1.palindrome

num = int(input('enter the value :'))  
temp = num  
rev = 0  
while (num > 0):  
 dig = num % 10  
 rev = rev \* 10 + dig  
 num = num // 10  
if (temp == rev):  
 print('the value is a palindrome!')  
else:  
 print('the value is not palindrome!')

2.factorial

x = int(input("Insert any number: "))  
fact=1  
while x > 1:  
 fact \*= x  
 x -= 1  
print("The result of factorial = ", fact)

3. fibbonacci series

def fib(n):  
 if n==1:  
 return 0  
 elif n==2:  
 return 1  
 else:  
 return (fib(n-1)+fib(n-2))  
  
print("fibbonacci series")  
for i in range(1,10):  
 print(fib(i),end=" ")

4. Armstrong number

n = int(input("Enter a number: "))  
sum = 0  
temp=n  
while n > 0:  
 dig= n%10  
 sum = sum + (dig \*\* 3)  
 n=n//10  
if temp == sum:  
 print(temp,"is an Armstrong number")  
else:  
 print(temp,"is not an Armstrong number")

5.calculator

choice = input("Enter operator to use:")  
A = int(input("Enter first number: "))  
B = int(input("Enter second number: "))  
if choice == '+':  
 ans=A+B  
elif choice == '-':  
 ans=A+B  
elif choice == '\*':  
 ans=A+B  
elif choice == '/':  
 ans=A+B  
else:  
 print("Invalid input")  
  
print("the answer is",ans)

6.patterns

for i in range(5):  
 for j in range(5):  
 print("\*",end=" ")  
 print(" ")  
  
for i in range(5):  
 for j in range(0,i+1):  
 print("\*",end=" ")  
 print(" ")  
  
for i in range(1,6):  
 for j in range(1,i+1):  
 print(j,end=" ")  
 print(" ")  
  
x=0  
for i in range(0,5):  
 x+=1  
 for j in range(0,i+1):  
 print(x,end=" ")  
 print(" ")  
  
for i in range(6,0,-1):  
 for j in range(0, i - 1):  
 print("\* ", end="")  
 print(" ")

7.leap year

def CheckLeap(Year):  
 if((Year % 400 == 0) or  
 (Year % 100 != 0) and  
 (Year % 4 == 0)):  
 print("Given Year is a leap Year");  
 else:  
 print ("Given Year is not a leap Year")  
Year = int(input("Enter the number: "))  
CheckLeap(Year)

8.prime no.

number = int(input("Enter any number:"))  
if number>1:  
 for i in range(2,number):  
 if (number%i)==0:  
 print(number, "is not prime number")  
 break  
 else:  
 print(number, "is prime number")

9. find Area in python

PI = 3.14  
r = float(input("Enter the radius of a circle:"))  
area = PI \* r \* r  
print("Area of a circle = %.2f" %area)

10.reverse a list

a=[5,"ram",10,"ravi",3]  
a.reverse()  
print(a)

11. Program to find the sum of all elements in a list

numbers = [6, 5, 54, 8, 4, 2, 5, 6, 21]  
sum = 0  
for i in numbers:  
 sum = sum + i  
print("The sum is", sum)

12.Average of list elements

numbers = [6, 5, 3, 8, 4, 2, 5, 6, 11]  
sum = 0  
for i in numbers:  
 sum = sum + i  
 avg = sum/len(numbers)  
  
print("The average is", avg)

#Max of list

numbers = [6, 66, 3, 8, 65, 2, 5, 6, 11]  
x = max(numbers)  
print(x)

#Min of list

numbers = [6, 0, 3, 8, 4, 2, 5, -4, 1]  
x = min(numbers)  
print(x)

#13. Write a Python program to create a dictionary grouping a sequence of key-value pairs into a dictionary of lists.

#a. Original list:  
#[('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]  
#Grouping a sequence of key-value pairs into a dictionary of lists:  
#{'yellow': [1, 3], 'blue': [2, 4], 'red': [1]}  
  
def grouping\_dictionary(l):  
 result = {}  
 for k, v in l:  
 result.setdefault(k, []).append(v)  
 return result  
colors = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]  
print("Original list:")  
print(colors)  
print("\nGrouping a sequence of key-value pairs into a dictionary of lists:")  
print(grouping\_dictionary(colors))

#14. Write a Python program to convert more than one list to nested dictionary.

#a. Original strings:  
#['S001', 'S002', 'S003', 'S004']  
#['Adina Park', 'Leyton Marsh', 'Duncan Boyle', 'Saim Richards']  
#[85, 98, 89, 92]  
#Nested dictionary:  
#[{'S001': {'Adina Park': 85}}, {'S002': {'Leyton Marsh': 98}}, {'S003': {'Duncan Boyle': 89}}, {'S004': {'Saim Richards': 92}}]  
def nested\_dictionary(l1, l2, l3):  
 result = [{x: {y: z}} for (x, y, z) in zip(l1, l2, l3)]  
 return result  
student\_id = ["S001", "S002", "S003", "S004"]  
student\_name = ["Adina Park", "Leyton Marsh", "Duncan Boyle", "Saim Richards"]  
student\_grade = [85, 98, 89, 92]  
print("Original strings:")  
print(student\_id)  
print(student\_name)  
print(student\_grade)  
print("\nNested dictionary:")  
ch = 'a'  
print(nested\_dictionary(student\_id, student\_name, student\_grade))

#15. Python program to check if a set is a subset of another set.

print("Check if a set is a subset of another set, using comparison operators and issubset():\n")  
setx = set(["apple", "mango"])  
sety = set(["mango", "orange"])  
setz = set(["mango"])  
print("x: ",setx)  
print("y: ",sety)  
print("z: ",setz,"\n")  
print("If x is subset of y")  
print(setx <= sety)  
print(setx.issubset(sety))  
print("If y is subset of x")  
print(sety <= setx)  
print(sety.issubset(setx))  
print("\nIf y is subset of z")  
print(sety <= setz)  
print(sety.issubset(setz))  
print("If z is subset of y")  
print(setz <= sety)  
print(setz.issubset(sety))

#16. Write a Python program to create a symmetric difference and set difference

setc1 = set(["green", "blue"])  
setc2 = set(["blue", "yellow"])  
print("Original sets:")  
print(setc1)  
print(setc2)  
r1 = setc1.symmetric\_difference(setc2)  
print("\nSymmetric difference of setc1 - setc2:")  
print(r1)  
r2 = setc2.symmetric\_difference(setc1)  
print("\nSymmetric difference of setc2 - setc1:")  
print(r2)  
setn1 = set([1, 1, 2, 3, 4, 5])  
setn2 = set([1, 5, 6, 7, 8, 9])  
print("\nOriginal sets:")  
print(setn1)  
print(setn2)  
r1 = setn1.symmetric\_difference(setn2)  
print("\nSymmetric difference of setn1 - setn2:")  
print(r1)  
r2 = setn2.symmetric\_difference(setn1)  
print("\nSymmetric difference of setn2 - setn1:")  
print(r2)

#17. Write a Python program to remove an empty tuple(s) from a list of tuples.

#a. Sample data: [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]  
#Expected output: [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']  
L = [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]  
L = [t for t in L if t]  
print(L)

#18. Write a Python program to calculate the average value of the numbers in a given tuple of tuples.

#a. Original Tuple:  
#((10, 10, 10, 12), (30, 45, 56, 45), (81, 80, 39, 32), (1, 2, 3, 4))  
#Average value of the numbers of the said tuple of tuples:  
#[30.5, 34.25, 27.0, 23.25]  
def average\_tuple(nums):  
 result = [sum(x) / len(x) for x in zip(\*nums)]  
 return result  
  
nums = ((10, 10, 10, 12), (30, 45, 56, 45), (81, 80, 39, 32), (1, 2, 3, 4))  
print ("Original Tuple: ")  
print(nums)  
print("\nAverage value of the numbers of the said tuple of tuples:\n",average\_tuple(nums))  
  
nums = ((1, 1, -5), (30, -15, 56), (81, -60, -39), (-10, 2, 3))  
print ("\nOriginal Tuple: ")  
print(nums)  
print("\nAverage value of the numbers of the said tuple of tuples:\n",average\_tuple(nums))

#19. Write a Python program to check the validity of a password (input from users).

*"""  
Validation :  
At least 1 letter between [a-z] and 1 letter between [A-Z].  
At least 1 number between [0-9].  
At least 1 character from [$#@].  
Minimum length 6 characters.  
Maximum length 16 characters.  
"""*import re  
p= input("Input your password")  
x = True  
while x:  
 if (len(p)<6 or len(p)>12):  
 break  
 elif not re.search("[a-z]",p):  
 break  
 elif not re.search("[0-9]",p):  
 break  
 elif not re.search("[A-Z]",p):  
 break  
 elif not re.search("[$#@]",p):  
 break  
 elif re.search("\s",p):  
 break  
 else:  
 print("Valid Password")  
 x=False  
 break  
  
if x:  
 print("Not a Valid Password")