1. [Palindrome Program In Python](https://www.edureka.co/blog/python-programs/" \l "1)

string=input((**"Enter a letter:"**))  
**if**(string==string[::-1]):  
 print(**"The letter is a palindrome"**)  
**else**:  
 print(**"The letter is not a palindrome"**)

1. [Factorial Program In Python](https://www.edureka.co/blog/python-programs/#2)

num = int(input(**"Enter a number: "**))  
  
factorial = 1  
**if** num < 0:  
 print(**"Sorry, factorial does not exist for negative numbers"**)  
**elif** num == 0:  
 print(**"The factorial of 0 is 1"**)  
**else**:  
 **for** i **in** range(1,num + 1):  
 factorial = factorial\*i  
 print(**"The factorial of"**,num,**"is"**,factorial)

[3.Fibonacci Series Program](https://www.edureka.co/blog/python-programs/#3)

a = int(input(**'enter the first element'**))  
b = int(input(**'enter the second element'**))  
n = int(input(**'enter the number of elements '**))  
print(a, b, end=**" "**)  
  
**while** n - 2:  
 c = a + b  
 a = b  
 b = c  
 print(c, end=**" "**)  
 n = n – 1

4.[Armstrong Number Program In Python](https://www.edureka.co/blog/python-programs/#4)

num = int(input(**"Enter a number: "**))  
sum = 0  
temp = num  
**while** temp > 0:  
 digit = temp % 10  
 sum += digit \*\* 3  
 temp //= 10  
  
*# display the result***if** num == sum:

print(num,**"is an Armstrong number"**)  
**else**:  
 print(num,**"is not an Armstrong number"**)

5.[Calculator Program](https://www.edureka.co/blog/python-programs/#5)

**def** add(a, b):  
 **return** a + b  
  
  
**def** sub(a, b):  
 **return** a - b  
  
  
**def** prod(a, b):  
 **return** a \* b  
  
  
**def** div(a, b):  
 **return** a / b  
  
  
**def** si(p, r, t):  
 **return** (p \* r \* t) / 100  
  
**def** sqr(num):  
 **return** num \*\* 2  
  
  
**def** sqrt(num):  
 **return** num \*\* 0.5  
  
  
print(sqr(10))  
*# to add two numbers, similarly you can use other functions for other operations.*

6.[Patterns Program In Python](https://www.edureka.co/blog/python-programs/#6)

n = int(input(**"Enter the number of rows"**))  
  
**for** i **in** range(0, n):  
  
 **for** j **in** range(0, i + 1):  
  
 print(**"\* "**, end=**""**)  
[Leap Year Program](https://www.edureka.co/blog/python-programs/#7)  
 print()

year = int(input(**'enter year'**))  
**if** year % 400 == 0:  
 print(**'it is a leap year'**)  
**elif** year % 4 == 0:  
 print(**'it is a leap year'**)  
**elif** year % 100 == 0:  
 print(**'not a leap year'**)  
**else**:  
 print(**'not a leap year'**)

8.[Prime Number Program In Python](https://www.edureka.co/blog/python-programs/#8)

num = int(input(**'enter number'**))  
**for** i **in** range(2, num):  
 **if** num% i == 0:  
 print(**'not a prime number'**)  
 **break  
else**:  
 print(**'prime number'**)

9. [Program To Find Area In Python](https://www.edureka.co/blog/python-programs/#9)

**import** math  
  
pi = math.pi  
  
  
**def** circle(radius):  
 **return** pi \* radius \*\* 2  
  
  
**def** cube(side):  
 **return** side \*\* 3  
  
  
**def** cylinder(radius, height):  
 **return** 2 \* pi \* radius + 2 \* pi \* height  
  
  
**def** sphere(radius):  
 **return** 2 \* pi \* (radius \*\* 2)  
  
  
print(circle(10))

10.[Program To Reverse A List](https://www.edureka.co/blog/python-programs/#10)

a **=** [1,2,3,4,5,6,7,8,9,10]

print(a[: : **-**1])

list1 = [11, 5, 17, 18, 23]

11.Program to find sum of all elements in list

total = sum(list1)  
print(**"Sum of all elements in given list: "**, total)

12.Find average, max, min of list elements

somelist = [1, 12, 2, 53, 23, 6, 17]  
max\_value = max(somelist)  
min\_value = min(somelist)  
avg\_value = 0 **if** len(somelist) == 0 **else** sum(somelist) / len(somelist)  
print(avg\_value)  
print(max\_value)  
print(min\_value)

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")