Practice Problem Sheet
(Dictionary)

<table>
<tr><th>SL</th><th>Problems</th><th>Difficulty Level</th></tr>
<tr><td>1</td><td>Write a python program to <b>Iterate over</b> a Dictionary.
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>{
"Iron Man": "Tony Stark",
"Captain America": "Steve Rogers",
"Thor": "God of Thunder",
"Hulk": "Bruce Banner"
}</td><td>Iron Man: Tony Stark
Captain America: Steve Rogers
Thor: God of Thunder
Hulk: Bruce Banner</td></tr>
<tr><td>{
"Pikachu": "Electric",
"Charmander": "Fire",
"Squirtle": "Water",
"Bulbasaur": "Grass/Poison"
}</td><td>Pikachu: Electric
Charmander: Fire
Squirtle: Water
Bulbasaur: Grass/Poison</td></tr>
</table></td><td>*</td></tr>
<tr><td>2</td><td>Write a python program to get the <b>maximum and minimum values</b> of a dictionary.
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>{'x': 500, 'y': 5874, 'z': 560}</td><td>Maximum Value: 5874
Minimum Value: 500</td></tr>
</table></td><td>*</td></tr>
<tr><td>3</td><td>Write a python program to write a function <code>check_key(d,key)</code> that takes a dictionary <code>d</code> and a key <code>key</code> as input and returns <code>True</code> if the key exists in the dictionary, otherwise <code>False</code>.
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>marvel = { "Iron Man": "Tony Stark", "Captain America": "Steve Rogers", "Thor": "God of Thunder", "Hulk": "Bruce Banner" }</td><td>Does 'Thor' exist? True
Does 'Loki' exist? False</td></tr>
</table></td><td>*</td></tr>
<tr><td>4</td><td>Suppose you are given two dictionaries.
Now create a new dictionary, <b>merging the two dictionaries</b>, so that the original two dictionaries remain unchanged.</td><td>*</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>dict1 = {"name": "Alice", "age": 25}
dict2 = {"city": "Wonderland", "age": 30}</td><td>merged_dict = {"name": "Alice", "age": 30, "city": "Wonderland"}</td></tr>
</table>
</td><td></td></tr>
<tr><td>**5**</td><td>Write a python program to **combine values** in a list of dictionaries.
<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>dict_list = [
    {"a": 10, "b": 20},
    {"a": 30, "b": 40},
    {"a": 50, "b": 60}
]</td><td>combined_dict = {"a": 90, "b": 120}</td></tr>
</table>
</td><td>*</td></tr>
<tr><td>**6**</td><td>Write a python program to **remove empty items** in the dictionary. Empty items means keys without any values(None).
<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>{'Harry': 'Gryffindor', 'Dobby': **None**, 'Nearly Headless Nick': ’Gryffindor’, 'Snape': 'Slytherin', ‘Cedric’: ‘Hufflepuff, ‘Peeves the Poltergeist’: **None**}</td><td>{'Harry': 'Gryffindor', 'Nearly Headless Nick': ’Gryffindor’, 'Snape': 'Slytherin', ‘Cedric’: ‘Hufflepuff}</td></tr>
</table>
</td><td>*</td></tr>
<tr><td>**7**</td><td>Write a python program to **get the highest value.**
<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>book_shop **=** {'sci fi': 12, 'mystery': 15, 'horror': 8, 'mythology': 10, 'young_adult': 4, 'adventure':14}</td><td>The Highest Selling Book genre is mystery and the number of books sold are 15</td></tr>
</table>
</td><td>*</td></tr>
<tr><td>**8**</td><td>Write a python program to

1) Find out the **eye color** from the info.
2) Find out whether the person has any **favorite color** which starts with **a vowel**. Return the name of the color if yes, otherwise return False.
3) Find out how many areas of interest and what are those.
4) Find whether the person has any **interest in gadgets**.
5) Find whether the person has height greater than **5 feet.**

info = {
'personal_data': {'name': 'Lauren', 'age': 20, 'major': 'Information Science', 'physical_features': {'color': {'eye': 'blue', 'hair': 'brown' }, 'height': "5.8"}},
'other': {'favorite_colors': ['purple', 'green', 'blue', ‘indigo’], 'interested_in': ['social media', 'intellectual property', 'copyright', 'music', 'books'] }
}</td><td>*</td></tr>
</table>

<table>
<tr>
<td><b>9</b></td>
<td>Write a python program that takes a string as an input from the user and counts the frequency of each character using the dictionary. For solving this problem, you need to use each <b>character as a key</b> and its <b>frequency as values</b>. [without using <b>count()</b> function]
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>"Next semester theke porte boshbo"</td><td>{ 'n': 1, 'e': 9, 'x': 1, 't': 5, 's': 4, 'm': 1, 'h': 2, 'k': 1, 'p': 1, 'o': 3, 'r': 2, 'b': 1}</td></tr>
<tr><td>"Python programming is fun"</td><td>{'p': 2, 'y': 1, 't': 1, 'h': 1, 'o': 2, 'n': 3, 'r': 2, 'g': 2, 'a': 1, 'm': 2, 'i': 2, 's': 1, 'f': 1, 'u': 1}</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td><b>10</b></td>
<td>Write a python program to <b>Convert Key-Value list Dictionary to List of Lists</b>
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>friends_dict = {
"Rachel Green": [25, "Fashion Designer"],
"Ross Geller": [27, "Paleontologist"],
"Monica Geller": [26, "Chef"],
"Chandler Bing": [28, "Statistical Analyst"],
"Joey Tribbiani": [29, "Actor"],
"Phoebe Buffay": [26, "Musician"]
}</td><td>friends_list = [
["Rachel Green", 25, "Fashion Designer"],
["Ross Geller", 27, "Paleontologist"],
["Monica Geller", 26, "Chef"],
["Chandler Bing", 28, "Statistical Analyst"],
["Joey Tribbiani", 29, "Actor"],
["Phoebe Buffay", 26, "Musician"]
]</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td><b>11</b></td>
<td>Write a python program to <b>sort</b> a dictionary by
1. Key
2. Value
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>sample_dict = {'apple': 5, 'banana': 2, 'orange': 3, 'grape': 4}</td><td>Sorted by Key: {'apple': 5, 'banana': 2, 'grape': 4, 'orange': 3}
Sorted by Value: {'banana': 2, 'orange': 3, 'grape': 4, 'apple': 5}</td></tr>
</table>
</td>
<td>*</td>
</tr>
<tr>
<td><b>12</b></td>
<td>Write a python program to <b>extract all the values of a particular key</b>.
<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>[{"character": "Wednesday Addams", "trait": "Dark humor"},
{"character": "Enid Sinclair", "trait": "Cheerful"},
{"character": "Thing", "trait": "Helpful"},
{"character": "Xavier Thorpe", "trait": "Artistic"}
]
<b>Key: “character”</b></td><td>['Wednesday Addams', 'Enid Sinclair', 'Thing', 'Xavier Thorpe']</td></tr>
</table>
</td>
<td>*</td>
</tr>
</table>

[{"character": "Eleven", "power": "Telekinesis"}, {"character": "Mike Wheeler", "power": "Loyalty"}, {"character": "Will Byers", "power": "Survival"}, {"character": "Max Mayfield", "power": "Bravery"}] **Key: “power”**	['Telekinesis', 'Loyalty', 'Survival', 'Bravery']

13

Write a Python program to **group the elements** of a given list based on the given function.

**

Input	Output
Original list & function: [7, 23, 3.2, 3.3, 8.4] Function name: floor	Group the elements of the said list based on the given function: {7: [7], 23: [23], 3: [3.2, 3.3], 8: [8.4]}
Original list & function: ['Red', 'Green', 'Black', 'White', 'Pink'] Function name: len:	Group the elements of the said list based on the given function: {3: ['Red'], 5: ['Green', 'Black', 'White'], 4: ['Pink']}

14

Write a Python program to **sort a nested dictionary by value**.

**

Input	Output
nested_dict = {'a': {'key': 3}, 'b': {'key': 1}, 'c': {'key': 2}}	{'b': {'key': 1}, 'c': {'key': 2}, 'a': {'key': 3}}

15

Write a Python program to

1. convert a list of tuples into a dictionary.
2. Sort the dictionary with its value in descending order.

**

Input	Output

Original List of Tuples:
[('Rabbit', 7), ('Elephant', 15), ('Squirrel', 3), ('Cat', 9), ('Dog', 12)]
	Converted Dictionary:
	{'Rabbit': 7, 'Elephant': 15, 'Squirrel': 3, 'Cat': 9, 'Dog': 12}
	
	Sorted Dictionary (descending):
	{'Elephant': 15, 'Dog': 12, 'Cat': 9, 'Rabbit': 7, 'Squirrel': 3}

16

The UIU database contains student data in the form of a dictionary. Each key is a student's name, and the corresponding value is another dictionary with the following information: the student's name, major, CGPA (Cumulative Grade Point Average), and completed credits.

As a software engineer working at UIU, you are tasked with optimizing this dictionary by removing

**

redundancy.

Write a Python program that:

1. Creates a new dictionary with the same student names as keys.

2. For each student, extracts only the "cgpa" and "completed_credits" information as the value.
3. Sorts the new dictionary based on CGPA in descending order.

Input	Output

uiu_database = {
"Jake Peralta": {
"name": "Jake Peralta",
"major": "Detective Work",
"cgpa": 3.8,
"completed_credits": 120
},
"Amy Santiago": {
"name": "Amy Santiago",
"major": "Detective Work",
"cgpa": 4.0,
"completed_credits": 125
},
"Terry Jeffords": {
"name": "Terry Jeffords",
"major": "Administration",
"cgpa": 3.9,
"completed_credits": 110
},
"Rosa Diaz": {
"name": "Rosa Diaz",
"major": "Undercover Operations",
"cgpa": 3.85,
"completed_credits": 100
}
}
	optimized_database = {
	"Amy Santiago": {"cgpa": 4.0, "completed_credits": 125},
	"Terry Jeffords": {"cgpa": 3.9, "completed_credits": 110},
	"Rosa Diaz": {"cgpa": 3.85, "completed_credits": 100},
	"Jake Peralta": {"cgpa": 3.8, "completed_credits": 120}
	}

17

In the upcoming year, 2024, a tech company is planning to update its HR records, which are implemented in Python dictionaries. The company intends to acknowledge and reward senior employees aged over 55 with a bonus of 10,000 BDT as they approach retirement. Additionally, a bonus of 5,000 BDT will be granted to employees whose performance exceeds 95%, and those with performance below 60% will be terminated from the records and will receive no bonus.

Write a Python program that:

1. Identifies employees aged over 55 and grants them a bonus of 10,000 BDT.

2. Grants an additional bonus of 5,000 BDT to employees with performance exceeding 95%.
3. Terminates employees whose performance is below 60% from the records and excludes them from the bonus.
4. Computes and displays:
   - The total number of employees eligible for bonuses in 2024.
   - The total bonus amount required to be given to the eligible employees.
   - The updated HR records with only the employees eligible for bonuses.

**

Input	Output

company_hr_register ={
 	
101: {'name': 'Alice', 'age': 35, 'performance': 90, 'salary': 50000},
 	
102: {'name': 'Bob', 'age': 58,'performance': 98, 'salary': 70000},
 	
103: {'name': 'Charlie', 'age': 45, 'performance': 85, 'salary': 60000},
 	
104: {'name': 'David', 'age': 60, 'performance': 75, 'salary': 55000},
 	
105: {'name': 'Eve', 'age': 28, 'performance': 92, 'salary': 48000},
 	
106: {'name': 'Frank', 'age': 50, 'performance': 55, 'salary': 52000},
 	
107: {'name': 'Grace', 'age': 62, 'performance': 97, 'salary': 75000}
 	
}
	Total Employees = 3
	Total Bonus Amount = 40000
	
	Updated Company HR Register = {102: {'name': 'Bob'}, 104: {'name': 'David'}, 107: {'name': 'Grace'}}

18

Write a Python program that:

- Ranks the heroes based on their total score in descending order.
- Calculates the average combat score of all universes.
- Finds the hero with the highest strategy score and the universe they belong to.
- Modifies the dictionary to add a new key `"overall_rating"` to each universe, which is the sum of `score`, `combat`, and `strategy`.

universes = {

"Earth-616": {"hero": "Spider-Man", "score": 89, "challenges": {"combat": 45, "strategy": 44}},

"Earth-199999": {"hero": "Iron Man", "score": 95, "challenges": {"combat": 50, "strategy": 45}},

"Earth-1610": {"hero": "Miles Morales", "score": 78, "challenges": {"combat": 38, "strategy": 40}},

"Earth-928": {"hero": "Spider-Man 2099", "score": 92, "challenges": {"combat": 48, "strategy": 44}},

"Earth-TRN123": {"hero": "Deadpool", "score": 85, "challenges": {"combat": 43, "strategy": 42}}

}

Output
Hero Rankings (by total score): 1. Iron Man (Earth-199999) - 190 2. Spider-Man 2099 (Earth-928) - 184 3. Spider-Man (Earth-616) - 178 4. Deadpool (Earth-TRN123) - 170 5. Miles Morales (Earth-1610) - 156

***

Average Combat Score: 44.8

Highest Strategy Score:
Iron Man (Earth-199999) - 45

Updated Universes with Overall Ratings:
{
Earth-616: {'hero': 'Spider-Man', 'score': 89, 'challenges': {'combat': 45, 'strategy': 44}, **'overall_rating': 178**}
Earth-199999: {'hero': 'Iron Man', 'score': 95, 'challenges': {'combat': 50, 'strategy': 45}, **'overall_rating': 190**}
Earth-1610: {'hero': 'Miles Morales', 'score': 78, 'challenges': {'combat': 38, 'strategy': 40}, **'overall_rating': 156**}
Earth-928: {'hero': 'Spider-Man 2099', 'score': 92, 'challenges': {'combat': 48, 'strategy': 44}, **'overall_rating': 184**}
Earth-TRN123: {'hero': 'Deadpool', 'score': 85, 'challenges': {'combat': 43, 'strategy': 42}, **'overall_rating': 170**}
}

19

Meera, a mathematician, has recently developed an interest in prime numbers. She has come across a fascinating concept called **Prime Powers**. Prime powers are numbers obtained by multiplying a prime number by itself a certain number of times. For example, 23=82^3 = 823=8 is a prime power because it is the result of multiplying the prime number 2 by itself three times. Similarly, 32=93^2 = 932=9 and 52=255^2 = 2552=25 are prime powers.

Meera wants to create a Python program that takes an integer n as input and generates a dictionary where:

- The keys are prime numbers.
- The values are lists containing all the prime powers of that prime from 2 to n (inclusive), but **excluding perfect squares**.

Furthermore, she wants the dictionary to **only include prime numbers as keys that have at least one valid prime power under 100**.

Write a Python program that:

1. Accepts an integer n as input.
2. Generates a dictionary with prime numbers as keys and their respective prime powers as values, excluding perfect squares.
3. Ensure that the prime powers are under 100.

Input	Output
Enter a number n: 30	Prime powers dictionary: {2: [8, 32], 3: [27], 5: [25], 7: [49], 11: [121]}
Enter a number n: 50	Prime powers dictionary:

***

	{2: [8, 32], 3: [27], 5: [25], 7: [49], 11: [121], 13: [169], 17: [289]}

20

BONUS FUN QUESTION

***

Daredevil Dress-Up Game

In this task, you are building a fun "dress-up" game for Daredevil. The player will select different clothing items for Daredevil (headgear, upper body, lower body, shoes, and weapon) from predefined categories. Each clothing item influences Daredevil's stats, which include Strength, Stealth, and Agility. However, there are some restrictions and conditions that must be followed when making selections.

Task:
You need to design a system where the player can choose items from different clothing categories, taking into account the following conditions:

Conditions:

1. **Exclusive Choices**: Some items cannot be worn together. For example:
   - **Red mask** cannot be worn with **red pants**.
   - **Red suit** unlocks the **cane** weapon.
   - **Black jacket** cannot be worn with **red boots**.
2. If the user selects conflicting items, notify them with an appropriate message and prompt them to choose again.
3. **Item Unlocking**:
   - Selecting the **red suit** unlocks the **cane** weapon. Ensure that this unlocking mechanism works properly.
4. **Default Options**:
   If the user does not make any selection for a category (by pressing Enter), the default option should be automatically selected as **'none'**.
5. **Stat Modifications**:
   Each clothing item affects Daredevil's stats:
   - **Red mask**: +5 Stealth
   - **Red suit**: +10 Strength
   - **Red pants**: +3 Agility
   - **Red boots**: +2 Agility
   - **Cane**: +7 Strength
6. **Final Output**:
   After the player has made their selections, output Daredevil's full outfit (with each item selected) and the updated stats (**Strength**, **Stealth**, **Agility**).

```
clothing = {

"headgear": ["red mask", "black helmet", "none", "yellow helmet"], "upper_body": ["red suit", "black jacket", "none"], "lower_body": ["red pants", "black pants", "hawaiian pants", "none"], "shoes": ["red boots", "black boots", "none"], "weapon": ["cane", "none"]

}

stats = { "Strength": 0, "Stealth": 0, "Agility": 0 }
```

Input	Output
Choose a headgear from the following options: red mask, black helmet, none Enter your choice for headgear: red mask Choose a upper_body from the following options: red suit, black jacket, none Enter your choice for upper_body: red suit Choose a lower_body from the following options: red pants, black pants, none Enter your choice for lower_body: black pants Choose a shoes from the following options: red boots, black boots, none Enter your choice for shoes: red boots Choose a weapon from the following options: cane, none Enter your choice for weapon: cane	**Daredevil's Outfit:** Headgear: Red mask Upper_body: Red suit Lower_body: Black pants Shoes: Red boots Weapon: Cane **Daredevil's Stats:** Strength: 17 Stealth: 5 Agility: 5