```
''' 1. Write a program to accept student name and marks fromthe user and create a dictionary. Also, display
 student marks by taking student name as input.'''

name = input("Enter student name: ")
marks = int(input("Enter student marks: "))
students = {name:marks}
search_name = input("Enter name to search for marks: ")
if search_name in students:
    print("Marks of", search_name, ":", students[search_name])
else:
    print("Student not found")
```

```
''' 2. Write a program to enter names and percentage of marks in a dictionary and display the information
 on the screen.'''

for _ in range(3):
    name = input("Enter name: ")
    percentage = float(input("Enter percentage: "))
    students = {name:percentage}
print("Student Information:", students)
```

```
''' 3. Write a program to take a user-input dictionary and print the sum of the values'''

dic={} //declaration
n=int(input("Enter the items: "))

for _ in range(n):
    key=input("Enter the key: ")
    value=int(input("Enter the value: "))
    dic[key]=value
print(sum(dic.values()))
```

```
''' 4. Make an English-to-French dictionary called e2f and print it. Here are your starter words: dog is
 chien, cat is chat, and walrus is morse'''

e2f={
    'dog':'chien',
    'cat':'chat',
    'walrus':'morse'
}
print("English to French :",e2f)
```

```
'''5. Adictionary which maps country names to Internet top-level domains (TLDs) is given as follows:
 tlds = {‘Canada’: ‘ca’, ‘United States’: ‘us’, ‘Mexico’: ‘mx’}
 Perform the following tasks and display the results:
 a) Check whether the dictionary contains the key ‘Canada’.
 b) Check whether the dictionary contains the key ‘France’.
 c) Iterate through the key-value pairs and display them in a two-column format.
 d) Add the key–value pair ‘Sweden’ and ‘sw’ (incorrect TLD).
 e) Update the value for the key ‘Sweden’ to ‘se’ (correct TLD).
 f) Use dictionary comprehension to reverse the keys and values.'''

print('Canada' in TLDs)
print('France' in TLDs)
for key in TLDs:
    print(key, ":", TLDs[key])
TLDs['Sweden'] = 'sw'
TLDs['Sweden'] = 'se'
reversed_TLDs = {v: k for k, v in TLDs.items()}
print("Reversed Dictionary:", reversed_TLDs)
```

```
''' 6. For the following dictionary, create lists of its keys, values, and items, and show those lists.
 roman numerals = {‘I’: 1, ‘II’: 2, ‘III’: 3, ‘V’: 5}'''

rm={'I':1,'II':2,'III':3,'V':5}
list1=[rm.keys()]
list2=[rm.values()]
list3=[rm.items()]
print(list1,list2,list3)
```

```
''' 7. Use dictionary comprehension to create a dictionary of the numbers 1–5 mapped to their cubes.'''

dic={}
for i in range(6):
    dic[i]=i**3
print(dic)
```

```
''' 8. Make a multilevel dictionary called life. Use these strings for the topmost keys: ‘animals’, ‘plants’, and
‘other’. Make the ‘animals’ key refer to another dictionary with the keys ‘cats’, ‘octopi’, and ‘emus’. Make the
‘cats’ key refer to a list of strings with the values ‘Henri’, ‘Grumpy’, and ‘Lucy’. Make all the other keys refer to
empty dictionaries.'''

life={
    'animals':{
        'cats':['Henri','Grumpy','Lucy'] ,
        'octopi':{},
        'emus':{}
    },
    'plants':{},
    'other': {}
}
print("LIFE: ",life)
```

```
''' 9. Write a program to find the number of occurrences of each letter present in a given string.,
e.g., str=‘mississippi’ ⇒ {‘m’: 1, ‘i’: 4, ‘s’: 4, ‘p’: 2}'''

string=input("Enter a String: ")
count={}
for s in string:
    if s in count:
        count[s]+=1
    else:
        count[s]=1
print(count)
```

```
''' 10. Write a program to find the number of occurrences of each vowel present in a given string, and also print the
vowels.'''

string=input("Enter a String: ").lower()
vowel={}
for s in string:
    if s in 'aeiou':
        vowel[s] = vowel.get(s, 0) + 1 # Increment count of vowel, default to 0 if not found
print(vowel)
```

```
''' 11. Write a function that takes a number as an input parameter and returns the corresponding text in
 words, e.g., on input 452, the function should return ‘Four Five Two’..'''

def n2w(n):
    d2w = {0: "Zero", 1: "One", 2: "Two", 3: "Three", 4: "Four", 5: "Five", 6: "Six", 7: "Seven", 8: "Eight", 9:
"Nine"}
    print("Number in Words:", " ".join(d2w[int(d)] for d in str(n)))
n=int(input("Enter a number: "))
n2w(n)
```

```
''' 12. Write a program that uses a dictionary to determine and print the number of duplicate words in a
 sentence. Treat uppercase and lowercase letters the same and assume there is no punctuation in the
 sentence.'''

str = input("Enter a string: ").lower()
wordcount = {}
for i in str:
    if i in wordcount:
        wordcount[i] += 1
    else:
        wordcount[i] = 1

duplicate = {k: v for k, v in wordcount.items() if v > 1}
print(duplicate)
```

```
''' 13. Write a function that receives a list of words, then determines and displays in alphabetical order only the
unique words. Treat uppercase and lowercase letters as the same. The function should use a set to get the unique words
in the list. Test your function with several sentences.'''

def uniqueWords(s):
    words = set(s.lower().split())# typecasting to sets because it removes duplicate
    print("Unique Words:",words)
s = input("Enter a Sentence: ")
uniqueWords(s)
```

```
'''14. Create a program that determines and displays the number of unique characters in a string entered by the user,
e.g., Hello, World! has 10 unique characters, while zzz has only one unique character. Use a dictionary or set to
solve this problem.'''

text = "shubhangini"
unique_chars = set(text)
print("Unique Characters Count:", len(unique_chars))
```

```
''' 15. Modify a script to play 1,000,000 games of craps. Use two dictionaries, wins and losses, to track the number
of games won and lost for each roll number. Update these dictionaries as the simulation progresses. For example, a
key-value pair 4: 50217 in the wins dictionary would mean that 50,217 games were won on the 4th roll. At the end of
the simulation, display:
 (i) The percentage of games won.
 (ii) The percentage of games lost.
 (iii) The percentage of games resolved on each roll.
 (iv) The cumulative percentage of games resolved up to each roll.'''

import random
wins, losses = {}, {}
for _ in range(1000000):
    roll = random.randint(1, 6) + random.randint(1, 6)
    if roll not in wins:
        wins[roll] = 0
        losses[roll] = 0
    wins[roll] += 1 if random.choice([True, False]) else 0
    losses[roll] += 1 if not random.choice([True, False]) else 0
print("Wins:", wins)
print("Losses:", losses)
```

```
'''16. Given the sets {10, 20, 30} and {5, 10, 15, 20}, use the mathematical set operators to produce the
 following sets:
 a) {30},
 b) {5, 15, 30}
 c) {5, 10, 15, 20, 30}
 d) {10, 20}'''

A = {10, 20, 30}
B = {5, 10, 15, 20}
a = A - B
b = A ^ B
c = A | B
d = A & B
print("a)", a)
print("b):", b)
print("c):", c)
print("d):", d)
```

```
''' 17. Using the sets {‘red’, ‘green’, ‘blue’}, and {‘cyan’, ‘green’, ‘blue’, ‘magenta’, ‘red’},
 display the results of:'''

A = {'red', 'green', 'blue'}
B = {'cyan', 'green', 'blue', 'magenta', 'red'}
#a) comparing the sets using each of the comparison operators.
print("A == B:", A == B)
print("A != B:", A != B)
print("A <= B:", A <= B)
print("A < B:", A < B)
print("A >= B:", A >= B)
print("A > B:", A > B)
#b) Combining the sets using mathematical set operators:")
print("A | B (Union):", A | B)
print("A & B (Intersection):", A & B)
print("A - B (Difference):", A - B)
print("A ^ B (Symmetric Difference):", A ^ B)
```

```
''' 18. You are given two lists of integers: list1 and list2. Write a Python function analyze list2) that performs the
following tasks:
 • Creates two sets, set1 and set2, from list1 and list2. sets(list1,list2)
 • Finds the symmetric difference of set1 and set2 (elements that are in either set, but not both).
 • For each element in the symmetric difference:– If the element is even, multiply it by 2.– If the element is odd,
add 5 to it.
 • Return a sorted list of the modified elements.'''

def analyze(list1, list2):
    set1 = set(list1) #typecasting
    set2 = set(list2)
    symmetric_diff = set1.symmetric_difference(set2)
    # set1 = {1, 2, 3} and set2 = {3, 4, 5}, then symmetric_diff will be {1, 2, 4, 5}
    modified_elements = []
    for element in symmetric_diff:
        if element % 2 == 0:# It is multiplied by 2 and added to the modified_elements list.
            modified_elements.append(element * 2)
        else:
            modified_elements.append(element + 5)
    return sorted(modified_elements)
list1 = [int(i) for i in input("Enter the first list: ").split()]
list2 = [int(i) for i in input("Enter the second list: ").split()]
result = analyze(list1, list2)
print(result)
```

```
'''19. Givenalonglist of words, write a Python function unique
 of words that:
 pairs(words)tofindalluniquepairs
 • Have no common letters (e.g., "cat" and "dogs" have no letters in common).
 • Each word in the pair should have at least 4 letters.
 • Each unique pair should be stored in a set as a tuple in lexicographical order.
 The function should return a set of all such unique pairs. Example:
 words = ["apple", "dogs", "cat", "bird", "fish", "zebra", "lion"]
 print(unique_pairs(words))'''

def unique_pairs(words):
    words = [word for word in words if len(word) >= 4]
    pairs = set()
    for i in range(len(words)):
        for j in range(i + 1, len(words)):
            if not set(words[i]) & set(words[j]):
                pairs.add(tuple(sorted((words[i], words[j]))))
    return pairs
words = ["apple", "dogs", "cat", "bird", "fish", "zebra", "lion"]
print(unique_pairs(words))
```

```
''' 20. Write a Python program to change Mukesh’s net worth to 9650 in the following dictionary:
 sample_dict = {
 ‘person1’: {‘name’: ‘Bezos’, ‘net worth’: 21,880},
 ‘person2’: {‘name’: ‘Elon’, ‘net worth’: 31,570},
 ‘person3’: {‘name’: ‘Mukesh’, ‘net worth’: 965}
 }'''

sample_dict = {
    'person1': {'name': 'Bezos', 'net worth': 21880},
    'person2': {'name': 'Elon', 'net worth': 31570},
    'person3': {'name': 'Mukesh', 'net worth': 965}
}
sample_dict['person3']['net worth'] = 9650
print(sample_dict)
```