

In [3]: *#Handling input and output*

```
if __name__ == "__main__":
    name=input("Enter the name")
    print("Welcome",name)
    num=int(input("Enter the number"))
    print(num)
    #out put (display the data)
    print("Python")
    print('P','y','t','h','o','n',sep="*")
    print("Welcome python",end="#")
    print("Happy learning")
```

Enter the namekeert higa

Welcome keert higa

Enter the number100

100

Python

P*y*t*h*o*n

Welcome python#Happy learning

In [2]: *#looping constructs*

```
if __name__ == "__main__":
    #while loop
    sum=0
    n=1
    while n <= 10:
        sum +=n
        n +=1
        print(sum)
    #while
    c=0
    while c<3:
        c +=1
        print("learn python")
    else:
        print("Completed the python course")
    #single statement while loop
    v=0
    #while v == 0: print("good at python")
    #for loop
    n=4
```

```
for i in range(0,n):
    print(i)
    st=['welcome','python','think','good']
```

```

for i in st:
    print(i)

list=['p','y','t','h','o','n']
for i in range(len(list)):
    print(list[i])
else:
    print("inside the for loop")

#inside the for loop with condition
for i in range(1,10):
    if i%2 == 0:
        print("Even")
    else:
        print("Odd")

#break statement
for i in 'python':
    if i == 'y' or i == 'n':
        break
    print("current letter",i)
#empty loop
for i in range(10):
    pass
    print("last",i)
#iterator
flowers=['jasmin','lily','lotus']
obj=iter(flowers)
while True:
    try:
        fruits=next(obj)
        print(fruits)
    except StopIteration:
        break

```

55

learn python
learn python
learn python

Completed the python course

0
1
2

```
3
welcome
python
think
good
p
y
t
h
o
n
inside the for loop
Odd
Even
Odd
Even
Odd
Even
Odd
Even
Odd
current letter y
last 9
jasmin
lily
lotus
```

In [4]: *#Array, list, set, tuple and dictionary*

```
l = []
l.append(1)
l.append(10)
print('list', l)
l.pop()
print('after pop', l)
s = set()
s.add(1)
s.add(10)
print('set', s)
s.remove(10)
print('after remove the element:', s)

t = tuple(l)
print("tuple", t)
d = dict()
d[1] = 'first'
d[10] = 'success'
```

```

print('dictionary',d)
del d[1]
print('dictionary',d)

list [1, 10]
after pop [1]
set {1, 10}
after remove the element: {1}
tuple (1,)
dictionary {1: 'first', 10: 'success'}
dictionary {10: 'success'}

```

In [11]: *#modules and functions*

```

#module
from random import *
from math import sqrt, factorial, sin, cos, radians
print(sqrt(25))
print(factorial(10))
print(dir(random))
print(sin(9))
print(cos(10))
print(radians(11))
print(random()*12)
print(random())
print(randint(0,5))
l=[1,2,3,4,5]
print(choice(l))
import time
print(time.time())
#inbuild functions
print('maximum',max(l))
print('minimum',min(l))
#user defined function
def add(a,b):
    return a+b
print(add(2,5))
#nested function
def mul(a,b):

```

```

def division(a,b):
    div = a//b
    print(div)
    division(a,b)
    return a*b

```

```

print(mul(5,3))

5.0
3628800
['__call__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__',
__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__self__', '__setattr__', '__sizeof__',
__str__', '__subclasshook__', '__text_signature__']
0.4121184852417566
- 0.8390715290764524
0.19198621771937624
6.738488616335607
0.6838635439235546
0
4
1616664938.6984787
maximum 5
minimum 1
7
1
15

```

In [24]: #File handling

```

f = open('keert higa.txt', 'w')
f.write('welcome to the python programming')
f.close()
f = open('keert higa.txt', 'r')
print(f.read())
for i in f:
    print(i)
f.close()
f = open('keert higa.txt', 'r')
print(f.read(5))
f.close()
f = open('keert higa.txt', 'a')

```

```

f.write(" i am interesting to learn new things")
f.close()
with open('keert higa.txt') as f:
    data = f.read()
    print(data)
f.close()
with open('keert higa.txt', 'w') as f:
    f.write('Hello python!!!!!!!')

```

```

f.close()
with open('keert higa.txt', 'r') as file:
    data = file.readlines()
    for i in data:
        print(i.split())
f.close()

```

welcome to the python programming

welco

welcome to the python programming i am interesting to learn new things

['Hello', 'python!!!!!!!']

In [1]: *#Exception handling*

```

def zerodiv():
    try:
        a = 5
        if a < 10:
            b = a/(a-5)
            print("Values of b=",b)
        except (ZeroDivisionError, NameError):
            print("The error occurred")
    def final():
        try:
            c = ((2.0+3.0)/(2.0-3.0))
        except ZeroDivisionError:
            print("not possible to divide the zero")
        finally:
            print("This must excuted always!")

```

```

def elseclas():
    try:
        c = ((2.0+3.0)/(2.0-3.0))
    except ZeroDivisionError:
        print("not possible to divide the zero")
    else:

```

```

print(c)
def raising_error():
    x=-1
    if x<0:
        raise Exception("Sor ry, no number s below zer o")
l=[1,2,3]
try:
    print('f ir st element: %d'%l[0])
    print("f iv th element: %d"%l[5])

```

```
except IndexError:
    print("The error occurred in index")
```

```
zerodiv()
elseclas()
final()
raising_error()
```

```
first element:1
The error occurred in index
The error occurred
- 5.0
This must excuted always!
```

In [2]: *#find second highest value from list*

```
if __name__ == "__main__":
    arr = list(map(int, input().split()))
    #arr.sort()
    #print(arr[-2])
    res = list()
    for i in arr:
        if i not in res:
            res.append(i)

    res.sort()
    print(res[-2])
```

```
2 4 10 28 4 28
10
```

In []: *#find the special character*

In [8]: **def** special_char(s):

```
    d = alpha = lw = uw = sp = 0
    for i in s:
        if i >= '0' and i <= '9':
            #print(i)
            d += 1
        elif (i >= 'a' and i <= 'z') or (i >= 'A' and i <= 'Z'):
            if (i >= 'a' and i <= 'z'):
                alpha += 1
            lw += 1
        else:
```

```

alpha += 1
uw += 1
else:
if i != ' ':
sp += 1

print("Digit:",d)
print("Alphabet:",alpha)
print("Special character:",sp)
print("Lower case:",lw)
print("Upper case:",uw)

if __name__ == "__main__":
s = list(input())
special_char(s)

```

Sat hyabama 2019 @
Digit: 4
Alphabet: 10
Special character: 1
Lower case: 9
Upper case: 1

In []: '''

6. Print of the String "Welcome". Matrix size must be N X M. (N is an odd natural number, and M is 3 times N.) The design should have 'WELCOME' written in the center. The design pattern should only use |, . and - characters. Input: N = 7, M = 21

Output:

```

-----|.-----
-----|.|.|.-----
---|.|.|.|.|.---
-----welcome-----
---|.|.|.|.|.---

-----|.|.|.|.-----
-----|.-----

'''

```

In [9]: **def** print_pattern(N,M):

```

for i in range(1,N,2):
print((i * "|.|.").center(M, "- "))
print("WELCOME".center(M,"- "))
for i in range(N-2,-1,-2):
print((i * "|.|.").center(M, "- "))

```

```

if __name__ == "__main__":
    n = int(input("Enter the row:"))
    m = int(input("Enter the column"))
    print_pattern(n,m)

```

```

Enter the row:7
Enter the column:21
-----|.-----
-----|.|.|.-----
---|.|.|.|.|.|.---
-----WELCOME-----
---|.|.|.|.|.|.---
-----|.|.|.-----
-----|.-----

```

In []: '''

```

7. Consider a function  $f(X) = X^3$ . Input is 'N' list. Each list contains 'M' elements.
From the list, find the maximum element. Compute:  $S = (f(X_1) + f(X_2) + f(X_3) + \dots + f(X_N)) \text{ Modulo } Z$ 
Input:
N = 3
Z = 1000
N1 = 2 5 1
N2 = 1 2 4 6 9
N3 = 10 9 11 4 5
Procedure:
maxn1 = 5
maxn2 = 9
maxn3 = 11
 $S = ((\text{maxn1})^3 + (\text{maxn2})^3 + (\text{maxn3})^3) \% Z$ 

```

Output:

185

'''

```

In [1]: if __name__ == "__main__":
        N = int(input("Enter the N"))
        Z = int(input("Enter the Z"))
        res = list()
        for i in range(N):
            n = list(map(int, input().split()))
            res.append(max(n)**3)
        print(res)

```

```
s = 0
for i in res:
    s += i
```

```
print(s)
r = s % Z
print(r)
```

```
Enter the N3
Enter the Z1000
2 5 1
1 2 4 6 9
10 9 11 4 5
[125, 729, 1331]
2185
185
```

In []: '''

8. Validate the Credit numbers based on the following conditions:

- Begins with 4, 5, or 6
- Contain exactly 16 digits
- Contains only numbers (0 to 9)
- For every 4 digits a hyphen (-) may be included (not mandatory). No other special character permitted.
- Must not have 4 or more consecutive same digits.

Input & Output:

```
4253625879615786 Valid
44244244424442444 Valid
```

```
5122- 2368- 7954- 3214 Valid
42536258796157867 Invalid
4424444424442444 Invalid
5122- 2368- 7954 - 3214 Invalid
44244x4424442444 Invalid
0525362587961578 Invalid
61234- 567- 8912- 3456 Invalid
```

'''

```
In [1]: def consec(n):
        s = list(n)
```

```

for i in range(len(s)-3):
    si = s[i]
    if s[i+1] == si and s[i+2] == si and s[i+3] == si:
        return False
    return True

```

```

def check_valid(n):
    s = list(n)
    l = len(s)
    if not (s[0] == '4' or s[0] == '5' or s[0] == '6'):
        print("Start", s[0])
        return False

```

```

if '-' in s:
    print("hyphen")
    d = n.split('-')
    for i in d:
        if len(i) != 4 or int(i) > 9999 or i == '':
            return False
    else:
        if l < 16 or l > 16:
            #print('len')
            return False
        for i in s:
            if not (i >= '0' and i <= '9'):
                print("greater")
                return False

```

```

if (i >= 'a' and i <= 'z') or (i >= 'A' and i <= 'Z'):
    print("Not alpha")
    return False
if not consec(n):
    print('consec')
    return False

```

```

return True

```

```

if __name__ == "__main__":
    n = input()
    if check_valid(n):

```

```
print("Valid")
else:
print("Invalid")
```

```
425362587961.786
greater
Invalid
```

In [40]: **import** re

```
#sample_list=['3123456789123456','9123- 4567- 8912- 3456','5123456789123456','7123 - 4567- 8912 - 3456','44244x4424442444']
pattern = '^[973][0-9]{15}|[973][0-9]{3}-[0-9]{4}-[0-9]{4}-[0-9]{4}$'
eachnumber = input()
```

```
result = re.match(pattern, eachnumber)
if result:
print(eachnumber+"- > "+"Valid card")
else:
print(eachnumber+"- > "+"Invalid card")
```

```
9123- 4567- 8912- 3456
9123- 4567- 8912- 3456- >Valid card
```