Course Documentation
Day 1 to Day 23

Day-1

File: day_1.py

```
a=100
print(a)
```

Day-2

File: Day_2.py

```
import keyword
print(keyword.kwlist)

i = 0o1234
print(i)
```

File: Day_2_Task_Solution.txt

Integer (int) – Questions

1. What data type is used to store whole numbers?
INT

2. Write an example of an int variable declaration.
a=10 , b=-10 , c =0 , d =0b101 , e = 0o12345678 , f = 0x0123456789abcdefg , a = int()

3. Can an int store a decimal number? (Yes/No)
NO

4. Which of the following is an integer: 5, 3.2, true?
5

5. Write a variable to store the number of students in a class.
students_in_class=50;

Float – Questions

1. What data type is used to store decimal numbers?
Float

2. Write an example of a float variable declaration.
a = float() , a = 3.2 , a=3e1

3. Can a float store whole numbers? (Yes/No)
Yes

4. Which of the following is a float: 10, 4.5, false?
4.5

5. Write a variable to store the price of an item.
Price_Of_Item = 30.0

Boolean (bool) – Questions

1. What data type stores true or false values?
bool

2. Write an example of a bool variable declaration.
A = True , B = false

3. How many values can a bool data type have?
2

4. Which of the following is a boolean value: 1, true, 2.5?
True

5. Write a variable to check whether a student has passed an exam.
Exam_Passes = True

File: Day_2_task.txt

Integer (int) – Questions

1. What data type is used to store whole numbers?
2. Write an example of an int variable declaration.
3. Can an int store a decimal number? (Yes/No)
4. Which of the following is an integer: 5, 3.2, true?
5. Write a variable to store the number of students in a class.

Float – Questions

1. What data type is used to store decimal numbers?
2. Write an example of a float variable declaration.
3. Can a float store whole numbers? (Yes/No)
4. Which of the following is a float: 10, 4.5, false?
5. Write a variable to store the price of an item.

Boolean (bool) – Questions

1. What data type stores true or false values?
2. Write an example of a bool variable declaration.
3. How many values can a bool data type have?
4. Which of the following is a boolean value: 1, true, 2.5?
5. Write a variable to check whether a student has passed an exam.

File: day_2.txt

today we installed python software in our systems, we use vs code to run codes.

Data type :
a data type defines the value of the variabale and what is stored inside the varibale and what operations cans be performed on the particular varibale
Types of varibales
1.primitive : stores only one value.
int = +ve , -ve , 0's , binary(0b1,0b101=5) , octal(0o01234567) , hexadecimal(0x012345679abcdef), a=int().
float = 1.2,-1, 2, 2.5e3(2.5 * 10**3) , a=float(), 2E3(2 * e**3).
complex
bool = True=1 , False=0
str
2.non-primitive : can store upto many values in a single vairable.

use type(variable_name) -> to find the daa type of a variable

Day-3

File: Day_3.py

```
"Complex"
l=2+3j
print(l.real)
print(l.imag)

m=3+2j
print(l*m)

"strings"

a = "this is a \
   string" #error
b= '''This is a multi-line
string example.'''

print(b)
print("----------------------------")
print([1,2,3])
```

File: Day_3.txt

Complex Data Type
Mixture of a real and a imaginary number for example "2+3j" , "2-3j" , "3j" ,
We can add , sub , divide , multiply

We can just print the real and imaginary part seperately
l=2+3j
print(l.real) #2.0
print(l.imag) #3.0

imaginary_value = j**2 = -1

String Data Type
It is a immutable data type
it is stored between quotation marks
can also be ' ' ," " ,""" """
iterable
slicing
allows duplicates
examples - l = 'vinay'

if you want to store any value in multiple lines without errors we must adn wshould use thirple quotation

example a = '''This is
vinay''' # no error

Non-Primitive Data Types
1.LIST
Stores Multiple Values In a Single Variable. -Hetrogenous
Values can be changed - Mutable
Can add New values
Ordered
Can Store Multiple DataTypes - Hetrogenous
Supports iteration
Supports slicing
allows duplicates
use .append(value) to add new values at the end # k.append(10) 10 is a value not a index position
use .remove(value) to remove a value # k.remove(20) 20 is a value not a index position
Examples ; k = [10,'string',3e1,2+3j,True]
use .insert(index, value) Inserts a value at a specific index position.
use .sort() Sorts the list in ascending order.
use .sort(reverse=True) Sorts the list in descending order.
use .index(10) Returns the index position of the first occurrence of the value.
use .count(Value) Counts how many times a value appears in the list.
use a =b.copy() Creates a shallow copy of the list (a new list with same elements).
use .clear() Removes all elements from the list and makes it empty.
use .count(value) how many times a values are repeated.
use fist_list.extend(second_list)

sum(variable) Adds all the numeric values in a list.
min(variable) Returns the smallest value in the list.
max(variable) Returns the largest value in the list.

2.NESTED LIST
A list inside a list.
j=[10,20,[100,300],[1000,3000]]
j[2][0] #100

Day-4

File: Day_4.py

```
"Tuple DataType"

l = ()
m = tuple()
n = 1, 2, 3, 4, 5
o = 'a',
t = (10,'python',3.2,True,2+3j,[10,20,30]) #tuple with different data types

print(type(l))  #tuple
print(type(m))  #tuple
print(type(n))  #tuple
print(type(o))  #tuple
print(type(t))  #tuple
print(t[1:4])  #slicing
print(t[-1])   #accessing last element

"Set DataType"

k = {20}
print(type(k))  #set

k={'vinay','vinay','kumar',20,30,20.5,30} #set with different data types and duplica
te values
print(k)  #duplicates will be removed


"Dictionary DataType"
d = {}
d1 = dict()
d2 = {'name':'vinay','age':22,'course':'python'}
print(d2['name']) #accessing value using key
d2['age'] = 23  #modifying value
```

File: Day_4.txt

Tuple DataType

Tuple is immutable where the values cant be updated or removed or added.
Supports indexing and slicing.
can have duplicates

We can declare tuple in the following ways.
j=()
k=tuple()

but is we declaration
j = (20) then this is a int not a tuple () those brackets are not very important , multiple values are those which matter in tuple.
j = (20,) then this is a tuple data type.

Tuples allow Hetrogenous values like a LIST
t = (10,'python',3.2,True,2+3j,[10,20,30]) #tuple

SET DataType
It doesnt allow duplicate values unlike list and tuple, declared in a flow brackets
it is Mutable we can add values after declaration too.
it does not store values in an order.
Hetrogenous
A set inside a set is not allowed i.e nested sets are not possible
.add()
.remove()

DICT DataType

keypair values
h = {} #dict
d={
'kumar' : 1234567890 ,
'sai' : 0987654321
}
print(d['sai'])

no indexing positions , we only use keys.
unordered
keys must always be immutable,which cannot be changed
k.update({'name':20}) #with using method
d['age']=20 #without usign a method

d.pop('name')

Day-5

File: Day_5.py

```
"Nested Dictionaries in Python"
d={'kumar':{'telugu':85,'hindi':90,'english':88}}
print(d['kumar'])
print(d['kumar']['hindi'])  #accessing nested value
d['kumar']['maths'] = 92  #adding new key-value pair in nested dictionary
print(d['kumar'])

d1 = {'name':'kumar','age':18,'name':'vinay'}  #duplicate keys
print(d1)  #last assignment will be considered

for x in d1:
    print(x)
print("-----")

for x in d1.values():
    print(x)
print("-----")

for x,y in d1.items():
    print(x,y)
print("-----")

print(len(d1))  #length of dictionary

print('----------')

"Operations"

"Operations on Sets"
s1 = {10,20,30,40}
s2 = {30,40,50,60}
for x in s2:
    s1.add(x)  #union operation
print(s1)


"Operations on Tuples"
t1 = (10,20,30)
t2 = (40,50,60)
t3 = t1 + t2  #concatenation
print(t3)

"Operations on Dictionaries"
d3 = {'a':10,'b':20,'c':30}
d4 = {'d':40,'e':50}

for k,v in d4.items():
    d3[k] = v  #merging two dictionaries
```

File: Day_5.txt

OPERATORS
Special symbol used to perform operation on variable and values.

1.Arithemetic Operators
+ , - , * , / , // , % , **

2. Comparison (Relational) Operators

3. Assignment Operators

4. Logical Operators

and many more

Day-6

File: Day_6.py

```
"Arithmetic Operators"

# Addition in Integers
a = 10
b = 5
print(a + b)          # 15
print("----------------")

# Addition in strings
str1 = "Hello, "
str2 = "World!"
print(str1 + str2)    # Hello, World!
print("----------------")

# Addition in Lists
list1 = [1, 2, 3]
list2 = [4, 5, 6]
print(list1 + list2)  # [1, 2, 3, 4, 5, 6]
print("----------------")

# Addition in Tuples
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
print(tuple1 + tuple2)   # (1, 2, 3, 4, 5, 6)
print("----------------")

# Addition in Sets
set1 = {1, 2, 3}
set2 = {4, 5, 6}
print(*set1, *set2)      # 1 2 3 4 5 6  (order may vary)
print("----------------")

# Addition in Dictionaries
dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}
print({**dict1, **dict2})   # {'a': 1, 'b': 2, 'c': 3, 'd': 4}
print("----------------")

# Multiplication in Integers
x = 4
y = 3
z = x * y
print(z)                 # 12
print("----------------")

# Multiplication in Strings and integers
str3 = "Hi! "
n = 3
print(str3 * n)          # Hi! Hi! Hi!
print("----------------")
```

```
# Multiplication in Strings
# str4 = "Na"
# str5 = "Batman!"
# print(str4 * str5)
# print("----------------")   # This will raise an error as multiplication is not de
fined between two strings

# Multiplication in Lists and integers
list3 = [7, 8, 9]
n2 = 2
print(list3 * n2)         # [7, 8, 9, 7, 8, 9]
print("----------------")

# Multiplication in Tuples and integers
tuple3 = (10, 11)
n3 = 3
print(tuple3 * n3)        # (10, 11, 10, 11, 10, 11)
print("----------------")

# Multiplication in Sets and integers
# set3 = {12, 13}
# n4 = 2
# print(set3 * n4)          # This will raise an error as multiplication is not defi
ned for sets
# print("----------------")


# # Multiplication in Dictionaries and integers
# dict3 = {'x': 10}
# n5 = 2
# print(dict3 * n5)         # This will raise an error as multiplication is not defi
ned for dictionaries
# print("----------------")

# Note: Multiplication is not defined for sets and dictionaries in Python, hence tho
se lines will raise errors if uncommented.


#Subtraction in Integers
m = 20
n = 8
print(m - n)             # 12
print("----------------")

# Subtraction in other data types
# str6 = "Hello"
# str7 = "World"
# print(str6 - str7)      # This will raise an error as subtraction is not defined f
or strings

#Division in Integers
p = 15
q = 3
print(p / q)             # 5.0
print("----------------")
```

```
#Floor Division in Integers
r = 17
s = 3
print(r // s)           # 5 this weill give the quotient without decimal
print("----------------")

#Exponentiation in Integers
t = 2
u = 3
print(t ** u)          # 8
print("----------------")


# Modulus in Integers
v = 20
w = 6
print(v % w)          # 2 gives the remainder
print("--------------------------------------------------------------------------------
")

# Modulus in other data types
# str8 = "Hello"
# str9 = "World"
# print(str8 % str9)    # This will raise an error as modulus is not defined for str
ings


"Assignment Operators"

#+=
x1 = 5
x2 = 10
x1 += x2
print(x1)              # 15
print("----------------")

#-=
y1 = 20
y2 = 8
y1 -= y2
print(y1)              # 12
print("----------------")

#*=
z1 = 4
z2 = 3
z1 *= z2
print(z1)              # 12
print("----------------")

#/=
a1 = 15
a2 = 3
a1 /= a2
print(a1)              # 5.0
```

```
print("----------------")

#//=
b1 = 17
b2 = 3
b1 //= b2
print(b1)              # 5
print("----------------")

#**=
b1 = 2
b2 = 3
b1 **= b2
print(b1)              # 8
print("----------------")

#%=
b1 = 20
b2 = 6
b1 %= b2
print(b1)              # 2
print("--------------------------------------------------------------------------------
")


#Membership Operators"
# in
my_list = [1, 2, 3, 4, 5]
res = 3 in my_list
print(res)             # True
print("----------------")

v = 'hello'
res2 = 'z' in v
print(res2)           # False
print("----------------")

# not in
my_dict = {'a': 1, 'b': 2}
res3 = 'c' not in my_dict
print(res3)           # True
print("----------------")

w = 'world'
res4 = 'o' not in w
print(res4)           # False
print("--------------------------------------------------------------------------------
")


"Identity Operators"
# is
a = [1, 2, 3]
b = a
print(a is b)         # True
print("----------------")
```

```
c = 500
d = 500
print(c is d)         # False
print("----------------")

# is not
e = 500
f = 500
print(e is not f)     # True
print("----------------")

g = [4, 5, 6]
h = g
print(g is not h)     # False

i = [10,20,30]
j = [10,20,30]
print(i is not j)     # True
print("--------------------------------------------------------------------------------
")
```

File: Day_6.txt

Today we are learning about Operators

2. Assignment Operators
+=, -=, *=, /=, //=, **=, %=

3. Special Operators
* Membership Operators
in and not in
Check weather a string or a number exists in a string or a set/list/dict(key) respectively.

* Identity Operators
Address from -5 to 255 saves in same address , -6 to - infinity and 256 to infinity dont save in same address
ex a=10
b=10
id(a) #123
id(b) #123
ex a=258
b=258
id(a) #123
id(b) #126

this operators check the address of these datatypes;
IS AND IS NOT

Day-7

File: Day_7.py

```
"Comparison Operators in Python"
a = "ab"
b = 'ba'
print(a == b)        # False
print("------------------------------------------------------------------------")

"Logical Operators"

# and
v1 = 100
v2 = 200
result = v1<v2 and v1!=v2
print(result)          # True
print("----------------")

# or
n=10
if n%2==0 or n%3==0:
    print("Divisible by 2 or 3")   # Divisible by 2 or 3
print("----------------")

# not
a = 100
b = not a>200
print(b)               # True
print("------------------------------------------------------------------------")



"Bitwise Operators"
# &
b1 = 5        # 0101
b2 = 3        # 0011
print(b1 & b2)     # 1  (0001)
print("----------------")

b3 = 8      # 1000
b4 = 6      # 0110
print(b3&b4)    # 0  (0000))
print("----------------")

# |
c1 = 5       # 0101
c2 = 3       # 0011
print(c1 | c2)     # 7  (0111)
print("----------------")

# ^
d1 = 5       # 0101
d2 = 3       # 0011
```

```
print(d1 ^ d2)     # 6  (0110)
print("----------------")

# ~
e1 = 6        # 0000....0110
print(~e1)       # -7 (1111....1001)
print("----------------")

# Left Shift
f1 = 5        # 0000....0101
f2 = 2
print(f1 << f2)    # 20 (0000....10100)
print("----------------")

# Right Shift
g1 = 20       # 0000....10100
g2 = 2
print(g1 >> g2)    # 5  (0000....0101)
print("----------------")

g3 = 5      # 0000....0011
g4 = 2
print(g3 >> g4)    # 0  (0000....0000)
print("--------------------------------------------------------------------------")
```

File: Day_7.txt

Comparision Operators
> , < , >= , <= , == , !=

Only prints the output in boolean i.e True/False

Logical Operators
'and'- only runs when both the conditons are True
'or' - runs when one condition is true among 2 conditons
'not'- converts true to False

Bitwise Operators
&(AND) - multiples two inputs by converting them into binary.
1*1 = 1 , 1*0 = 0 , 0*0 = 0 , 0*1 = 0

|(OR) - add the inputs by converting them in binary.
1+1 = 1 , 1+0 = 1 , 0+1 = 1 , 0+0 = 0

^(XOR) - add the inputs by converting them in binary expect 1 ^ 1 =0.
1+1 = 0 , 1+0 = 1 , 0+1 = 1 , 0+0=0

~(NOT) - add a number to the variable and converts the sign to minus
a = 6 , ~6 = -7

<<(LEFT SHIFT) - move the number to the left with the given number of zeros. || adds the given number of zeros at the end.
5 << 2 = 101 ==> 10100 ==> 16+4 = 20

>>(RIGHT SHIFT) - move that many times to the right side of the binay value.
5>>2 = 00101 ==> 00010 ==> 00001 == 1

Day-8

File: Day_8.py

```
"Conditional Statements in Python"
n = 10
#if condition:
if n > 0:  #
    print("Positive number")
    print("----------------")

#else condition:
if n>0:
    print("Positive number")
else:
    print("Non-positive number")

print("----------------")

#elif condition:
if n > 0:
    print("Positive number")
elif n == 0:
    print("Zero")
else:
    print("Negative number")

print("----------------")



"if a person is eligible to vote or not"
age = 20

if age >= 18:
    print("Eligible to vote")
else:
    print("Not eligible to vote")
print("----------------")


"a char is vowel or consonant"
char = 'a'
if char in 'aeiouAEIOU':
    print(char, "is a vowel")
else:
    print(char, "is a consonant")


"Check if a number is multiple of 3 and 5"
num = 15

if num%3==0 and num%5==0:
    print(num, "is a multiple of both 3 and 5")
```

```
else:
    print(num, "is not a multiple of both 3 and 5")
```

File: Day_8.txt

Conditional Statments
Check the Statments and perform a action according to it.

There are maily 4 types of conditional Statments.
1. if Statment
2. if else Statment
3. if elif Statment
4. nested if Statment

File: Day_8_task.py

```
"1. Check whether a number is positive or negative"
a=10
if a>=0:
    print("Positive Number")
else:
    print("Negative Number")

"2. Check whether a number is even or odd."
b=7
if b%2==0:
    print("Even Number")
else:
    print("Odd Number")

"3. Check whether a number is divisible by 5."
c=20
if c%5==0:
    print("Divisible by 5")
else:
    print("Not Divisible by 5")

"4. Find the greater of two numbers."
d=15
e=25
if d>e:
    print("Greater number is", d)
else:
    print("Greater number is", e)

"5. Find the greatest of three numbers."
f=12
g=9
h=20
```

```
if f>g and f>h:
    print("Greatest number is", f)
elif g>f and g>h:
    print("Greatest number is", g)
else:
    print("Greatest number is", h)

"6. Check whether a year is a leap year."
year=2020
if (year%4==0 and year%100!=0) or (year%400==0):
    print("Leap Year")
else:
    print("Not a Leap Year")

"7. Check whether a person is eligible to vote."
age=18
if age>=18:
    print("Eligible to vote")
else:
    print("Not eligible to vote")

"8. Check whether a character is a vowel or consonant."
char='a'
if char in 'aeiouAEIOU':
    print("Vowel")
else:
    print("Consonant")

"9. Check whether a number is divisible by both 3 and 7."
i=21
if i%3==0 and i%7==0:
    print("Divisible by both 3 and 7")
else:
    print("Not divisible by both 3 and 7")

"10. Check whether a number is single-digit, double-digit, or more."
j=123
if j<10:
    print("Single digit")
elif j<100:
    print("Double digit")
else:
    print("More than double digit")
```

Day-9

File: Day_9.py

```
"Notes"
n=5200
if n>0:
  d1 = n//500
  print("Number of 500 Rs notes:", d1)
  n = n%500

if n>0:
  d2 = n//200
  print("Number of 200 Rs notes:", d2)
  n = n%200

if n>0:
  d3 = n//100
  print("Number of 100 Rs notes:", d3)
  n = n%100

  "Calculator"
num1 = 10
num2 = 5
operation = '+'
if operation == '+':
    print("Addition:", num1 + num2)

elif operation == '-':
    print("Subtraction:", num1 - num2)

elif operation == '*':
    print("Multiplication:", num1 * num2)

elif operation == '/':
    if num2 != 0:
      print("Division:", num1 / num2)
```

Day-10

File: Day_10.py

```
"Loops"
a = 'for_loop'

for i in range(len(a)):
    print(a[i])

s = "while_loop"

for x in s:
    print(x)

t = [1,2,{'a':10, 'b':20},(5,6,7),{'a','b','c'}]

for item in t:
    print(item)

b = {1,2,3,4,5,'hello',6}
for num in b:
    print(num)

c = {'name':'Alice', 'age':30, 'city':'New York'}
for key in c:
    print(key, c[key])
```

File: Day_10.txt

Loops
A loop is used when we know how many times we want to repeat something

syntax:
for variable_name in range(n):
code_body

for variable_name in sequence:
code_body

Examples:
for i in 'python':
print(i)

Day-11

File: Day_11.py

```
"Nested For Loops In Python"

for i in range(1, 4):
    for j in range(1, 4):
        print(f'i = {i}, j = {j}')

print("--------------")

for x in ['A', 'B', 'C']:
    for y in [1, 2, 3]:
        print(f'x = {x}, y = {y}')


print("--------------")


for x in range(20,30):
    count = 0
    for y in range(1,x):
        if x%y == 0:
            count += 1

    if count == 1:
        print(f'{x} is a prime number')

print("--------------")

s = ['python', 'java', 'fullstack' , 'c++']

for x in range(0,len(s)):
    if x%2 == 0:
        print(f'Index: {s[x]}')

print("--------------")

for x in s:
    for y in x:
        if y in 'aeiou':
            print(f'Vowel: {y}')
    print()

print("--------------")

for x in s:
    count = 0
    for y in x:
        if y in 'aeiou':
            count += 1
    if count > 1:
        print(f'Word with at least 2 vowels: {x}')
```

```
print("--------------")

for i in range(100,150):
    y = str(i)[::-1]
    if(i == int(y)):
        print(f'Palindrome number: {i}')

print("--------------")


for i in range(100,1000):
    length = len(str(i))
    a = 0
    for j in str(i):
        a += int(j)**length
    if a == i:
        print(f'Armstrong number: {i}')
```

Day-12

File: Day_12.py

```
"Nested For Loops - Star Patterns"

# for row in range(1,6):
#   for col in range (1,6):
#     print("* ",end="")
#   print()

# for i in range(2,10):
#   for x in range(i):
#     print(x,end="")
#   print()

# for i in range(1,6):
#   for j in range(1,i+1):
#     print(j,end=" ")
#   print()

# for i in range(5):
#   temp = 64
#   for j in range(i+1):
#     temp+=1
#     print(chr(temp),end=" ")
#   print()

# for i in range(6,0,-1):
#   for j in range(i):
#     print(j,end=" ")
#   print()

"While Loop"
# a=0
# while a<10:
#   print(a,end=" ")
#   a+=1

# a=-1
# while a>=-10:
#   print(a,end=" ")
#   a-=1

# a=-10
# while a<=-1:
#   print(a,end=" ")
#   a+=1

# j=121
# temp=j
# sum=0
# while j>0:
#   a=j%10
```

```
#   sum=sum*10+a
#   j=j//10
# print(sum)

# if(temp==sum):
#   print("Palindrome")

j = 23456
sum=0
while j>0:
  d = j%10
  j//=10


  if(d%2==0):
    sum+=d
print(sum)
```

File: Day_12.txt

Nested For loops for star patterns

always use end = "" to print the loop in the same line

for i in range (1,5):
print(*,end="") #****

for i in range (1,5):
print(*,end=" ") #* * * *

While Loop
When we dont know the range but we know the condition we use while loops

Syntax:
initialize_variable
while _condition_:
loop_body
increment

File: Day_12_task.py

```
"Print sum of even,odd , prime digits in a number"

num1 = 23456
sumE=0
while num1>0:
  dig = num1%10
  num1 //=10
  if(dig%2==0):
```

```
    sumE+=dig
print("sum of even numbers",sumE)

num2 = 23456
sumO=0
while num2>0:
  digi = num2 % 10
  num2 //= 10
  if(digi%2!=0):
    sumO+=digi
print("Sum of odd numbers", sumO)

num3 = 23456
sumP = 0
while num3>0:
  digit = num3 % 10
  num3 //= 10

  add = 0
  for i in range (1,digit+1):
    if digit%i==0:
      add+=1
  if add==2:
    sumP+=digit

print(sumP)
```

Day-13

File: Day_13.py

```
# user = input("Enter username ")
# print(user)

pas = "vinay123"
password = input("enter password: ")

while password != pas:
  password=input("Enter correct password: ")
```

File: Day_13.txt

input function
- To take input from the user via keyboard
- Always returns a string , even if the user enters number

File: Day_13_task.py

```
"Print from 1 to 20 numbers"
j=1
while j<=20:
  print(j,end=" ")
  j+=1

print()
print("--------------------")

"Print even numbers between 1-50"
k=1
while k<51:
  if(k%2==0):
    print(k,end=" ")
  k+=1

print()
print("--------------------")

"Print odd nummber from 100-50 in reverse order"
l=100
while l>=50:
  if(l%2!=0):
    print(l,end=" ")
  l-=1
```

```
print()
print("-------------------")

"Print squares from number 1 to 10"
a = 1
while a<=10:
  print(a**2,end=" ")
  a+=1

print()
print("-------------------")

"Find the sum of first n natural numbers using while loop"
b = 10
i=1
sum=0
while i<=b:
  sum+=i
  i+=1
print(sum)

print()
print("-------------------")

"Reverse a give number using while loop"
a= 1234
sum=0
while a>0:
  dig = a % 10
  sum = sum *10 + dig
  a//=10
print(sum)

print()
print("-------------------")

"count the number of digits in a given number"
b=1234
sum=0
while b > 0:
  dig = b %10
  b //=10
  sum+=1
print(sum)

print()
print("-------------------")

"sum of n numbers"
b=1234
sum=0
while b > 0:
  dig = b %10
  b //=10
  sum+= dig
```

```
print(sum)

print()
print("-------------------")

"Multiplication table"
t = 5
a=1

while a <=10:
  print(f"{t} x {a} = {t*a}")
  a+=1

print()
print("-------------------")

"find the factorial of a number "
a=5
sum=1
while a > 0:
  sum *= a
  a-=1
print(sum)

print()
print("-------------------")
```

Day-14

File: Day_14.py

```
# "Palindrome using while loop"
# a = 121
# temp = a
# sum=0

# while a > 0:
#   digit = a % 10
#   sum = sum * 10 + digit
#   a//=10
# if (temp==sum):
#   print("palindrome")
# else:
#   print("Not Palindrome")


# "Between Palindrome"

# for num in range(100,150):
#   temp = num
#   sum=0
#   while num > 0:
#     digit = num % 10
#     sum = sum * 10 + digit
#     num//=10
#   if (temp==sum):
#     print(temp,"is palindrome")
#   else:
#     print(temp,"is Not Palindrome")

# "Fibbinocci Series"
# n =1
# a,b = 0,1
# while n<=10:
#   print(a,end = " ")
#   a,b=b,a+b
#   n+=1

# "Max Digit from a number"
# a = 12934
# temp = 0
# while a > 0:
#   digit = a %10
#   a//=10
#   if (digit > temp):
#     temp = digit
# print (temp)


# "Min Digit From a number"
# b = 129340
```

```
# temp = 10
# while b > 0:
#   digit = b%10
#   b//=10
#   if (digit < temp):
#     temp = digit
# print (temp)

# "Nested While Loop"
# a = 1
# while a <= 5:
#   b=0
#   while b < a:
#     print("*",end = " ")
#     b+=1
#   a+=1
#   print()

# "5 - 8 Tables using nested while loop"
# a= 5
# while a<=8:
#   b = 1
#   while b <=10:
#     print(f"{a} x {b} = {a*b}")
#     b+=1
#   a+=1
#   print()

"printing each alphabet individually"

h = "python"
i = 0
while i < len(h):
  if(h[i] in "aeiouAEIOU"):
    print(h[i])
  i+=1
```

File: Day_14.txt

Control Statments/Jumping Statments:
This changes the flow of the execution

break -> terminates the execution
continue -> skips to the current iternation and goes to the next one
pass -> just a place holder,does nthg

Day-15

File: Day_15.py

```
"String functions"
k = "python"
print(k.upper())

print(ord("z"))
print(chr(65))

diff = (ord("a")-ord("A"))
up = ""
for i in k:
  if ord(i) >= 97 and ord(i) <= 122:
    d = ord(i) - 32
    up += chr(d)
print(up)

k = "pYtHoN"
sum = ""
for i in k:
  if ord(i)>=65 and ord(i)<=90:
    d = ord(i) + 32
    sum+= chr(d)
  elif ord(i) >= 97 and ord(i)<= 122:
    d = ord(i) - 32
    sum += chr(d)
print(sum)


i = "support indexing tables"
count = 0
for x in i:
  count +=1

print(count)
print(i.count("p"))

d = i.split()
print(d)

for x in d:
  print(len(x),end = " ")

for i in k:
  print(i.isupper(),i)
```

File: Day_15.txt

STRING FUNCTIONS

str = 'i am a coder'

UPPER()
-To make all the charaters CAPTIAL WORDS
-Variable_name.upper()

LOWER()
-To make all the charaters SMALL WORDS
-Variable_name.lower()

SWAPCASE()
-To make all the characters to theirs opposite case use
-variable_name.swapcase()

ASCII VALUES
to print the ASCII VALUES of a alphabet
ord(Variable_name) #prints number
chr(Number) #print alphabet

ENDSWITH()
-how to check if the word ends with certain alphabets
-Variable_name.endswith("what_to-check) #k.endswith("a")

STARTSWITH()
-how to check if the word starts with a certain alphabets
-Variable_name.startswith("What_to_check) #k.startswith("S")

COUNT()
-to find the count of the word:
-Variable_name.count("what to search for")

SPLIT()
-to convert a string to a perferct list
-variable_name.split()

ISUPPER()
-Tell us wheater a alphabet is upper
-Variable_name.isupper()

ISLOWER()
-Tells us wheater a alphabet is small
-Variable_name.islower()

Day-16

File: Day_16.py

```
j = "MakesFirstLetter"

s=0

for i in j:
  s+=1
print(s)

n =''
for x in j:
  if x.isupper():
    n += '_' + x
  else:
    n+=x

print(n)

m=""
for a in n:
  m+=a.strip("_")

print(m)

o = ""
for b in n:
  if b.isalpha():
    o+=b
print(o)


j = "letters"
print(j.count("t"))


l1 = [1,2,3]
l2 = [10,20,30]

print(l1+l2)


str = "vinay"
print(str.endswith("ay"))

print(str.capitalize())

f_name = input("Enter First Name:")
print(len(f_name))


str = "$$$$$fabjvyfw$$  VVBNBVCDSDXC$X C$V$"
```

```
print(str.count("$"))
```

File: Day_16.txt

String Functions

str = 'i am a coder'

ENDSWITH()
str.endswith("er") => return TRUE if it really ends with er.

CAPITALIZE()
str.capitalize() => Caps the 1st char

REPLACE()
str.replace(old,new) => replaces all old occerences
ex-
str.replace('a','z') => i zm z coder

FIND()
str.find(word) => gives index when it appears 1st time.
str.find('a') => 2
str.find('k') => -1 because it is not found

COUNT()
str.count("a") => tells how many time a char repeats

Day-17

File: Day_17_task.py

```
# Reverse a string without using slicing or reverse().
str1 = "vinay"
temp = ''
for chr in str1:
  temp = chr + temp
print (temp)
print("-----------------------------")

# Check whether a string is a palindrome.
str2 = "vinaniv"
temp2 = str2[::-1]
if str2 == temp2:
  print("palindrome")
else:
  print("Not Palindrome")
print("-----------------------------")
# Count the frequency of each character in a string.
str3 = "aaaabbbccc"
temp3 = {}
for i in str3:
  if i in temp3:
    temp3[i]+=1
  else:
    temp3[i] = 1
print(temp3)
print("-----------------------------")
# Find the first non-repeating character in a string.
str4 = "aaabbbbdeeef"
temp4 = {}
for chr4 in str4:
  if chr4 in temp4:
    temp4[chr4] +=1
  else:
    temp4[chr4] = 1

for i in str4:
  if temp4[i] == 1:
    print(i)
    break
print("-----------------------------")

# Remove duplicate characters from a string.
str5 = 'aaabbcdddef'
temp5 = ''

for i in str5:
  if i not in temp5:
    temp5 += i
print(temp5)
print("-----------------------------")
```

```
# Find the maximum occurring character in a string.
str6 = "aaaaaabbbbbbbbcccddddddeeeeee"
temp6 ={}
for i in str6:
  if i in temp6:
    temp6[i]+=1
  else:
    temp6[i] = 1

s_temp = 0
char = ''
for j in temp6:
  if temp6[j]>s_temp:
    s_temp = temp6[j]
    char = j
print(char,temp6[char])
print("-----------------------------")

# Count the number of vowels and consonants in a string.
str7 = "aajfnebuvbucvqqckjqbvbqonknassnznadqbipokmsmazlioueyhunt"
vowles = 0
conso = 0
for i in str7:
  if i in "aeiouAEIOU":
    vowles+=1
  else:
    conso +=1
print(vowles,conso)
print("-----------------------------")

# Reverse each word in a string.
str8 = " always follow dreams"
s_str8 = str8.split()

for idx in range(len(s_str8)):
    temp8 = ''
    for ch in s_str8[idx]:
        temp8 = ch + temp8
    s_str8[idx] = temp8

ss_str8 = " ".join(s_str8)
print(ss_str8)
print("-----------------------------")
# Check whether two strings are anagrams.
str9 = "listen"
str10 = 'silent'

temp9= {}
temp10={}
for i in str9:
  if i in temp9:
    temp9[i] +=1
  else:
    temp9[i] = 1
```

```
for i in str10:
  if i in temp10:
    temp10[i] +=1
  else:
    temp10[i] = 1

if temp9 == temp10:
  print("anagram")
else:
  print("Not anagram")
print("-----------------------------")

# Replace spaces with special characters (e.g., @)
str11 = "mknbuvy hbjnbvh gjknobvh g"
r_str11 = str11.replace(" ","@")
print(r_str11)
print("-----------------------------")
```

Day-18

File: Day_18.py

```
# Print the characters in the prime index
str1 = "abcdefghijklmnopqrstuvwxyz"

for i in range(1,len(str1)):
  if i>1:
    for j in range(2,i):
      if i%j==0:
        break
    else:
      print(str1[i],i)

#replace vowels with *

str2 = "zqwaexsrcdtvfygbuhnijmolp"
temp2 = ''
for i in str2:
  if i in "aeiou":
     temp2+= "*"
  else:
     temp2+=i
print(temp2)
print("----------")

str3 = "zqwaexsrcdtvfygbuhnijmolp"
temp3 = ''
for i in str3:
   if i in 'aeiou':
      str3=str3.replace(i,"*")
print(str3)

# check if 2 strings are anagrams
a = 'tea'
b = 'eat'

if sorted(a) == sorted(b):
  print("anagram")

from collections import Counter
print(Counter(a))

temp9= {}
temp10={}
for i in a:
  if i in temp9:
    temp9[i] +=1
  else:
    temp9[i] = 1

for i in b:
  if i in temp10:
```

```
    temp10[i] +=1
  else:
    temp10[i] = 1

if temp9 == temp10:
  print("anagram")
else:
  print("Not anagram")

# reverse each word in a string
k = "reverse each word string".split()

for i in range(len(k)):
  temp3 = ''
  for j in k[i]:
    temp3 = j + temp3
  k[i] = temp3
print(" ".join(k))

k = "cap starting of each start"
temp1 = ""
for x in range(0,len(k)):
  if x == 0:
    temp1+=k[x].upper()
  elif k[x-1] == " ":
    temp1+= k[x].upper()
  else:
    temp1+=k[x]
print(temp1)

# Remove duplicates from list and sort
k =[10,20,30,40,10,20]
d=list(set(k))
d.sort()
print(d)
temp = []
for x in k:
  if x not in temp:
    temp.append(x)

# Sum of int and float from list
j = [10,11,12,3.2,'python',10.2]
temp = 0
for i in j:
  if type(i)==int or type(i)==float:
    temp+= i
print(temp)

# Find minimum element in list
k = [10,11,12,5,2]
temp = k[0]
for i in k:
  if temp > i:
    temp = i
print(temp)
```

```
# Filter even numbers and collect strings
l = [10,20,11,13,'c++','python',3+2j]
temp=[]

for i in l:
  if type(i)==int and i%2==0:
    print(i)
  if type(i)== str:
    temp.append(i)
print(temp)

# Double even numbers in list
j = [10,11,12,13]
temp = []
for i in j:
  if i%2==0:
    temp.append(i*2)
  else:
    temp.append(i)
print(temp)

for i in range(0,len(j)):
  if i%2==0:
    j[i] = j[i]*2
print(j)
```

Day-19

File: Day_19.py

```
# PYTHON STRING METHODS – SMALL EXAMPLES WITH OUTPUTS

# ---------------- CASE CONVERSION ----------------
print("hello".upper())          # HELLO
print("HELLO".lower())          # hello
print("hello world".title())    # Hello World
print("python language".capitalize())  # Python language
print("PyThOn".swapcase())      # pYtHoN
print("HELLO".casefold())       # hello

# ---------------- SEARCH / FIND ----------------
print("banana".find("na"))      # 2
print("banana".rfind("na"))     # 4
print("apple".index("p"))       # 1
print("apple".rindex("p"))      # 2
print("banana".count("a"))      # 3
print("python".startswith("py"))  # True
print("python".endswith("on"))    # True

# ---------------- CHECKING ----------------
print("Hello".isalpha())        # True
print("1234".isdigit())         # True
print("abc123".isalnum())       # True
print("hello".islower())        # True
print("HELLO".isupper())        # True
print("   ".isspace())          # True
print("Hello World".istitle())  # True

# ---------------- MODIFY / REPLACE ----------------
print("hello world".replace("world","python"))  # hello python
print("  hi  ".strip())         # hi
print("  hi".lstrip())          # hi
print("hi  ".rstrip())          # hi
print("unhappy".removeprefix("un"))   # happy
print("testing.py".removesuffix(".py"))  # testing

# ---------------- SPLIT / JOIN ----------------
print("a b c".split())          # ['a', 'b', 'c']
print("a-b-c".rsplit("-",1))    # ['a-b', 'c']
print("hi\nhello".splitlines()) # ['hi', 'hello']
print("-".join(['a','b','c']))  # a-b-c

# ---------------- ALIGN / FORMAT ----------------
print("hi".center(6,"*"))       # **hi**
print("hi".ljust(5,"-"))        # hi---
print("hi".rjust(5,"-"))        # ---hi
print("7".zfill(3))             # 007
print("My age is {}".format(20)) # My age is 20

# ---------------- ENCODING ----------------
```

```
print("hi".encode())            # b'hi'
```

Day-20

File: Day_20.py

```
# Check if list is in ascending order
k = [10,10,12,13,14]
is_ascending = True
for i in range(len(k)-1):
  if k[i+1]<k[i]:
    is_ascending = False
    break

if is_ascending:
  print("Ascending")
else:
  print("Not Ascending")

# Calculate average of list
k = [10,20,30,40]
sum=0
count=0
for i in k:
  sum+=i
  count+=1
print(sum/count)

# Sort list using bubble sort
l = [10,11,12,13,14,9]
for i in range(len(l)):
  for j in range(i+1,len(l)):
    if l[j]<l[i]:
      l[i],l[j]=l[j],l[i]
print(l)

# Find pairs that sum to target value
k = [1,2,3,4,5,6,7,8]
for i in range(len(k)):
  for j in range(i+1,len(k)):
    if k[i]+k[j] == 10:
      print (k[i],k[j])

# Remove duplicates from list
k=[10,11,12,13,10]
temp = []
for i in k:
  if i not in temp:
    temp+=[i]
print(temp)

# Check if list elements are unique
is_unique = True
for i in range(len(k)):
  for j in range(i+1,len(k)):
    if k[i]==k[j]:
```

```
      is_unique = False
      break
if is_unique:
  print("Unique")
else:
  print("Not Unique")

# Find second maximum element
k=[10,12,13,15,15]
temp=[]
for i in range(len(k)):
  for j in range(i+1,len(k)):
    if k[i]>k[j]:
      k[i],k[j]=k[j],k[i]
maxi = k[-1]
tem = 0
for l in k:
  if l>tem and l<maxi:
    tem = l
print(tem)

# Find second minimum element
k = [10,11,12,13,14]

min1 = min2 = float('inf')

for i in k:
    if i < min1:
        min2 = min1
        min1 = i
    elif i < min2 and i != min1:
        min2 = i

print(min2)

# Replace negative numbers with 0
k = [10,-12,-8,-9,13,14]
for i in range(len(k)):
  if k[i] < 0:
    k[i]=0
print(k)

# Sort list with 0s and 1s
k=[1,0,1,0,1,0,1,0]
for i in range(len(k)):
  for j in range(i+1,len(k)):
    if k[i]>k[j]:
      k[i],k[j]=k[j],k[i]
print(k)

# Separate positive and negative numbers
k=[10,-2,-3,11,13,-4]
pos = []
neg = []
for i in k:
  if i<0:
```

```
    neg+=[i]
  else:
    pos+=[i]
print(pos,neg)

# Flatten list of lists
k=[[10,20,30],[100,200,300,400]] #[10,20,30,100,200,300,400]
temp=[]
for i in k:
  for j in i:
    temp+=[j]
print(temp)

# Flatten nested list with mixed types
k=[10,20,30,[100,200,300,400]] #[10,20,30,100,200,300,400]
temp =[]
for i in k:
  if type(i) != int:
    temp.extend(i)
  else:
    temp.append(i)
print(temp)
```

Day-21

File: Day_21.py

```
'set datatype'
a = {1, 2, 3}
b = {3, 4, 5}

print("Initial a:", a)        # Initial a: {1, 2, 3}
print("Initial b:", b)        # Initial b: {3, 4, 5}

# add() → add ONE element
a.add(4)
print("After adding 4 to a:", a)   # {1, 2, 3, 4}

# update() → add MULTIPLE elements
a.update([5, 6])
print("After updating a with [5, 6]:", a)  # {1, 2, 3, 4, 5, 6}

# remove() → remove element (error if not present)
a.remove(2)
print("After removing 2 from a:", a)  # {1, 3, 4, 5, 6}

# discard() → remove element safely (no error)
a.discard(10)
print("After discarding 10 from a:", a)  # {1, 3, 4, 5, 6}

# pop() → remove & return RANDOM element
removed = a.pop()
print("After popping element", removed, "from a:", a)
# remaining elements after pop

# union() → combine both sets (does NOT modify a)
print("Union of a and b:", a.union(b))  # {3, 4, 5, 6}
# other form → print(a | b)

# intersection() → common elements
print("Intersection of a and b:", a.intersection(b))  # {3, 4, 5}
# other form → print(a & b)

# difference() → elements in a but NOT in b
print("Difference of a and b (a - b):", a.difference(b))
# other form → print(a - b)

# symmetric_difference() → elements not common
print("Symmetric difference of a and b:", a.symmetric_difference(b))
# other form → print(a ^ b)

# intersection_update() → keep ONLY common elements
c = a.copy()
c.intersection_update(b)
print("After intersection_update on c:", c)
# other form → c &= b
```

```
# difference_update() → remove b elements from a
d = a.copy()
d.difference_update(b)
print("After difference_update on d:", d)
# other form → d -= b

# symmetric_difference_update() → keep non-common elements
e = a.copy()
e.symmetric_difference_update(b)
print("After symmetric_difference_update on e:", e)
# other form → e ^= b

# copy() → create duplicate set
f = a.copy()
print("Copy of a (f):", f)

# isdisjoint() → check if no common elements
print("Is a disjoint with {10,11}?", a.isdisjoint({10, 11}))  # True

# issubset() → check if all elements exist in a
print("Is {3} subset of a?", {3}.issubset(a))  # True
# other form → print({3} <= a)

# issuperset() → check if a contains all elements
print("Is a superset of {3}?", a.issuperset({3}))  # True
# other form → print(a >= {3})

# clear() → remove all elements
a.clear()
print("After clearing a:", a)  # set()
```

Day-22

File: Day_22.py

```
# ALL PYTHON DICTIONARY METHODS (COPY–PASTE READY)

d = {'a': 1, 'b': 2}

# clear() → removes all key–value pairs
d.clear()                          # {}

# copy() → returns a shallow copy
d2 = d.copy()                      # {'a': 1, 'b': 2}

# fromkeys() → creates dictionary from keys with same value
new_dict = dict.fromkeys(['x','y','z'], 0)   # {'x': 0, 'y': 0, 'z': 0}

# get() → safely get value (no error if key missing)
d.get('a')                         # 1

# items() → returns key–value pairs
d.items()                          # dict_items([('a',1), ('b',2)])

# keys() → returns all keys
d.keys()                           # dict_keys(['a','b'])

# values() → returns all values
d.values()                         # dict_values([1,2])

# pop() → removes key and returns its value
d.pop('a')                         # 1

# popitem() → removes and returns last inserted item
d.popitem()                        # ('b', 2)

# setdefault() → returns value if key exists, else inserts key with value
d.setdefault('c', 3)               # 3

# update() → adds/updates multiple key–value pairs
d.update({'d': 4})                 # {'a':1,'b':2,'d':4}


# -------- IMPORTANT DICTIONARY OPERATIONS --------

# add / update key
d['e'] = 5

# access value
d['e']                             # 5

# delete key
del d['e']

# check key existence
```

```
'e' in d                            # True / False

# length of dictionary
len(d)

# iterate over keys
for k in d:
    print(k)

# iterate over values
for v in d.values():
    print(v)

# iterate over key–value pairs
for k, v in d.items():
    print(k, v)

std_re = [
    {'std_id':101,'std_name':'kumar','std_age':14},
    {'std_id':102,'std_name':'sai','std_age':15},
    {'std_id':103,'std_name':'mohan','std_age':13}
]

for x in std_re:
    if x['std_id']==101:
        for k,v in x.items():
            print(k,'==',v)

l=[]

for i in range(3):
    new={}
    key = input("give new key: ")
    value = int(input("give new value: "))
    new[key] = value
    l.append(new)

print(l)
```

Day-23

File: Day_23.py

```
# Dictionary iteration using items()
s = {'a':10,'b':20}

for x in s.items():
  print(x)

# Create dictionary from list of tuples
h = dict([('a',10),('b',20)])
print(h)

# Reverse dictionary (swap keys and values)
j = {'name':'ajay','mobile':123456789}
temp = {}
for k,v in j.items():
  temp[v] = k

print(temp)

# Create dictionary with word lengths
words = ['apple','banana','kiwi']
temp ={}
tem = {}
for k in words:
  temp[k] = len(k)

for i,j in temp.items():
  tem[j] = i

print(temp)
print(tem)

# Count characters in each word
for k in words:
  count = 0
  for j in k:
    count+=1
  temp[k] = count
print(temp)

# Find divisors of a number
j = 10  #{10:[1,2,5,10]}
temp=[]
for i in range(1,j+1):
  if j%i==0:
    temp.append(i)

tem = {}
tem[j]=temp
print(tem)
```

```
# Find divisors for each element in list
l = [10,20,30]
old={}
for i in l:
  new = []
  for j in range(1,i+1):
    if i%j == 0:
      new.append(j)
  old[i]=new
print(old)

# Create nested dictionary with squares and cubes
keys = {}
for i in range(1,4):
  values = {}
  values['s'] = i**2
  values['c'] = i**3
  keys[i] = values

print(keys)

# Classify numbers as even/odd
k=[10,22,21,23]
d = {}
for i in k:
  if i%2==0:
    d[i]='even'
  else:
    d[i]='odd'
print(d)

# Zip keys and values to create dictionary
key=['name','age','city']
value=['john',25,'new york']

temp ={}
for i in range(len(key)):
    temp[key[i]] = value[i]
print(temp)

for a,b in zip(key,value):
  temp[a]=b
print(temp)

print(dict(zip(key,value)))

# Check prime numbers and store in dictionary
j=[11,12,13,14,17]
temp = {}
for i in j:
  count = 0
  for k in range(1,i):
    if i%k==0:
      count+=1
  if count == 1:
    temp[i]='prime'
```

```
  else:
    temp[i]='not prime'
print(temp)

# Create dictionary of even squares
temp = {}
for i in range(1,11):
  if i%2==0:
    temp[i]=i**2
print(temp)

# Count character frequency in string
d = 'success' #{s:3,u:1,c:2,e:1}
res = {}

for i in d:
  if i not in res:
    res[i]=1
  else:
    res[i]+=1
print(res)
```