**PYTHON QUESTION BANK WITH ANSWERS**

**Unit – 1**

**Questions 2 & 3**

**2. Discuss the int(), float(), str(), chr() and complex() type conversion functions with examples.**

**3.Discuss the ord(), hex(), oct(), complex() and float() type conversion functions with examples.**

# **Ans:- *Type Conversion in Python***

**Python defines type conversion functions to directly convert one data type to another which is useful in day to day and competitive programming. This article is aimed at providing the information about certain conversion functions.**

**1. int(a,base) : This function converts any data type to integer. ‘Base’ specifies the base in which string is if data type is string.**

**2. float() : This function is used to convert any data type to a floating point number**

**# Python code to demonstrate Type conversion**

**# using int(), float()**

**# initializing string**

**s = "10010"**

**# printing string converting to int base 2**

**c = int(s,2)**

**print ("After converting to integer base 2 : ", end="")**

**print (c)**

**# printing string converting to float**

**e = float(s)**

**print ("After converting to float : ", end="")**

**print (e)**

**Output:**

**After converting to integer base 2 : 18**

**After converting to float : 10010.0**

**str() : Used to convert integer into a string.**

**complex(real,imag) : : This function converts real numbers to complex(real,imag) number.**

|  |
| --- |
| **# Python code to demonstrate Type conversion**  **# using  dict(), complex(), str()**    **# initializing integers**  **a = 1**  **b = 2**    **# initializing tuple**  **tup = (('a', 1) ,('f', 2), ('g', 3))**    **# printing integer converting to complex number**  **c = complex(1,2)**  **print ("After converting integer to complex number : ",end="")**  **print (c)**    **# printing integer converting to string**  **c = str(a)**  **print ("After converting integer to string : ",end="")**  **print (c)**    **# printing tuple converting to expression dictionary**  **c = dict(tup)**  **print ("After converting tuple to dictionary : ",end="")**  **print (c)** |

**Output:**

**After converting integer to complex number : (1+2j)**

**After converting integer to string : 1**

**After converting tuple to dictionary : {'a': 1, 'f': 2, 'g': 3}**

****chr(number) :**: This function**converts number to its corresponding ASCII character.****

|  |
| --- |
| **# Convert ASCII value to characters**  **a = chr(76)**  **b = chr(77)**    **print(a)**  **print(b)** |

**Output:**

**L**

**M**

****hex() :**This function is to convert **integer to hexadecimal string**.**

****oct() :**This function is to convert **integer to octal string**.**

|  |
| --- |
| **# Python code to demonstrate Type conversion**  **# using  ord(), hex(), oct()**    **# initializing integer**  **s = '4'**    **# printing character converting to integer**  **c = ord(s)**  **print ("After converting character to integer : ",end="")**  **print (c)**    **# printing integer converting to hexadecimal string**  **c = hex(56)**  **print ("After converting 56 to hexadecimal string : ",end="")**  **print (c)**    **# printing integer converting to octal string**  **c = oct(56)**  **print ("After converting 56 to octal string : ",end="")**  **print (c)** |

**Output:**

**After converting character to integer : 52**

**After converting 56 to hexadecimal string : 0x38**

**After converting 56 to octal string : 0o70**

**4.Describe the is and is not operators and type() function. Also, discuss why Python is called as dynamic and strongly typed language**

****‘is’ operator –** Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.**

|  |
| --- |
| **# Python program to illustrate the use**  **# of 'is' identity operator**  **x = 5**  **if (type(x) is int):**  **print("true")**  **else:**  **print("false")** |

**Output:**

**true**

**Python is strongly typed as the interpreter keeps track of all variables types. It's also very dynamic as it rarely uses what it knows to limit variable usage. In Python, it's the program's responsibility to use built-in functions like isinstance() and issubclass() to test variable types and correct usage. Python tries to stay out of your way while giving you all you need to implement strong type checking.**

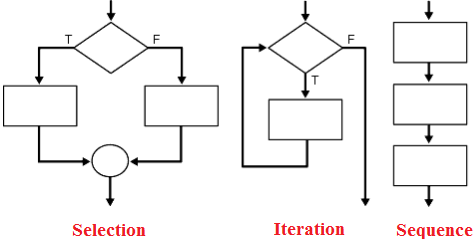
* **In a weakly typed language a compiler / interpreter will sometimes change the type of a variable. For example, in some languages (like [JavaScript](https://wiki.python.org/moin/JavaScript)) you can add strings to numbers 'x' + 3 becomes 'x3'. This can be a problem because if you have made a mistake in your program, instead of raising an exception execution will continue but your variables now have wrong and unexpected values. In a strongly typed language (like Python) you can't perform operations inappropriate to the type of the object - attempting to add numbers to strings will fail. Problems like these are easier to diagnose because the exception is raised at the point where the error occurs rather than at some other, potentially far removed, place.**
* **In a statically typed language, the type of variables must be known (and usually declared) at the point at which it is used. Attempting to use it will be an error. In a dynamically typed language, objects still have a type, but it is determined at runtime. You are free to bind names (variables) to different objects with a different type. So long as you only perform operations valid for the type the interpreter doesn't care what type they actually are.**

**Unit – 2**

**1.Illustrate the different types of control flow statements available in Python with flowcharts.**

**According to the structure theorem, any computer program can be written using the basic control structures . A control structure (or flow of control) is a block of programming that analyses variables and chooses a direction in which to go based on given parameters. In simple sentence, a control structure is just a decision that the computer makes. So, it is the basic decision-making process in programming and flow of control determines how a computer program will respond when given certain conditions and parameters.**

**There are two basic aspects of computer programming: data and instructions . To work with data, you need to understand variables and data types; to work with instructions, you need to understand control structures and statements. Flow of control through any given program is implemented with three basic types of control structures: Sequential, Selection and Repetition**



## **Sequential**

**Sequential execution is when statements are executed one after another in order. You don't need to do anything more for this to happen.**

## **Selection**

**Selection used for decisions, branching - choosing between 2 or more alternative paths.**

* **if**
* **if...else**
* **switch**

## **Repetition**

**Repetition used for looping, i.e. repeating a piece of code multiple times in a row.**

* **while loop**
* **do..while loop**
* **for loop**

**These control structures can be combined in computer programming. A sequence may contain several loops; a loop may contain a loop nested within it, or the two branches of a conditional may each contain sequences with loops and more conditionals. From the following lessons you can understand the control structures and statements in Python language.**

**2. Write a Program to Prompt for a Score between 0.0 and 1.0. If the Score is out of range, print an error. If the score is between 0.0 and 1.0, print a grade using the following table**

****score = input("Enter Score: ")**  
**try:**  
**score69=float(score)**  
**except:**  
**print('Try a number dumb!!')**  
**quit()**  
**if score69 >= 1.0 or score69 < 0.0:**  
**print('Out of Range Dumb!!')**  
**elif score69 >= 0.9:**  
**print('A')**  
**elif score69 >= 0.8:**  
**print('B')**  
**elif score69>=0.7:**  
**print('C')**  
**elif score69>=0.6:**  
**print('D')**  
**elif score69<0.6:**  
**print('F')****

**3. Write a program to display the fibonacci sequences up to nth term where n is provided by the user.**

**A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8…. The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means to say the nth term is the sum of (n-1)th and (n-2)th term.**

### **Source Code:**

1. **# Program to display the Fibonacci sequence up to n-th term where n is provided by the user**
3. **# take input from the user**
4. **nterms = int(input("How many terms? "))**
6. **# first two terms**
7. **n1 = 0**
8. **n2 = 1**
9. **count = 2**
11. **# check if the number of terms is valid**
12. **if nterms <= 0:**
13. **print("Plese enter a positive integer")**
14. **elif nterms == 1:**
15. **print("Fibonacci sequence:")**
16. **print(n1)**
17. **else:**
18. **print("Fibonacci sequence:")**
19. **print(n1,",",n2,end=', ')**
20. **while count < nterms:**
21. **nth = n1 + n2**
22. **print(nth,end=' , ')**
23. **# update values**
24. **n1 = n2**
25. **n2 = nth**
26. **count += 1**

### **Output :**

**How many terms? 10**

**Fibonacci sequence:**

**0, 1, 1, 2, 3, 5, 8, 13, 21, 34,**

### **Explanation**

**Here, we ask the user for the number of terms in the sequence. We initialize the first term to 0 and the second term to 1. If the number of terms is more than 2, we use a while loop to find the next term in the sequence by adding the preceding two terms. We then interchange the variables (update it) and continue on with the process.**

**4. Write a program to repeatedly check for the largest number until the user enters “done”.**

**largest = None  
smallest = None  
while True:  
try:  
num = raw\_input("Enter a number: ")  
if num == "done":  
break  
#print (num)  
num = int(num)  
if largest is None or largest < num:  
largest = num  
elif smallest is None or smallest > num:  
smallest = num  
except ValueError:  
print("Invalid input")**

**print ("Maximum is", largest)  
print ("Minimum is", smallest)**

**#Enter 7,2,bob,10,4 u will get desired output**

**5. Write a program to find the sum of all Odd and Even numbers up to a number specified by the user.**

**This is a C program to find the sum of odd and even numbers from 1 to N.**

**Problem Description**

**The program takes the number N and finds the sum of odd and even numbers from 1 to N.**

**Problem Solution**

**1. Take the number N upto which we have to find the sum as input.  
2. Using for loop take the elements one by one from 1 to N.  
3. Using if,else statements separate the element as even or odd.  
4. Add the even and odd numbers separately and store it in different variables.  
5. Print the sum separately and exit.**

**Program/Source Code**

2. **#include <stdio.h>**
4. **void main()**
5. **{**
6. **int i, num, odd\_sum = 0, even\_sum = 0;**
8. **printf("Enter the value of num\n");**
9. **scanf("%d", &num);**
10. **for (i = 1; i <= num; i++)**
11. **{**
12. **if (i % 2 == 0)**
13. **even\_sum = even\_sum + i;**
14. **else**
15. **odd\_sum = odd\_sum + i;**
16. **}**
17. **printf("Sum of all odd numbers = %d\n", odd\_sum);**
18. **printf("Sum of all even numbers = %d\n", even\_sum);**
19. **}**

**Program Explanation**

**1. User must first enter the number upto which he/she wants to find the sum and is stored in the variable num.  
2. Using for loop take the elements one by one from 1 to num.  
3. Use if,else statement for each element to find whether it is odd or even by dividing the element by 2.  
4. Initialize the variables odd\_sum and even\_sum to zero.  
5. If the element is even,then increment the variable even\_sum with the current element.  
6. If the element is odd,then increment the variable odd\_sum with the current element.  
7. Print the variables odd\_sum and even\_sum separately and exit.**

**Case 1:**

**Enter the value of num**

**10**

**Sum of all odd numbers = 25**

**Sum of all even numbers = 30**

**Case 2:**

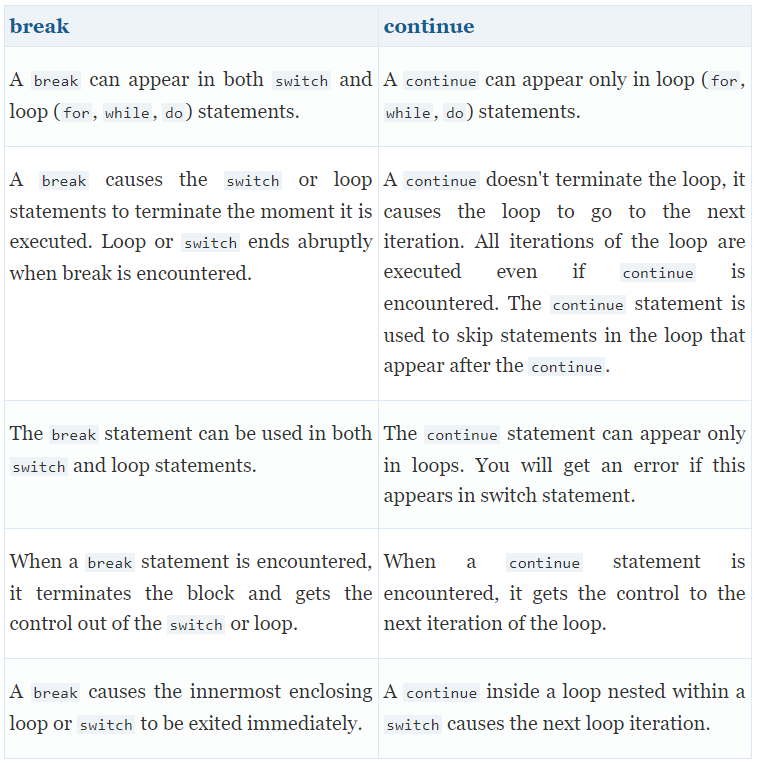
**Enter the value of num**

**100**

**Sum of all odd numbers = 2500**

**Sum of all even numbers = 2550**

**6. Explain the need for continue and break statements. Write a program to check whether a number is prime or not. Prompt the user for input.**

****

**Program to check if a number is prime or not**

**num = 407**

**# To take input from the user**

**#num = int(input("Enter a number: "))**

**# prime numbers are greater than 1**

**if num > 1:**

**# check for factors**

**for i in range(2,num):**

**if (num % i) == 0:**

**print(num,"is not a prime number")**

**print(i,"times",num//i,"is",num)**

**break**

**else:**

**print(num,"is a prime number")**

**# if input number is less than**

**# or equal to 1, it is not prime**

**else:**

**print(num,"is not a prime number")**

**Output**

**407 is not a prime number**

**11 times 37 is 407**

**6.Describe the syntax for the following functions and explain with an example. a) abs() b) max() c) divmod() d) pow() e) len()**

**Python 3 comes with many built-in functions that you can readily use in any program that you’re working on. Some functions enable you to [convert data types](https://www.digitalocean.com/community/tutorials/how-to-convert-data-types-in-python-3), and others are specific to a certain type, like [strings](https://www.digitalocean.com/community/tutorials/an-introduction-to-string-methods-in-python-3).**

**This tutorial will go through a few of the built-in functions that can be used with numeric data types in Python 3. We’ll go over the following functions:**

* **abs() for absolute value**
* **divmod() to find a quotient and remainder simultaneously**
* **pow() to raise a number to a certain power**
* **round() to round a number to a certain decimal point**
* **sum() to calculate the sum of the items in an iterable data type**

### **abs()**

* **The built-in function abs() will return the absolute value of a number that you pass to it. In mathematics, **absolute value** refers to the distance that a number is on the number line from 0. Absolute value does not take into consideration which direction from zero the number lies, meaning that negative numbers will be represented with positive numbers.**
* **To give some examples, the absolute value of 15 is 15, the absolute value of -74 is 74, and the absolute value of 0 is 0.**
* **Absolute value is an important concept for calculus and real analysis, but it also makes sense when we think about everyday situations like distance travelled. For example, if we are trying to get somewhere that is 58 miles away but we travel 93 miles instead, we overshot our original destination. If we want to calculate now how many miles left to travel to get to the intended destination, we’ll end up with a negative number, but we can’t travel negative miles.**
* **Let’s use abs() to solve this problem:**

***syntax***

* **miles\_from\_origin = 58 # Miles destination is from origin**
* **miles\_travelled = 93 # Miles travelled from origin to destination (too many)**
* **# Calculate how many miles destination is from current location:**
* **miles\_to\_go = miles\_from\_origin - miles\_travelled**
* **print(miles\_to\_go) # Print how many miles left (a negative number)**
* **print(abs(miles\_to\_go)) # Use absolute value to account for negative number**
* **Copy**
* **Output**
* **-35**
* **35**

**Divmod()**

**Because both floor division (which returns a quotient), and modulo division (which returns a remainder), are closely related, it can be useful to use a function that combines both operations at once.**

**The Python built-in function divmod() combines the two, returning first the quotient that comes from floor division, then the remainder.**

**Because divmod() will be working with two numbers, we need to pass two numbers to it.**

**divmod(a,b)**

**Copy**

**With this function we are basically performing the following:**

**a // b**

**a & b**

**syntax**

**words = 80000 # How many words in our book**

**per\_page\_A = 300 # Option A, 300 words per page**

**per\_page\_B = 250 # Option B, 25- words per page**

**print(divmod(words,per\_page\_A)) # Calculate Option A**

**print(divmod(words,per\_page\_B)) # Calculate Option B**

**Output**

**(266, 200)**

**(320, 0)**

**Because the function divmod() can take both integers and floats, let’s also go through an example that uses floats:**

**a = 985.5**

**b = 115.25**

**print(divmod(a,b))**

**Output**

**(8.0, 63.5)**

**Keep in mind that you can use the floor division operator // and the modulo operator % to verify what divmod() did:**

**print(a//b)**

**print(a%b)**

**Output**

**8.0**

**63.5**

**Power()**

**In Python, you can use [the operator](https://www.digitalocean.com/community/tutorials/how-to-do-math-in-python-3-with-operators" \l "power) \*\* to raise a number by an exponent, or you can use the built-in function pow() which takes in two numbers.**

**To see how the pow() function works, let’s say we are doing research on bacteria and want to see how many bacteria we’ll have at the end of the day if we start with 1. The particular bacteria we’re working with doubles each hour, so we’ll be calculating 2 (doubling) to the power of the total number of hours (24 in our case).**

**hours = 24**

**total\_bacteria = pow(2,hours)**

**print(total\_bacteria)**

**Copy**

**Output**

**16777216**

**We passed two integers to the pow() function and determined that by the end of this 24-hour period, we’ll have over 16 million bacteria.**

**In mathematics, if we want to calculate 3 to the power of 3, it is generally written like this:**

**3³**

**The computation that we are completing is 3 x 3 x 3, which is equal to 27.**

**To calculate 3³ in Python, we would type out pow(3,3).**

**The function pow() will take both integers and floats, and provides an alternative to using the \*\* operator when you intend to raise numbers to a certain power.**

**8.Write Pythonic code to solve the quadratic equation ax\*\*2 + bx + c = 0 by getting input for coefficients from the user**

**# Solve the quadratic equation ax\*\*2 + bx + c = 0**

**# import complex math module**

**import cmath**

**a = 1**

**b = 5**

**c = 6**

**# calculate the discriminant**

**d = (b\*\*2) - (4\*a\*c)**

**# find two solutions**

**sol1 = (-b-cmath.sqrt(d))/(2\*a)**

**sol2 = (-b+cmath.sqrt(d))/(2\*a)**

**print('The solution are {0} and {1}'.format(sol1,sol2))**

**Output**

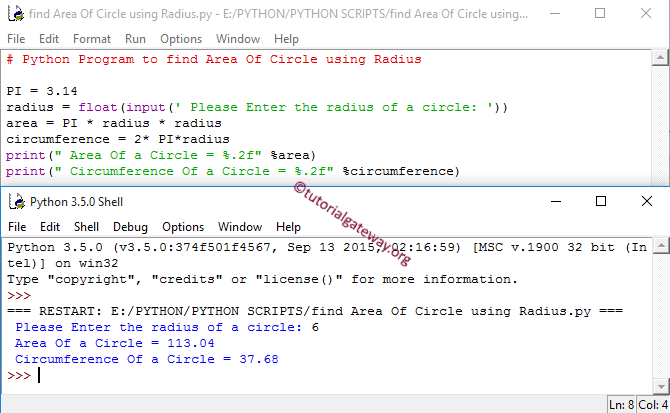
**Enter a: 1**

**Enter b: 5**

**Enter c: 6**

**The solutions are (-3+0j) and (-2+0j)**

**9. Find the area and perimeter of a circle using functions. Prompt the user for input.**

****

**import math**

**circumference = float(input(' Please Enter the Circumference of a circle: '))**

**area = (circumference \* circumference)/(4 \* math.pi)**

**print(" Area Of a Circle = %.2f" %area)**

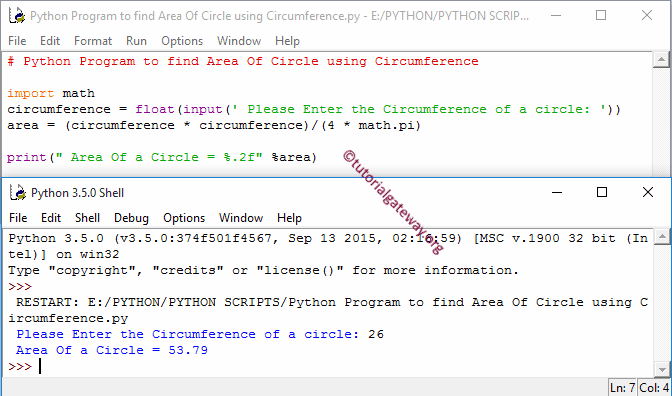
**First, We imported the math library and this supports us to use all the mathematical functions in Python programming. In this [Python](https://www.tutorialgateway.org/python-tutorial/) example we can call the PI value using *math.pi***

**import math**

**Next line of code allows user to enter the value of a circumference.**

**circumference = float(input(' Please Enter the Circumference of a circle: '))**

**Using the circumference, this program will calculate the area of circle as per the formula: A= C²⁄ 4π**

**Output:  
**

**10. Write a Python program using functions to find the value of nPr and nCr without using inbuilt factorial() function**

**Given two numbers n and r, the task is to find the value of nPr.**

**nPr represents n [permutation](https://www.geeksforgeeks.org/permutation-and-combination/) r which is calculated as n!/(n-k)!. Permutation refers to the process of arranging all the members of a given set to form a sequence. The number of permutations on a set of n elements is given by n!, where “!” represents factorial.**

**# Python3 program to calculate nPr**

**import math**

**def fact(n):**

**if (n <= 1):**

**return 1**

**return n \* fact(n - 1)**

**def nPr(n, r):**

**return math.floor(fact(n) /**

**fact(n - r))**

**# Driver code**

**n = 5**

**r = 2**

**print(n, "P", r, "=", nPr(n, r))**

**# This code contributed by Rajput-Ji**

**Output:**

**5P2 = 20**

**11. Write a program to print the sum of the following series 1 + 1/2 + 1/3 +. …. + 1/n**

**If inverse of a sequence follows rule of an A.P i.e, Arithmetic progression, then it is said to be in Harmonic Progression.In general, the terms in a harmonic progression can be denoted as : 1/a, 1/(a + d), 1/(a + 2d), 1/(a + 3d) …. 1/(a + nd).  
As Nth term of AP is given as ( a + (n – 1)d) .Hence, Nth term of harmonic progression is reciprocal of Nth term of AP, which is : 1/(a + (n – 1)d)  
where “a” is the 1st term of AP and “d” is the common difference.**

**# Python program to find the sum of series**

**def sum(n):**

**i = 1**

**s = 0.0**

**for i in range(1, n+1):**

**s = s + 1/i;**

**return s;**

**# Driver Code**

**n = 5**

**print("Sum is", round(sum(n), 6))**

**# This code is contributed by Chinmoy Lenka**

**12. Write a function which receives a variable number of strings as arguments. Find unique characters in each string**

**Variable length argument is a feature that allows a function to receive any number of arguments. There are situations where we want a function to handle variable number of arguments according to requirement.  
1) Sum of given numbers.  
2) Minimum of given numbers.  
and many more.**

**Variable number of arguments are represented by three dotes (…)**

**// C program to demonstrate working of**

**// variable arguments to find average**

**// of multiple numbers.**

**#include <stdarg.h>**

**#include <stdio.h>**

**int average(int num, ...)**

**{**

**va\_list valist;**

**int sum = 0, i;**

**va\_start(valist, num);**

**for (i = 0; i < num; i++)**

**sum += va\_arg(valist, int);**

**va\_end(valist);**

**return sum / num;**

**}**

**// Driver code**

**int main()**

**{**

**printf("Average of {2, 3, 4} = %d\n",**

**average(2, 3, 4));**

**printf("Average of {3, 5, 10, 15} = %d\n",**

**average(3, 5, 10, 15));**

**return 0;**

**}**

**Output:**

**Average of {2, 3, 4} = 3**

**Average of {3, 5, 10, 15} = 10**

**13. Check if the items in the list are sorted in ascending or descending order and print suitable messages accordingly. Otherwise, print “Items in list are not sorted”**

**14. Write Pythonic code to multiply two matrices using nested loops and also perform transpose of the resultant matrix**

**In Python, we can implement a matrix as nested list (list inside a list).**

**We can treat each element as a row of the matrix.**

**For example X = [[1, 2], [4, 5], [3, 6]] would represent a 3x2 matrix.**

**The first row can be selected as X[0]. And, the element in first row, first column can be selected as X[0][0].**

**Multiplication of two matrices X and Y is defined only if the number of columns in X is equal to the number of rows Y.**

**If X is a n x m matrix and Y is a m x l matrix then, XY is defined and has the dimension n x l (but YX is not defined). Here are a couple of ways to implement matrix multiplication in Python.**

**Syntax**

**# Program to multiply two matrices using nested loops**

**# 3x3 matrix**

**X = [[12,7,3],**

**[4 ,5,6],**

**[7 ,8,9]]**

**# 3x4 matrix**

**Y = [[5,8,1,2],**

**[6,7,3,0],**

**[4,5,9,1]]**

**# result is 3x4**

**result = [[0,0,0,0],**

**[0,0,0,0],**

**[0,0,0,0]]**

**# iterate through rows of X**

**for i in range(len(X)):**

**# iterate through columns of Y**

**for j in range(len(Y[0])):**

**# iterate through rows of Y**

**for k in range(len(Y)):**

**result[i][j] += X[i][k] \* Y[k][j]**

**for r in result:**

**print(r)**

**Output**

**[114, 160, 60, 27]**

**[74, 97, 73, 14]**

**[119, 157, 112, 23]**

**15. Write Python program to sort words in a sentence in decreasing order of their length. Display the sorted words along with their length**

**Examples:**

**Input : to learn programming refer geeksforgeeks**

**Output : geeksforgeeks learn programming refer to**

**Input : geeks for geeks**

**Output : for geeks geeks**

**Syntax:**

**# Function to sort the words**

**# in ascending order**

**def sortedSentence(Sentence):**

**# Splitting the Sentence into words**

**words = Sentence.split(" ")**

**# Sorting the words**

**words.sort()**

**# Making new Sentence by**

**# joining the sorted words**

**newSentence = " ".join(words)**

**# Return newSentence**

**return newSentence**

**# Driver's Code**

**Sentence = "to learn programming refer geeksforgeeks"**

**# Print the sortedSentence**

**print(sortedSentence(Sentence))**

**Sentence = "geeks for geeks"**

**# Print the sortedSentence**

**print(sortedSentence(Sentence))**

**Output:**

**geeksforgeeks learn programming refer to**

**for geeks geeks**

**The syntax for sorting by creating a new list is:**

**sorted\_list = sorted(unsorted\_list, key=len)**

**# Python code to sort a list by creating**

**# another list Use of sorted()**

**def Sorting(lst):**

**lst2 = sorted(lst, key=len)**

**return lst2**

**# Driver code**

**lst = ["rohan", "amy", "sapna", "muhammad",**

**"aakash", "raunak", "chinmoy"]**

**print(Sorting(lst))**

****output****

**['amy', 'rohan', 'sapna', 'aakash', 'raunak', 'chinmoy', 'muhammad']**

****The syntax for sorting without creating a new list is:****

**unsorted\_list.sort(key=len)**

**# Python code to sort a list without**

**# creating another list Use of sort()**

**def Sorting(lst):**

**lst.sort(key=len)**

**return lst**

**# Driver code**

**lst = ["rohan", "amy", "sapna", "muhammad",**

**"aakash", "raunak", "chinmoy"]**

**print(Sorting(lst))**

****output****

**['amy', 'rohan', 'sapna', 'aakash', 'raunak', 'chinmoy', 'muhammad']**

**16. Write Pythonic code to create a function called most\_frequent that takes a string and prints the letters in decreasing order of frequency. Use dictionaries.**

**def most\_frequent(string):**

**d = dict()**

**for key in string:**

**if key not in d:**

**d[key] = 1**

**else:**

**d[key] += 1**

**return d**

**print most\_frequent('aabbbc')**

**Returning:**

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

**Now I need to:**

1. **reverse the pair**
2. **sort by number by decreasing order**
3. **only print the letters out**