2['name']**\*: Willis**
15. d1 == d2: **False**
16. len(d2): **7**
17. for key in d1.keys():

print(key)

name

**age**

**(3, 'm')**

**5**

**20**

**12**

1. for key in d2.keys():

print(d2[key])

Willis

**24**

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

**blue**

**20**

**123**

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

**\****After* update.

**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.

**import datetime**

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

**print(now.year)**

**print(now.month)**

**print(now.day)**

1. Display today’s datetime in this format Date is dd/mm/yyyy, Time is H:M:S

**print (now.strftime("%d/%m/%Y %H:%M:%S"))**

1. Calculate the duration, represented by the difference in days, between today’s date and the end of year date.

**from datetime import date**

**today = date.today()**

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

**print((endOfYear – today).days)**

**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.