Notes by Abdullah Salfi

# Basics of

# Python

## Programming

## With 20 Solved Coding Examples

Python is a versatile programming language that is widely used in various fields, including web development, data analysis, artificial intelligence, and more. In this document, we will cover the basic concepts of Python programming, including variables, data types, operators, control flow statements, collections, functions, and more.

See All Pages➡️

# 1. Variables

In Python, variables are used to store data values.Variables are created by assigning a value to a name using the assignment operator `=`. Unlike some other programming languages, in Python, you don't need to explicitly declare the data type of a variable. Python determines the data type automatically based on the value assigned to the variable.

Example:

```
x = 10      # Here, 'x' is a variable storing an integer value 10

name = "John"      # Here, 'name' is a variable storing a string value "John"
```

# 2. Data Types

Integers:
Integers are whole numbers without any decimal point. For example, `10`, `-5`, `1000`, etc.

Floating-Point Numbers:
(floats) are numbers with a decimal point. For example, `3.14`, `2.71828`, `-0.5`, etc.

Strings:
Strings are sequences of characters enclosed in single (' ') or double (" ") quotes. For example, `"hello"`, `'world'`, `"Python is fun!"`, etc.

Booleans:
Booleans represent truth values. They can either be `True` or `False`.

print()  Print is built in function to display integers strings and more....

Note: Always try to enter comments using # hashtag in start of line to understand your code
Note: Always take care of indentation by using either Spaces or Tabs while coding

Examples Codes:

```
1   print ("Hello World")
2
3   my_num = 5
4   print(my_num, type(my_num))
5
6   my_float = 2.5
7   my_string = "How are you?"
8   print(my_float, type(my_float), my_string, type(my_string))
9
10  bool_t , bool_f = True , False
11  print(type(bool_t) , type(bool_f))
```

Output:

```
Hello World
5 <class 'int'>
2.5 <class 'float'> How are you? <class 'str'>
<class 'bool'> <class 'bool'>
```

## Assignment Operators

Assignment operators are used to assign values to variables.

Example:

```
x = 10  # Assigns the value 10 to variable x
y += 5  # Equivalent to y = y + 5
```

## Comparison Operators

Comparison Operators: Comparison operators are used to compare two values and return a boolean result (True or False).

Example:

```
x == y  # Checks if x is equal to y
x != y  # Checks if x is not equal to y
x < y   # Checks if x is less than y
x > y   # Checks if x is greater than y
x <= y  # Checks if x is less than or equal to y
x >= y  # Checks if x is greater than or equal to y
```

## Logical Operators

Logical operators are used to combine conditional statements.

Example:

```
x and y  # Returns True if both x and y are True
x or y   # Returns True if either x or y is True
not x    # Returns True if x is False
```

## Identity Operators

Identity operators are used to compare the memory location of two objects.

Example:

```
x is y     # Returns True if both variables refer to the same object
x is not y # Returns True if both variables do not refer to the same object
```

### Membership Operators

Membership operators are used to test if a value or variable is found in a sequence (e.g., list, tuple, string).

Example:

```
x in myList    # Returns True if x is present in myList
x not in myList  # Returns True if x is not present in myList
```

## 4. Control Flow Statements

Control flow statements allow you to control the execution of your code based on certain conditions or loop through a sequence of code multiple times.

### If Else Statements

If-else statements allow you to execute different blocks of code based on whether a condition is true or false. You can also chain multiple conditions using elif (short for "else if").
Example:

```
# If Else Statements
x = 10
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

### While Loop

While loop repeatedly executes a block of code as long as a given condition is true. It's useful when you want to repeat an action until a specific condition is met.
Example:

```
# List example
my_list = [1, 2, 3, 4, 5]
```

### For Loop

While loop repeatedly executes a block of code as long as a given condition is true. It's useful when you want to repeat an action until a specific condition is met.
Example:

```
# For Loop
for i in range(5):
    print(i)
```

# 4. Collections

Collections in Python are used to store multiple values in a single variable. Python provides several built-in collection types, each with its own unique characteristics

## Lists

Lists are ordered collections of items. They are mutable, meaning you can change the elements after creation. Lists are defined using square brackets [].
Example:

```
# List example
my_list = [1, 2, 3, 4, 5]
```

## Tuples

Tuples are similar to lists but are immutable, meaning their elements cannot be changed after creation. Tuples are defined using parentheses ().

Example:

```
# Tuple example
my_tuple = (1, 2, 3, 4, 5)
```

## Sets

Sets are unordered collections of unique items. They do not allow duplicate elements. Sets are defined using curly braces {}.

Example:

```
# Set example
my_set = {1, 2, 3, 4, 5}
```

## Dictionaries

Dictionaries: Dictionaries are collections of key-value pairs. Each key is associated with a value, and keys must be unique within a dictionary. Dictionaries are defined using curly braces {} and key-value pairs separated by colons : .
Example:

```
# Dictionary example
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}
```

# 5. Nested Collections

In Python, it's possible to have collections (lists, tuples, sets, or dictionaries) nested within other collections. This allows for more complex data structures where elements themselves can be collections.

## Nested Lists

Nested lists are commonly used to represent matrices, tables, trees, or any other hierarchical data structures.
Example of accessing elements in a nested list:

```
nested_list = [1, 2, [3, 4]]
```

## Nested Tuples

Nested tuples are tuples that contain other tuples as elements.

Example:

```
# Nested tuple example
nested_tuple = ((1, 2), (3, 4), (5, 6))
```

## Nested Sets

Nested Sets are unordered collections of unique elements. Nested sets occur when one or more sets are elements of another set.

Example of a nested set:

```
nested_set = {1, 2, {3, 4}}
```

## Nested Dictionaries

Nested dictionaries are dictionaries where the values can themselves be dictionaries.

Example:

```
# Nested dictionary example
nested_dict = {
    'person1': {'name': 'Alice', 'age': 30},
    'person2': {'name': 'Bob', 'age': 25},
    'person3': {'name': 'Charlie', 'age': 35}
}
```

# 6. Functions

Functions in Python are blocks of reusable code that perform a specific task. They allow you to break down your program into smaller, more manageable pieces, making your code more organized, readable, and modular.

## Function Definition

To define a function in Python, you use the def keyword followed by the function name and parentheses ( ), which may contain parameters (inputs) to the function. The function body is then indented below the function definition.

Example:

```
def greet(name):
    print("Hello, " + name + "!")
```

## Function Call

Function Call: To execute a function and perform its task, you simply use the function name followed by parentheses ( ), optionally passing any required arguments (inputs) inside the parentheses.

Example:

```
greet("Alice")
```

**- Different Methods of Creating Functions:**

**Function without Parameters:**

```
def greet():
    print("Hello, world!")

greet()  # Function call
```

**Function with Parameters:**

```
def greet(name):
    print("Hello, " + name + "!")

greet("Alice")  # Function call with argument
```

# Functions

## Function with Default Parameters:

```
def greet(name="world"):
    print("Hello, " + name + "!")

greet()  # Function call without argument (uses default value)
greet("Alice")  # Function call with argument
```

## Function with Return Value:

```
def add(x, y):
    return x + y

result = add(3, 5)  # Function call with arguments and storing the return value
print("Result:", result)
```

## Function with Variable Number of Arguments (Arbitrary Arguments):

```
result = add(1, 2, 3, 4, 5)  # Function call with variable number of arguments
print("Result:", result)
def add(*args):
    sum = 0
    for num in args:
        sum += num
    return sum
```

## Function with Keyword Arguments (kwargs):

```
def greet(**kwargs):
    print("Hello, " + kwargs['name'] + "!")

greet(name="Alice")  # Function call with keyword argument
```

## Lambda Functions (Anonymous Functions):

```
double = lambda x: x * 2
print(double(5))  # Output: 10
```

# Some Useful Built-in Function in Python

## Function to handle strings:

len(string): Returns the length of the string.
string.upper(): Returns a copy of the string converted to uppercase.
string.lower(): Returns a copy of the string converted to lowercase.
string.strip(): Returns a copy of the string with leading and trailing whitespace removed.
string.split(separator): Splits the string into a list of substrings based on the specified separator.
string.replace(old, new): Returns a copy of the string with all occurrences of the substring 'old' replaced by 'new'.

string.find(substring): Returns the index of the first occurrence of the substring within the string, or -1 if not found.

string.startswith(prefix): Returns True if the string starts with the specified prefix; otherwise, returns False.

## Strings:

- `len()`: Returns the length of a string.
- `str()`: Converts an object into a string.
- `upper()`: Converts a string to uppercase.
- `lower()`: Converts a string to lowercase.
- `strip()`: Removes leading and trailing whitespaces from a string.
- `split()`: Splits a string into a list of substrings based on a delimiter.
- `join()`: Joins elements of an iterable into a string using a separator.

# Some Useful Built-in Function in Python

## Function to handle lists:

Lists:

len(list): Returns the number of elements in the list.
list.append(element): Adds an element to the end of the list.
list.pop(): Removes and returns the last element from the list.
list.insert(index, element): Inserts an element at the specified index in the list.
list.remove(element): Removes the first occurrence of the specified element from the list.
list.index(element): Returns the index of the first occurrence of the specified element in the list.
list.count(element): Returns the number of occurrences of the specified element in the list.
list.sort(): Sorts the elements of the list in ascending order.
list.reverse(): Reverses the order of the elements in the list

## Lists:

`len()`: Returns the length of a list.

`list()`: Creates a new list.

`append()`: Adds an element to the end of a list.

`extend()`: Adds elements from an iterable to the end of a list.

`insert()`: Inserts an element at a specified position in a list.

`remove()`: Removes the first occurrence of a specified value from a list.

`pop()`: Removes and returns an element at a specified index; if no index is specified, removes and returns the last element.

`index()`: Returns the index of the first occurrence of a value in a list.

`count()`: Returns the number of occurrences of a value in a list.

`sort()`: Sorts the elements of a list in ascending order.

`reverse()`: Reverses the order of elements in a list.

These are some of the basic built-in functions in Python for handling strings, tuples, dictionaries, sets, and lists.

# Some Useful Built-in Function in Python

### Function to handle tuples:

Tuples:

len(tuple): Returns the number of elements in the tuple.

tuple.count(value): Returns the number of occurrences of a specified value in the tuple.

tuple.index(value): Returns the index of the first occurrence of the specified value in the tuple.

## Tuples:

`len()`: Returns the length of a tuple.

`tuple()`: Converts an iterable to a tuple.

`count()`: Returns the number of occurrences of a value in a tuple.

`index()`: Returns the index of the first occurrence of a value in a tuple.

# Some Useful Built-in Function in Python

## Function to handle sets:

len(set): Returns the number of elements in the set.
set.add(element): Adds an element to the set.
set.remove(element): Removes the specified element from the set.
set.discard(element): Removes the specified element from the set if it is present.
set.pop(): Removes and returns an arbitrary element from the set.
set.clear(): Removes all elements from the set

**Sets:**

`len()`: Returns the number of elements in a set.
`set()`: Creates a new set.
`add()`: Adds an element to a set.
`remove()`: Removes a specified element from a set; raises an error if the element is not present.
`discard()`: Removes a specified element from a set if it is present, without raising an error if the element is not present.
`pop()`: Removes and returns an arbitrary element from a set; raises an error if the set is empty.
`union()`: Returns a new set containing all unique elements from two or more sets.
`intersection()`: Returns a new set containing common elements from two or more sets.
`difference()`: Returns a new set containing elements present in the first set but not in the second set.
`symmetric_difference()`: Returns a new set containing elements that are present in either of the sets, but not in both.

# Some Useful Built-in Function in Python

## Function to handle dictionaries:

Dictionaries:

len(dictionary): Returns the number of key-value pairs in the dictionary.
dictionary.keys(): Returns a view object that displays a list of all the keys in the dictionary.
dictionary.values(): Returns a view object that displays a list of all the values in the dictionary.
dictionary.items(): Returns a view object that displays a list of key-value tuples in the dictionary.
dictionary.get(key): Returns the value associated with the specified key, or a default value if the key is not found.

dictionary.pop(key): Removes the key-value pair with the specified key from the dictionary and returns the value.

## Dictionaries:

`len()`: Returns the number of key-value pairs in a dictionary.
`dict()`: Creates a new dictionary.
`keys()`: Returns a view object of all keys in a dictionary.
`values()`: Returns a view object of all values in a dictionary.
`items()`: Returns a view object of all key-value pairs in a dictionary.
`get()`: Returns the value of a specified key, or a default value if the key does not exist.
`update()`: Updates a dictionary with key-value pairs from another dictionary or an iterable of key-value pairs.

# Example Coding Tasks for Practice with Solutions

**Question 1**

Write a program that takes a string as input and counts the number of vowels and consonants in the string. (5 marks) Input: Hello, World! Output: Number of vowels: 3 Number of consonants: 7

**Solution**

```
# Taking input string from user
random_string = input('Type a string to know number of count vowels and number of consonants: ')

# displaying extracted alphabets only
print('Your provided string: ', random_string)
# intializing void string variable
result = ""

# using for loop to extract only alphabets from user input string
for char in random_string:

  # using if statement and isalpha function() to check alphabets only
  if char.isalpha():

    # storing extracted alphabets in a variable
    result += char

# displaying extracted alphabets only
print('Extracted all alphabets from your string: ', result)

# initializing two void integer variables

vowel_count = 0
consonant_count = 0

# storing vowels(both in upper and lower case) in a string variable

vowels = 'AaEeIiOoUu'

# using for loop to iterate string stored in 'result' variable
for display_string in result:

   # using if_else statement to compare and count vowels and consonants in string 'result'
  if display_string in vowels:
    vowel_count +=1
  else:
    consonant_count +=1

# printing number of vowels and consonants in string 'result'
print('Number of vowels:' , vowel_count)
print('Number of consonants:' , consonant_count)
```

```
Type a string to know number of count vowels and number of consonants: dEaI u 1#
Your provided string:  dEaI u 1#
Extracted all alphabets from your string:  dEaIu
Number of vowels: 4
Number of consonants: 1
```

# Example Coding Tasks for Practice with Solutions

**Question 2**

Write a Python program that accepts a filename from the user and prints the extension of the file. (5 marks) Sample filename: document.docx
Output: docx

**Solution**

```
[ ]  # Taking filename in a string from user
     file_name = input('Type a filename with its extension here: ')

     # Using Split(), dividing filename by using "." into two parts(i-e filename and extension) of list
     split_f_name = file_name.split(".")

     # displaying and confirming type of splitted input filename string
     print(split_f_name)
     print(type(split_f_name))

     # printing the extension of file by using indicing
     print(split_f_name[1])
```

```
Type a filename with its extension here: f.rar
['f', 'rar']
<class 'list'>
rar
```



**Question 3**

Write a Python program to check if value 200 exists in the following dictionary. (5 marks) Sample_dict = {'a': 100, 'b': 200, 'c': 300}

**Solution**

```
[ ]  # Initializing dictionary with keys and values
     Sample_dict = {'a': 100, 'b': 200 , 'c' : 300 }

     # Using if else statement to check and print whether value 200 exists in dictionary or not
     if 200 in Sample_dict.values():
       print('value 200 exists in Dictionary' )
     else:
       print('value 200 does not exist')
```

```
value 200 exists in Dictionary
```

**Question 4**

The given tuple is a nested tuple. Write a Python program to print the value 20. (5 marks) tuple1 = ("Orange", [10, 20, 30], (5, 15, 25))

**Solution**

```
[ ]  # Initializing nested tuples in a single tuple 'tuple1'
     tuple_1 = ("Orange", [10, 20, 30], (5, 15, 25))
     print (tuple_1[1][1])
     #x=0
     #y=0
     #for items in tuple1:
      # if 20 in tuple1:
      #   print('item in main tuple: ', 20)

      # x+=1
      # for sub_items in tuple1:
      #   if 20 in tuple1[x][y]:
      #     print('item in sub tuple or list:',20)
      #   y+=1
```

```
20
```

# Example Coding Tasks for Practice with Solutions

**Question 5**

Write a Python program that calculates the Body Mass Index (BMI) for a person based on their weight (in kilograms) and height (in meters). The BMI is calculated using the following formula: (5 marks) BMI = (weight) / (height^2) Instructions: • Prompt the user to enter their weight in kilograms. • Prompt the user to enter their height in meters. • Calculate the BMI using the provided formula. • Display the calculated BMI to the user. Additionally, provide an interpretation of the BMI according to the following categories: • BMI < 18.5: Underweight • 18.5 <= BMI < 25: Normal weight • 25 <= BMI < 30: Overweight • BMI >= 30: Obese

**Solution**

```
# Taking weight in an input as an integer from user
Weight =int(input('Enter weight in kilogram: '))
# Assigning variable to weight
a = Weight
# printing 'a'
print('Weight = ', a)
# Taking Height in an input as an integer from user
Height = float(input('Enter height in meters: '))
# Assigning variable to height square
b = Height**2
# printing height to avoid error
print('Height = ', Height)
# using Formula "BMI = Weight/Height^2"
BMI = a/b
# printing 'BMI
print('Body Mass Index (BMI) = ', BMI )
# using condition 'if else' to know where the BMI lies, "Underweight, Normal or Overweight"
if BMI < 18.5:
  print('Underweight')
elif 18.5 <= BMI < 25:
  print('Normal weight')
elif 25 <= BMI < 30:
  print('Overweight')
else:
  print('Obese')
```

```
Enter weight in kilogram: 66
Weight =  66
Enter height in meters: 1.7
Height =  1.7
Body Mass Index (BMI) =  22.837370242214536
Normal weight
```

# Example Coding Tasks for Practice with Solutions

**Question 6**

Write a Python program that prompts the user to input a string. The program should count the frequency of each character in the string and then print the result as a dictionary.(5 marks) EXAMPLE: If the user enters "hello" , the program should output {'h': 1, 'e': 1, 'l': 2, 'o': 1} indicating that 'h' appears once, 'e' appears once, 'l' appears twice, and 'o' appears once in the input string.

**Solution**

```
# Taking string as an input from user
String_input = input('Type a string here: ')
# Initializing Dictionary
dict ={}
# Using for loop to iterate each character and 'i' is the characters in string
for i in String_input:
  # printing characters 'i'
  print(i)
  # using if condition to know the frequency of each character and intializing dictionary too
  if i in dict:
    #if character lie more than one then, add this into the frequency of that character
    dict[i] += 1
    print(dict[i])
  else:
    # if character occurs just once, then print it once
    dict[i] = 1
    print(dict[i])
print(dict)
```

```
Type a string here: hello
h
1
e
1
l
1
l
2
o
1
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

# Example Coding Tasks for Practice with Solutions

**Question 7**

Write a Python program that takes a string input from the user and checks if it is a valid email address. If it is valid, print "Valid email address", otherwise print "Invalid email address".(5 marks) Hint: Check if the string contains "@" and ends with ".net", or ".com" to check validity. Example: If the email is "abc@example.com", then it is a valid email, otherwise an Invalid email.

**Solution**

```
[ ]  # Taking email address as a string input from user
     Email_address = input('Enter the email address: ')
     # Using split() to separate email address into parts, i.e, before '.' and after '.'
     split_email = Email_address.split(".")
     # printing last word of the email like "com" or "net" etc
     print(split_email[-1])
     # Giving variables to fundamental and special characters
     a = "@"
     b = "com"
     c = "net"
     # Using 'if else' to know the email is valid or not, if valid then email should have "@" and any one [".com",".net"] from the following
     if a in split_email[0] and b or c in split_email[-1]:
       print('Valid email address')
```

```
else:
  # Otherwise print invalid email without having sufficient characters
  print('Invalid email address')
```

```
Enter the email address: abc@gmail.com
com
Valid email address
```

**Question 8**

Given a tuple of tuples, write a Python program to print the sum of the elements in each inner tuple. (5 marks) Input: ((1, 2, 3), (4, 5, 6), (7, 8, 9)) Output: 6, 15, 24

**Solution**

```
# Initializing the given tuple that is nested tuple
given_tuple = ((1, 2, 3),(4, 5, 6), (7, 8, 9))
# Using for loop t iterate values in 'given tuple'
for item in given_tuple:
  # Defing variable to the sum of elements of each inner tuple
  j = (item[0]+ item[1]+ item[2])
  # printing the sum of elements of each inner tuple'
  print(j, end = " ")
```

```
6 15 24
```

**Question 9**

Create a list containing the 10 numbers . Print the first Three prime numbers from the list. (5 marks) Expected Output: List of the first three prime numbers: [2, 3, 5]

**Solution**

```
[ ]  # Initializing the list having first 10 natural numbers
     my_list= (1,2,3,4,5,6,7,8,9,10)
     # Using for loop to iterate the values
     for num in my_list:
       # Using if statement to exclude 1 and less then 1 values from the list
       if num <= 1:
         # Using 'pass' to not to include those values <=1
         pass
         # Using elif to print 2 and 3 because ther are prime numbers
       elif num <= 3:
         # printing thses numbers and using 'end' to print these numbers in a single line
         print(num, end = ' ')
         #using elif to exclude all multiples of 2 and 3 and 7 from the list
       elif num % 2 == 0 or num % 3 == 0 or num% 7 == 0:
         # Using 'pass' to not to include those values
         pass
         # Printng all remaing numbers that will be prime numbers in the list and using 'end' to print these numbers in a single line
       else:
           print(num, end = " ")
```

```
2 3 5
```

# Example Coding Tasks for Practice with Solutions

**Question 10**

Given a list of dictionaries representing students' information (name, age, grade), Write a Python program to sort the students based on their grades in descending order and print the sorted list. (5 marks) Dictionary: [ {"name": "John", "age": 20, "grade": 85}, {"name": "Alice", "age": 22, "grade": 90}, {"name": "Bob", "age": 21, "grade": 80} ]

**Solution**

```
# Initializing the list 'y' having students names with their age and grades in a list of dictionary
y = [{"name": "John", "age": 20, "grade": 85},{"name": "Alice", "age": 22, "grade": 90},{"name": "Bob", "age": 21, "grade": 80}]
# Using for loop and giving variables 'x' , 'y' and 'z' to names, age and grades
for x in y:
  # printing each variable w.r.t their attributes
  print(x['name'])
  print(x['age'])
  print(x['grade'])
  # Using 'lambda functions to sort the students based on their grades, then use reverse function to sort the grades in descending order
sorted_dict = (sorted(y, key=lambda x:x['grade'], reverse=True))
# printng sorted dictonary
print(sorted_dict)
```

```
John
20
85
Alice
22
90
Bob
21
80
[{'name': 'Alice', 'age': 22, 'grade': 90}, {'name': 'John', 'age': 20, 'grade': 85}, {'name': 'Bob', 'age': 21, 'grade': 80}]
```

**Question 11**

Create a program that takes a list of multiple numbers separated with commas (,) from the user and finds the total number of unique values in the list and also prints them. (5 marks) Expected_format: Total number of unique values: 6 Unique values in the list: [1, 2, 3, 4, 5, 6

**Solution**

```
# Taking my_int_list as an integer input from user
my_int_list = int(input("Type the numbers here: "))
# Using split() to split integer from ","
split_my_int_list = my_int_list.split(",")
# Printing splited string
print(split_my_int_list)
# Initializing set in order to get output in the form of set
set_1 = set()
# Using for loop to iterate integers in 'my_int_list'
for i in split_my_int_list:
  # Use add() to find the total
  set_1.add(i)
# Using len() to know the number of unique valus in the list using variable 'j'
j = len(set_1)
# Printing the variable 'j'
print("Total number of unique value: ", j)
# printing added set
print("Unique values in the list: " , set_1)
```

```
Type the numbers here: 4,5,6,7,8,5,7
['4', '5', '6', '7', '8', '5', '7']
Total number of unique value:  5
Unique values in the list:  {'8', '6', '5', '7', '4'}
```

# Example Coding Tasks for Practice with Solutions

**Question 12**

Write a program that takes two lists as input and returns a new list containing elements that are common to both lists.(Hint: Use list methods to implement this.) (5 marks) Input Lists: List 1: [1, 2, 3, 4, 5] List 2: [4, 5, 6, 7, 8] Common Elements: [4, 5]

**Solution**

```
[ ]  # Taking List1 and List2 with their elemnets
     List1 = [1, 2, 3, 4, 5]
     List2 = [4, 5, 6, 7, 8]
     # using lambda () find common elements or intersection of List1 and List2 and giving them the variable 'x'
     x= list(filter(lambda x:x in List1, List2))
     #x = [value for value in List1 if value in List2]
     # Printing the variable 'x'
     print("Common Elements" , x)
```

```
[4, 5]
```

**Question 13**

Write a program that takes two dictionaries as input and merges them into a new dictionary. If there are common keys, combine their values. (5 marks) dict1= {'a':5, 'b':10, 'c':15} dict2= {'b':20, 'c':25, 'd':30} Output: {'a': 5, 'b': 30, 'c': 40, 'd': 30}

**Solution**

```
[ ]  #from collections import Counter
     # Initializng dict1 and dict2 with their keys and values
     dict1= {"a":5, "b":10, "c":15}
     dict2= {"b":20, "c":25, "d":30}
     # Introducing dict3 that is equal to dict2
     dict3 =dict2
     #dict3= Counter(dict1) + Counter(dict2)
     # Using for loop to iterate the values and representing with variabe 'i' for keys and ' j' for values
     for i, j in dict1.items():
       # Using 'if statement to compare keys and values in dict 2 with elements in dict3 ,if they are same then, add their values
         if i in dict2:
             dict3[i]= j + dict2[i]
         else:
           # if not, then leave it as they are by including values in dict 3
             dict3[i]= j
             # printing the values and adding the values of the same keys avaliable in dict1 and dict2
     print(dict3)
```

```
{'b': 30, 'c': 40, 'd': 30, 'a': 5}
```

**Question 14**

Write a program that takes two numbers, base and exponent, and calculates the result of raising base to the power of exponent using the ** operator. (5 marks) Input: Base: 2 Exponent: 3 Result: 8

**Solution**

```
# Taking Base as an int input from user
Base =int(input("Enter the base here: "))
# Taking Exponent as an int input from user
Exponent = int(input("Enter the exponent here: "))
# Taking Output by making the formula base^exponent
Output = Base ** Exponent
# printing Output of the given base and exponent
print("Result= ", Output)
```

```
Enter the base here: 3
Enter the exponent here: 2
Output=  9
```

# Example Coding Tasks for Practice with Solutions

**Question 18**

Implement a Python function is_palindrome() that checks whether a passed string is a palindrome (reads the same forward and backward) and tests it with "radar".(5 marks)

**Solution**

```
[ ]  # is_palindrome() is defined by using "s" asa parameter, which is the string to be checked
     def is_palindrome(s):
       # The funtion returns True if the string is Palindrome and false otherwise
       return s == s[::-1]
     # Assigning the string "radar" to the variable
     string1 = "radar"
     # Using if condition to check whether the string is palindrome
     if is_palindrome(string1):
       # If the string is palindrome then print it is palindrome
       print("Passed string is a palindrome: " , string1)
     else:
       # If not then print it is not pslindrome
         print("Passed string is not a palindrome: ", string1)
```

```
Passed string is not a palindrome do
```

**Question 19**

Write a program that returns a list of even numbers from a given list. (5 marks) Input: [2, 5, 3, 7, 9, 53, 10, 32, 65, 76, 98]

**Solution**

```
[ ]  # Taking list1 having random numbers in the list
     list1 = [2, 5, 3, 7, 9, 53, 10, 32, 65, 76, 98, 102]
     # initializing 'y' and list2
     y= 0
     list2  = []
     # Using for loop to iterate the values in list 1
     for i in list1:
       # Using if statement to know the even numbers by dividing each number by 2
       if i%2== 0:
         # add the value to the variable to the end of the list
         list2.append(i)
          # Value is incremented by one
         y+=1
         # variable containing the value to be printed
     print(list2)
```

```
[2, 10, 32, 76, 98, 102]
```

**Question 20**

Given two lists keys = ['a', 'b', 'c'] and values = [1, 2, 3], write a Python program to create a dictionary from these lists. (5 marks) Expected Output: {'a': 1, 'b': 2, 'c': 3}

**Solution**

```
[ ]  # The list variable 'keys1' is being initialized
     keys1 = ['a', 'b', 'c']
     # The list variable 'keys1' is being initialized
     values1 = [1, 2, 3]
     # Initializing dictionary
     dict2 = {}
     #dict1 = dict(zip(keys1, values1))
     # Using for loop and use this function "range(len(keys))", because we need both the index and the value of each element in the list
     for i in range(len(keys1)):
       # keys and values lists together based on their indices
          dict2[keys1[i]] = values1[i]
     #print(dict1)
     # displays the entire dictionary including al keys and their values corresponding values
     print(dict2)
```

```
{'a': 1, 'b': 2, 'c': 3}
```

# End