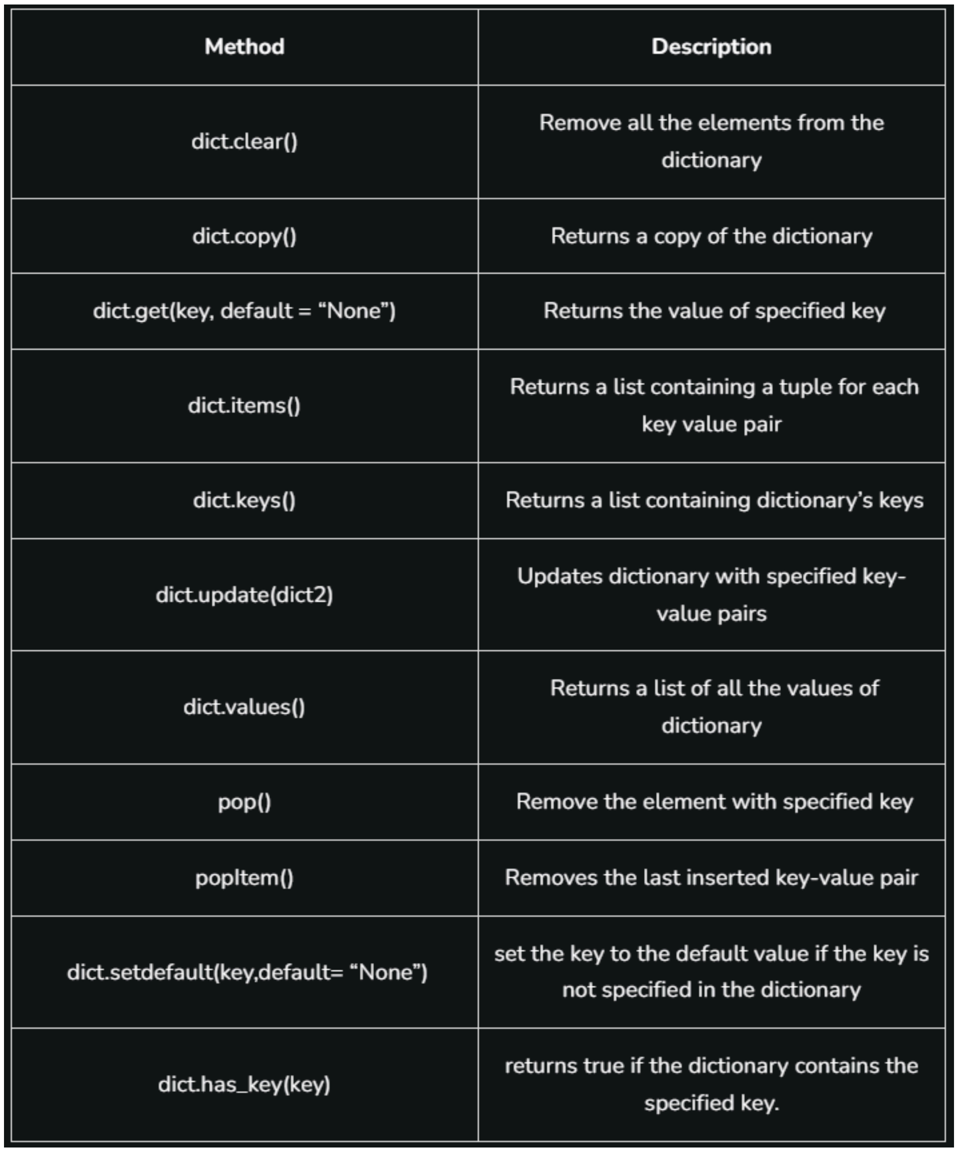

Method	Description
dict.clear()	Remove all the elements from the dictionary
dict.copy()	Returns a copy of the dictionary
dict.get(key, default = “None”)	Returns the value of specified key
dict.items()	Returns a list containing a tuple for each key value pair
dict.keys()	Returns a list containing dictionary’s keys
dict.update(dict2)	Updates dictionary with specified key-value pairs
dict.values()	Returns a list of all the values of dictionary
pop()	Remove the element with specified key
popItem()	Removes the last inserted key-value pair
dict.setdefault(key,default= “None”)	set the key to the default value if the key is not specified in the dictionary
dict.has_key(key)	returns true if the dictionary contains the specified key.

```
a = dict(one=1, two=2, three=3)
b = {'one': 1, 'two': 2, 'three': 3}
c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
d = dict([('two', 2), ('one', 1), ('three', 3)])
e = dict({'three': 3, 'one': 1, 'two': 2})
f = dict({'one': 1, 'three': 3}, two=2)
a == b == c == d == e == f
```

```
True
```

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
tel
```

```
{'jack': 4098, 'sape': 4139, 'guido': 4127}
```

```
{'jack': 4098, 'sape': 4139, 'guido': 4127}
tel['jack']
```

```
4098
```

```
del tel['sape']
tel['irv'] = 4127
tel
```

```
{'jack': 4098, 'guido': 4127, 'irv': 4127}
```

```
{'jack': 4098, 'guido': 4127, 'irv': 4127}
list(tel)
sorted(tel)
```

```
['guido', 'irv', 'jack']
```

```
'guido' in tel
```

```
True
```

```
'jack' not in tel
```

```
False
```

1. Write a Python program to sort Counter by value.

```
dict= {'Math':81,
       'Physics':83,
       'Chemistry':87, }
x = list(dict.items())
x.sort()
print('Ascending order is ',x)
l=list(dict.items())
l.sort(reverse=True)
print('Descending order is',l)
```

```
Ascending order is  [('Chemistry', 87), ('Math', 81), ('Physics', 83)]
Descending order is [('Physics', 83), ('Math', 81), ('Chemistry', 87)]
```

```
d={1:2,3:4,4:3,2:1,0:0}
l=list(d.items())
l.sort()
print('Ascending order is',l)
l=list(d.items())
l.sort(reverse=True)
print('Descending order is',l)

dict=dict(l)

print("Dictionary",dict)
```

```
Ascending order is [(0, 0), (1, 2), (2, 1), (3, 4), (4, 3)]
Descending order is [(4, 3), (3, 4), (2, 1), (1, 2), (0, 0)]
Dictionary {4: 3, 3: 4, 2: 1, 1: 2, 0: 0}
```

```
Dict = {"Name": "Javatpoint",
        "Language": "Python",
        "Data Structure": "Dictionary"}
print("Our dictionary is:", Dict)

keys = list(Dict.keys())
values = list(Dict.values())

print("Value whose key is to be retrieved:")
Value = "Python"
print(Value)

index = values.index(Value)

Key = keys[index]
print(f"The key of {Value} in the dictionary is: ", Key)
```

```
Our dictionary is: {'Name': 'Javatpoint', 'Language': 'Python', 'Data Structure': 'Dictionary'}
Value whose key is to be retrieved:
Python
The key of Python in the dictionary is:  Language
```

```
def get_value(item):
    return item[1]

dict1 = {'GEEKS': 2888, 'For': 1765, 'geeks': 2750}
sorted_items = sorted(dict1.items(), key=get_value)  # Sort items based on values using the get_value function
new_dict = {key: value for key, value in sorted_items}  # Reconstruct dictionary with sorted items
print(new_dict)
```

```
['For', 'geeks', 'GEEKS']
```

Solve

```
dictionary= {
             'Math':81,
             'Physics':83,
             'Chemistry':87,
}

SortedList = sorted(dictionary.items(),key=lambda x:x[1],reverse = True)
print(SortedList)
```

```
[('Physics', 83), ('Math', 81), ('Chemistry', 87)]
```

```
dictionary = {'Math': 81, 'Physics': 83, 'Chemistry': 87}

sorted_list = sorted(dictionary.items(), key=lambda x: x[1], reverse=True)
print(sorted_list)
```

```
[('Chemistry', 87), ('Physics', 83), ('Math', 81)]
```

```
dictionary = {'Math': 81, 'Physics': 83, 'Chemistry': 87}
l=[]
di={}
for y in dictionary .values():
  l.append(y)
l.sort(reverse= True)
for y in l:
  for x in dictionary :
    if dictionary [x] == y:
      di.update({x:y})
print(di)
```

```
{'Chemistry': 87, 'Physics': 83, 'Math': 81}
```

```
dictionary = {'Math': 81, 'Physics': 83, 'Chemistry': 87}
e = {}
for v in reversed(sorted(dictionary.values())):
  for k in dictionary:
    if dictionary[k] == v and k not in e:
        e.update({k:v})
print(e)
```

```
{'Chemistry': 87, 'Physics': 83, 'Math': 81}
```

```
dictionary= {
    'Math':81,
    'Physics':83,
    'Chemistry':87,
}
list1 = []
list2 = []
for i in dictionary.values():
  list1.append(i)
  print(list1)

list1.reverse()
print(list1)
for j in range(len(list1)):
  for item in dictionary.items():
    if list[j] in item:
      list2.append(item)
print(list2)
```

```
[81]
[81, 83]
[81, 83, 87]
[87, 83, 81]
[]
```

```
d= {
    'Physics':83,
    'Math':81,
    'Chemistry':87,
    }


e = {}
for v in sorted(d.values()):  #  v =  81 83 87
  for k in d:   # for k in d:   k =  'Physics' , 'Math'  , 'Chemistry'
    if d[k] == v and k not in e:   # d['Math']  -> 81  == 81
        e.update({k:v})
print("Descending to Ascending:")
print(e)

d= {'Math':81,
       'Physics':83,
       'Chemistry':87, }

e = {}
for v in reversed(sorted(d.values())):
  for k in d:
    if d[k] == v and k not in e:
        e.update({k:v})
print("Ascending to Descending:")
print(e)
```

```
Descending to Ascending:
{'Math': 81, 'Physics': 83, 'Chemistry': 87}
Ascending to Descending:
{'Chemistry': 87, 'Physics': 83, 'Math': 81}
```

2. Write a Python program to combine values in a list of dictionaries.

```
list1 = [{'item': 'item1',
          'amount': 400},

         {'item': 'item2',
          'amount': 300},

         {'item': 'item1',
          'amount': 750}]

for key,value in dict(list1).items:
  print(key)
  print(value)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-1277c49822ec> in <cell line: 10>()
      8           'amount': 750}]
      9
---> 10 for key,value in dict(list1).items:
     11   print(key)
     12   print(value)

TypeError: 'dict' object is not callable
```

```
list1 = [{'item': 'item1',
          'amount': 400},

         {'item': 'item2',
          'amount': 300},

         {'item': 'item1',
          'amount': 750}]

for item in list1:
    for key, value in item.items():
        # print(key)
        print(value)
```

```
item1
400
item2
300
item1
750
```

```
list1 = [{'item': 'item1', 'amount': 400},
         {'item': 'item2', 'amount': 300},
         {'item': 'item1', 'amount': 750}]


output = {}
for i in range(len(list1)):
  # if list1[i]['item'] in output:
  #   for j in output.keys():
      list1[i]['item']=list1[i]['amount']
print(output)
```

```
{}
```

```
list1 = [{'item': 'item1', 'amount': 400},
         {'item': 'item2', 'amount': 300},
         {'item': 'item1', 'amount': 750}]
x = dict(list1)

for i in range(len(list1)):
  for a,b in list1[i].items():
```

```
d1={1:100,2:200,3:300}
d2={1:200,2:700,3:300,4:400}
for i,j in d1.items():
  for i1,j1 in d2.items():
    if i==i1:
      d1[i]=j+j1
print (d1)
```

```
{1: 300, 2: 900, 3: 600}
```

solve

```
list1 = [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]
dict = {}
for i in list1:                          # i  =      {'item': 'item1', 'amount': 750}
  if i['item'] in dict.keys():      # if i['item']  -> 'item1'    in dict.keys():
    dict[i['item']]+=i['amount']
  else:
    dict[i['item']]=i['amount']
print(dict)

# output:
# dict = {'item1':400,'item2':300}
```

```
{'item1': 1150, 'item2': 300}
```

3. Write a Python program to create a dictionary from a string.

Note: Track the count of the letters from the string.

Solve

```
dict={}
for i in "w3resource":
  count=0
  for j in "w3resource":
    if i == j:
      count+=1
    dict[i]=count
print(dict)
```

```
{'w': 1, '3': 1, 'r': 2, 'e': 2, 's': 1, 'o': 1, 'u': 1, 'c': 1}
```

```
A = "w3resource"
dict={}
for i in A:
  B = A.count(i)
  dict[i] = B
print(dict)
```

```
{'w': 1, '3': 1, 'r': 2, 'e': 2, 's': 1, 'o': 1, 'u': 1, 'c': 1}
```

4. Create a dictionary by extracting the keys from a given dictionary

Solve

```
sample_dict = {
                "name": "Kelly",
                "age": 25,
                "salary": 8000,
                "city": "New york"
                }
dict = {}
keys = ['name','salary']
for i in sample_dict.keys():
  if i in keys:
    dict[i]=sample_dict[i]
print(dict)
```

```
{'name': 'Kelly', 'salary': 8000}
```

```
sample_dict = {
                "name": "Kelly",
                "age": 25,
                "salary": 8000,
                "city": "New york"
                }
dict = {}
dict.update({"name":sample_dict["name"],"salary":sample_dict["salary"]})
print(dict)
```

```
{'name': 'Kelly', 'salary': 8000}
```

5. Write a Python program to convert a list to a list of dictionaries.

Solve

```
color = ["Black"      , "Red"       , "Maroon"     , "Yellow"]
code = ["#000000"  ,   "#FF0000"  ,   "#800000"  ,  "#FFFF00"]

output=[]
for i in range(len(color)):
  empty={}
  empty['color_name'] = color[i]
  empty['color_code']=code[i]
  output.append(empty)
print(output)
```

```
[{'color_name': 'Black', 'color_code': '#000000'}, {'color_name': 'Red', 'color_code': '#FF0000'}, {'color_name': 'Maroon', 'color_code'
```

6. Write a Python program that takes a sentence as input and calculates the frequency of each word. The program should output a dictionary where keys are unique words, and values are the frequencies of those words.

Solve

```
A = "the quick brown fox jumps over the lazy dog the lazy dog barks"
x = A.split()
print(x)
dict={}
for i in x:     # ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', 'the', 'lazy', 'dog', 'barks']
  count = 0
  for j in x:   # ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', 'the', 'lazy', 'dog', 'barks']
    if i == j:
      count+=1
  dict[i]=count
print(dict)
```

```
['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', 'the', 'lazy', 'dog', 'barks']
{'the': 3, 'quick': 1, 'brown': 1, 'fox': 1, 'jumps': 1, 'over': 1, 'lazy': 2, 'dog': 2, 'barks': 1}
```

7. Given a list of integers, write a Python program to create a dictionary where the keys are the integers, and the values are their squares, but only for odd integers.

Solve

```
A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
dict = {}
for i in A:
  if i%2!=0:
    dict[i]=i*i
print(dict)
```

```
{1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
```

8. UIU database has a dictionary containing student data. Each key is a student's name,

and the corresponding value is a dictionary with information such as the student's name,

major, CGPA (Cumulative Grade Point Average), and completed credits. Imagine you

are a software engineer working at UIU, you are assigned to write a Python program that will transform this dictionary filled with redundancy to a optimized one.

Now, Create a new dictionary with the same student names as keys and values generated by extracting the "cgpa" and "completed_credits" information for each student and arrange them sorted based on their CGPA.

Solve

```
Original_student_data={

                        'Emma': {'name': 'Emma', 'major':
                        'Computer Science', 'cgpa': 3.8,
                        'completed_credits': 90},

                        'Daniel': {'name': 'Daniel', 'major':
                        'Electrical Engineering', 'cgpa': 3.5,
                        'completed_credits': 75},

                        'Sophia': {'name': 'Sophia', 'major':
                        'Mechanical Engineering', 'cgpa': 3.2,
                        'completed_credits': 60}
}

empty1 = {}
empty2 = {}
values = Original_student_data.values()
for i in values:
  empty2 = {}
  empty2['cgpa']= i['cgpa']
  empty2['completed_credits']= i['completed_credits']

  empty1[i["name"]]=empty2
print(empty1)
sortedlist = sorted(empty1.items(),key = lambda x:x[1]['cgpa'])
print(sortedlist)
sortedlist = sorted(empty1.items(),key = lambda x:x[1]['cgpa'],reverse=True)
print(sortedlist)
```

```
{'Emma': {'cgpa': 3.8, 'completed_credits': 90}, 'Daniel': {'cgpa': 3.5, 'completed_credits': 75}, 'Sophia': {'cgpa': 3.2, 'completed_cr
[('Sophia', {'cgpa': 3.2, 'completed_credits': 60}), ('Daniel', {'cgpa': 3.5, 'completed_credits': 75}), ('Emma', {'cgpa': 3.8, 'complet
[('Emma', {'cgpa': 3.8, 'completed_credits': 90}), ('Daniel', {'cgpa': 3.5, 'completed_credits': 75}), ('Sophia', {'cgpa': 3.2, 'complet
Original_student_data={

                        'Emma': {'name': 'Emma', 'major':
                        'Computer Science', 'cgpa': 3.8,
                        'completed_credits': 90},

                        'Daniel': {'name': 'Daniel', 'major':
                        'Electrical Engineering', 'cgpa': 3.5,
                        'completed_credits': 75},

                        'Sophia': {'name': 'Sophia', 'major':
                        'Mechanical Engineering', 'cgpa': 3.2,
                        'completed_credits': 60}
}

empty1 = {}
empty2 = {}
list1 = []
values = Original_student_data.values()
for i in values:
  empty2 = {}
  empty2['cgpa']= i['cgpa']
  empty2['completed_credits']= i['completed_credits']
  for j in Original_student_data.keys():
    if j not in empty1:
      empty1[j]=empty2
print(empty1)
```

```
{'Emma': {'cgpa': 3.8, 'completed_credits': 90}, 'Daniel': {'cgpa': 3.8, 'completed_credits': 90}, 'Sophia': {'cgpa': 3.8, 'completed_cr
```

Solve

```
Original_student_data = {
    'Emma': {'name': 'Emma', 'major': 'Computer Science', 'cgpa': 3.8, 'completed_credits': 90},
    'Daniel': {'name': 'Daniel', 'major': 'Electrical Engineering', 'cgpa': 3.5, 'completed_credits': 75},
    'Sophia': {'name': 'Sophia', 'major': 'Mechanical Engineering', 'cgpa': 3.2, 'completed_credits': 60}
}

empty1 = {}
empty2 = {}

for student in Original_student_data.values():
    # Create a new dictionary for each student
    empty2 = {}
    # Copy 'cgpa' and 'completed_credits' from Original_student_data to empty2
    empty2['cgpa'] = student['cgpa']
    empty2['completed_credits'] = student['completed_credits']

    # Add empty2 to empty1 with the student's name as key
    empty1[student['name']] = empty2

print(empty1)
```

```
{'Emma': {'cgpa': 3.8, 'completed_credits': 90}, 'Daniel': {'cgpa': 3.5, 'completed_credits': 75}, 'Sophia': {'cgpa': 3.2, 'completed_cr
```

9. In the upcoming year 2024, a tech company is looking to update its HR records implemented in Python dictionaries. The company plans 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 a performance below 60% will be terminated from the records and will receive no bonus.

Compute the total number of employees in the company in 2024, the overall bonus amount required and show the updated dictionary.

```
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},
}
updated_company_hr_register =  {}
for i in company_hr_register:
  empty1 = {}
  if company_hr_register[i]['age']>=55 and company_hr_register[i]['performance']>=90:
    empty1['name'] = company_hr_register[i]['name']
    updated_company_hr_register[i] = empty1
print(updated_company_hr_register)
```

```
{102: {'name': 'Bob'}, 107: {'name': 'Grace'}}
```

```
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},
}

updated_company_hr_register={}
total_employees = 0
for i in company_hr_register:
    if company_hr_register[i]["age"]>=55 and company_hr_register[i]["performance"]>=90:
        updated_company_hr_register[i]={"name":company_hr_register[i]["name"]}
        total_employees+=1
print(f'total_employees = {total_employees}')
print
print(updated_company_hr_register)
```

```
total_employees = 2
{102: {'name': 'Bob'}, 107: {'name': 'Grace'}}
```

Differnt output but good code

```
def process_hr_records(company_hr_register):
  """
  Processes the HR records, calculates bonuses, and updates the dictionary.

  Args:
      company_hr_register: A dictionary containing employee information.

  Returns:
      A tuple containing the total number of employees, total bonus amount,
      and the updated HR register dictionary.
  """

  total_employees = 0
  total_bonus_amount = 0
  updated_register = {}

  for emp_id, employee_data in company_hr_register.items():
    age = employee_data["age"]
    performance = employee_data["performance"]

    # Update employee data with bonus information
    bonus = 0
    if age > 55:
      bonus += 10000  # Senior employee bonus
    if performance > 95:
      bonus += 5000  # Performance bonus

    # Remove employees with low performance
    if performance < 60:
      continue

    # Update employee data in the new dictionary
    updated_data = {"name": employee_data["name"]}
    if bonus > 0:
      updated_data["bonus"] = bonus
    updated_register[emp_id] = updated_data

    total_employees += 1
    total_bonus_amount += bonus

  return total_employees, total_bonus_amount, updated_register

# Test the function with the sample data
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, total_bonus_amount, updated_company_hr_register = process_hr_records(company_hr_register.copy())

print("Total Employees:", total_employees)
print("Total Bonus Amount:", total_bonus_amount, "BDT")
print("Updated HR Register:")
print(updated_company_hr_register)
```

```
Total Employees: 6
Total Bonus Amount: 40000 BDT
Updated HR Register:
{101: {'name': 'Alice'}, 102: {'name': 'Bob', 'bonus': 15000}, 103: {'name': 'Charlie'}, 104: {'name': 'David', 'bonus': 10000}, 105: {'
```

Solve

```
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},
}

updated_company_hr_register={}
total_employees = 0
total_bonus = 0
for i in company_hr_register:
    if company_hr_register[i]["age"]>=55 and company_hr_register[i]["performance"]>=90:
        updated_company_hr_register[i]={"name":company_hr_register[i]["name"]}
        total_employees+=1
    if company_hr_register[i]["age"]>55 :
        total_bonus+=10000
    if company_hr_register[i]["performance"]>95:
        total_bonus+=5000
    if company_hr_register[i]["performance"] < 60:
        continue
print(f'total_employees = {total_employees}')
print(f'total_bonus_amount = {total_bonus}')
print(updated_company_hr_register)
```

```
total_employees = 2
total_bonus_amount = 40000
{102: {'name': 'Bob'}, 107: {'name': 'Grace'}}
```

10. A mathematician Meera has found interest in prime numbers. She discovered a really

fascinating concept called Prime Powers. Prime powers are numbers obtained by

multiplying a prime number by itself a certain number of times. For example, 2^^3 is a

prime power because it is the result of multiplying the prime number 2 by itself three

times which is also a prime number (2 * 2 * 2 = 8). Similarly, 2^^2,2^^5 all are prime powers.

She is trying to create a Python program that takes an integer n as input and generates

a dictionary where the keys are prime numbers and the values are lists containing

all the prime powers from 2 to n. However, she does not want to include the prime

powers that are perfect squares and does not want to display the keys that have no

values. Hint: make sure the value of prime powers are under 100.

```
n = int(input())
f = 0
if n == 1 or n ==0:
  f = 1
for i in range(2,n):
  if n%i == 0:
    f=1
if f==1:
  print('Number is not prime')
else:
  print('number is prime')
```

```
number=int(input("Input: "))
empty={}
PowerOfTwo=[]
PowerOfThree=[]
if n == 1 or n ==0:
  f = 1
for i in range(2,n):
  if n%i == 0:
    f=1
if f==1:
  a = 0
else:
  a = 1
for i in range(2,number):
  if number%i != 0 and i:
    PowerOfTwo.append(i)
print(PowerOfTwo)
```

```
Input: 10
[3, 4, 6, 7, 8, 9]
```

```
def get_prime_powers(n):
  """
  This function takes an integer n as input and returns a dictionary where:
    - Keys are prime numbers
    - Values are lists containing prime powers of the corresponding key
      (less than or equal to n) and are not perfect squares

  Args:
      n: An integer representing the upper limit for prime powers.

  Returns:
      A dictionary containing prime powers as described.
```

```
# Input an integer from the user
num = int(input("Enter an integer: "))

# Check if the input integer is prime or not
if num <= 1:
    is_prime = False
else:
    is_prime = True
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            is_prime = False
            break

# Print the result
if is_prime:
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def generate_prime_powers(n):
    prime_powers_dict = {}

    for i in range(2, n + 1):
        if is_prime(i):
            prime_powers = []
            prime = i
            power = prime
            prime1 = 1
            while power < 100:
                x = power % prime
                if power != prime * prime:
                    prime_powers.append(power)
                    print(prime_powers)
                power = power * prime
                prime1+=1
            if prime_powers is not []:
                prime_powers_dict[prime] = prime_powers

    return prime_powers_dict

n = int(input("Enter the value of n: "))
prime_powers = generate_prime_powers(n)
print("Prime powers dictionary:")
print(prime_powers)
```