lecture : 1

```
# 1.output

print("hello world")
```

```
hello world
```

```
print("hello")
print("world")
```

```
hello
world
```

```
print("hello" ,end="-")
print("world")
```

```
hello-world
```

variable

```
a=5
b=6
print(a+b)
```

```
11
```

```
!pip install pywhatkit
```

```
Requirement already satisfied: pywhatkit in c:\users\jayra\anaconda3\
lib\site-packages (5.4)
Requirement already satisfied: Pillow in c:\users\jayra\anaconda3\lib\
site-packages (from pywhatkit) (10.4.0)
Requirement already satisfied: pyautogui in c:\users\jayra\anaconda3\
lib\site-packages (from pywhatkit) (0.9.54)
Requirement already satisfied: requests in c:\users\jayra\anaconda3\
lib\site-packages (from pywhatkit) (2.32.3)
Requirement already satisfied: wikipedia in c:\users\jayra\anaconda3\
lib\site-packages (from pywhatkit) (1.4.0)
Requirement already satisfied: Flask in c:\users\jayra\anaconda3\lib\
site-packages (from pywhatkit) (3.0.3)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\jayra\
anaconda3\lib\site-packages (from Flask->pywhatkit) (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\jayra\
anaconda3\lib\site-packages (from Flask->pywhatkit) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\jayra\
```

```
anaconda3\lib\site-packages (from Flask->pywhatkit) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\jayra\
anaconda3\lib\site-packages (from Flask->pywhatkit) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\jayra\
anaconda3\lib\site-packages (from Flask->pywhatkit) (1.6.2)
Requirement already satisfied: pymsgbox in c:\users\jayra\anaconda3\
lib\site-packages (from pyautogui->pywhatkit) (1.0.9)
Requirement already satisfied: pytweening>=1.0.4 in c:\users\jayra\
anaconda3\lib\site-packages (from pyautogui->pywhatkit) (1.2.0)
Requirement already satisfied: pyscreeze>=0.1.21 in c:\users\jayra\
anaconda3\lib\site-packages (from pyautogui->pywhatkit) (1.0.1)
Requirement already satisfied: pygetwindow>=0.0.5 in c:\users\jayra\
anaconda3\lib\site-packages (from pyautogui->pywhatkit) (0.0.9)
Requirement already satisfied: mouseinfo in c:\users\jayra\anaconda3\
lib\site-packages (from pyautogui->pywhatkit) (0.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
jayra\anaconda3\lib\site-packages (from requests->pywhatkit) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\jayra\
anaconda3\lib\site-packages (from requests->pywhatkit) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\jayra\
anaconda3\lib\site-packages (from requests->pywhatkit) (1.26.20)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\jayra\
anaconda3\lib\site-packages (from requests->pywhatkit) (2024.12.14)
Requirement already satisfied: beautifulsoup4 in c:\users\jayra\
anaconda3\lib\site-packages (from wikipedia->pywhatkit) (4.12.3)
Requirement already satisfied: colorama in c:\users\jayra\anaconda3\
lib\site-packages (from click>=8.1.3->Flask->pywhatkit) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\jayra\
anaconda3\lib\site-packages (from Jinja2>=3.1.2->Flask->pywhatkit)
(2.1.3)
Requirement already satisfied: pyrect in c:\users\jayra\anaconda3\lib\
site-packages (from pygetwindow>=0.0.5->pyautogui->pywhatkit) (0.2.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\jayra\
anaconda3\lib\site-packages (from beautifulsoup4->wikipedia-
>pywhatkit) (2.5)
Requirement already satisfied: pyperclip in c:\users\jayra\anaconda3\
lib\site-packages (from mouseinfo->pyautogui->pywhatkit) (1.9.0)
```

```
import pywhatkit


pywhatkit.sendwhatmsg("+916299996489",
                      "muh me lega",
                      18, 30)
```

```
In 25195 Seconds WhatsApp will open and after 15 Seconds Message will
be Delivered!
```

```
#dynamic typing :) veriable creation ke time pe data type nahi batana
#static tying :) variable creation ke time pe data type batana


##Dynamic binding :) variable ka data type change ho sakta hai
a=5
print(a)
print(type(a))
a="jay"
print(a)   # yaha pe a variable ka type change ho gya pahle integer
the and then string
print(type(a))

#static binding :) static binding me type change nahi kar sakte hain
matlab ki agar koi variable integer type ka hai to uss pure program me
#same variable ko redeclare nahi kar sakte hain
```

```
5
<class 'int'>
jay
<class 'str'>
```

```
# multiple variable single line me
a,b,c=1,2,3
print(a,b,c)
```

```
1 2 3
```

```
# multiple variable ko same value dena
a=b=c=12
print(a,b,c)
```

```
12 12 12
```

```
# there are total 32 keywords in python
   # identifiers :) name of variables , functions ,list ,tuples ,
class etc
     #rules:)
        #similar as other languages
```

user input

```
input()
```

```
 abc
```

```
'abc'
```

```
input("apna naam bata : ")
```

```
apna naam bata :  jay
```

```
'jay'
```

```
input("roll no. bata") # re baba output me gadbad hai qki input number dale hain but aaya string formate me
```

```
roll no. bata 19
```

```
'19'
```

```
#chalo is gadbad ko thik karte hain but kaise --> type casting re baba
# type casting karte time ek baat jarur dhyam rakhna ki kya wo cast ho sakta hai nahi to gadbad ho jayega re baba
int(input("roll no. bata"))
```

```
roll no. bata 19
```

```
19
```

```
#addition of two number
a=int(input("enter first number"))
b=int(input("enter second number"))
c=a+b
print("the sum of ",a , "&" , b," is ",c)
print(type(a)) # isme type conversion permanent ho jata hai qki conversion ke baad store ho raha hai
```

```
enter first number 10
enter second number 12
```

```
the sum of  10 & 12  is  22
<class 'int'>
```

```
a=(input("enter first number"))
b=(input("enter second number"))
c=int(a)+int(b)
print("the sum of ",a , "&" , b," is ",c)
print(type(a))   # type conversion permanent nahi hua hai qki basically store to string me hi hua hai
```

```
enter first number 10
enter second number 2
```

```
the sum of  10 & 2  is  12
<class 'str'>
```

literals

- row value jo ki variable me store hote hain

- a=10 => a--> variable name , = --> operator ,10 --> literals

```
#integer literals
print('integer')
a=0b1010 #binary literals
print(a)
b=100 #decimal
print(b)
c=0o310 #octal
print(c)
d=0x12c # hexa decimal

#float literals
print('float')
float_1=10.5
float_2 = 1.5e2 # 1.5 *10^2
float_3 = 1.5e-3 # 1.5 *10^-3
print(float_3)


#complex literals
print('complex')

x=3.14j
print(x.real ,x.imag)

#string
print('string')

string="this is python"
multiline_str="""hii
                    i
                    am jay """
print(string)
print(multiline_str)


# boolean  :) true-->1 && false -->0
print('boolean')
a=True+5
print(a)
a=False+8
print(a)

# none
#yadi aap  sure nahi ho ki kisi variable ka kya value du but aap uss
variable ko future me use karna chahate ho to none se declare karke
rakh do
print('none')
k=None
```

```
print(type(k))
print(k)
```

```
integer
10
100
200
float
0.0015
complex
0.0 3.14
string
this is python
hii
                    i
                    am jay
boolean
6
8
none
<class 'NoneType'>
None
```

OPERATORS

- Arithmetic Operator
- Relational Operator
- Logical Operator
- Bitwise Operator
- Assignment operator
- Membership Operator

python me increment and decrement operator nahi hota hai

```
#Arithmetic Operator
print(5+6)

print(5-6)

print(5*6)

print(5/2) #gives exact output  in float format

print(5//2) # integer division(or,floor division)  gives number line
pe left bala

print(5%2)
```

```python
print(3%5) # 3 --> yaha pahle chota hai to uska answer chota digit ho jayega

print(5**2) # power of operator
```

```
11
-1
30
2.5
2
1
25
```

```python
#Relational operator
print(4<5)

print(4>=4)

print(4!=7)
```

```
True
True
True
```

```python
#logical Operator
print(1 or 0)

print(1 and 0)

print(not 1)

print(4 and 6) #
```

```
1
0
False
6
```

```python
#Bitwise operator

#bitwise and
print(2&3) # 2 and 3 ka binary likho and then bitwise and operation lagao

#bitwise or
print(2|3)

#bitwise XOR
print(2^3)

#bitwise not
```

```
print(~3)

#bitwise leftshift       =>2 se devide karte jata hai jitna shift
karna hota hai
print(4<<2)

#bitwise right-shift   => 2 se devide karte jata hai jitna shift karna
hota hai  odd rahata hai to (value - 1) hoke 2 se devide hota hai
print(4>>2)

print(15>>1)



2
3
1
-4
16
1
7

#Assignment Operator
a=2
a+=2  # a=a+2
print(a)

4

#Membership Operator
#in/not in

print('D' in 'Delhi') #--> true

print('l' not in 'Delhi') #--> false
print(1 in [2,3,4,5,6])  #--> false

True
False
True

print(-5//2)

-3

#program -Find the sum of 3 digit number entered by the user

number = int(input("enter 3 digit number"))
sum=0
rem=number%10
sum=sum+rem
```

```
number=number//10
rem=number%10
sum=sum+rem
number=number//10
rem=number%10
sum=sum+rem
number=number//10
print(sum)
```

```
enter 3 digit number 345
```

```
12
```

if else in Python

```
# login program and identification
#email -> jayram.10125@gmail.com
#password -> 1234
email=input("enter email ")
password=input("enter password : ")

if email =="jayram.10125@gmail.com" and password == '1234' :
    print("login successfully")
else :
    print("please enter valid login password")
```

```
enter email  jayram.10125@gmail.com
enter password :  1234
```

```
login successfully
```

```
email=input("enter email ")
password=input("enter password : ")

if email =="jayram.10125@gmail.com" and password == '1234' :
    print("login successfully")
elif email=="jayram.10125@gmail.com" and password !='1234':
    print("incorrect possword")
    password=input("enter password again")
    if(password=="1234"):
        print("login successfully")
    else:
        print("beta tumse nn ho payega")

else :
    print("please enter valid login password")
```

```
enter email  jayram.10125@gmail.com
enter password :  1111
```

```
incorrect possword
```

```
enter password again 123

beta tumse nn ho payega
```

```
#if-else example
# find minimum of three numbers
# Menu driven Program

#min of three
a=int(input("enter first number : "))
b=int(input("enter second number : "))
c=int(input("enter third number : "))

if a<b and a<c :
    print ("smallest is : ",a)
elif b<c : # a agar chota nahi hai to bacha kya b and c
    print("smallest is  : ",b)
else :
    print("smallest is : ",c)
```

```
enter first number :  1
enter second number :  2
enter second number :  3

smallest is :  1
```

```
#Menu driven program(calculator)
fnum=int(input("enter the first number"))
snum=int(input("enter the second number"))
op=input("enter the operation ")

if op=='+' :
    print(fnum+snum)
elif op=="-":
    print(fnum-snum)
elif op=="*":
    print(fnum*snum)
elif op=="/":
    print(fnum/snum)
```

```
enter the first number 5
enter the second number 4
enter the operation  +

9
```

```python
#mini ATM operation
menu=input("""
       Hi! How can I help you
       1. Enter 1 for pin change
       2. Enter 2 for balance check
       3. Enter 3 for withdrawl
       4. Enter 4 for exit
       """)
if menu=="1":
    print("pin change")
elif menu=="2":
    print("balance")
elif menu=="3":
    print("withdrawl")
else:
    print("exit")
```

```
       Hi! How can I help you
       1. Enter 1 for pin change
       2. Enter 2 for balance check
       3.Enter 3 for withdrawl
       4.enter 4 for exit
        4

exit
```

Modules in Python

module is basically file jo ki already likha hua hai

- math
- keywords
- datetime
- random

```python
# math
import mathjqbd # agar galat module ka name daloge to wo error dega
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[22], line 2
      1 # math
----> 2 import mathjqbd
```

```
ModuleNotFoundError: No module named 'mathjqbd'
```

```
import math
```

```
math.factorial(5)
```

```
120
```

```
math.floor(6.8)
```

```
6
```

```
## keyword
print(keyword.kwlist) # bina kewword module ko import kye hi agar kuch
uss module ka access karoge to bhi error aayega but ek baar
# kar dye to bindass ho jao baar baar karne ki naad nahi hai
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[83], line 1
----> 1 print(keyword.kwlist)

NameError: name 'keyword' is not defined
```

```
import keyword
```

```
print(keyword.kwlist)  #sare keyword ko print kar dega
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try',
'while', 'with', 'yield']
```

```
#random
import random
print(random.randint(1,100)) # 1 to 100 ke bich random number generate
karte rahega
```

```
65
```

```
# datetime
import datetime
print(datetime.datetime.now())
```

```
2025-02-12 22:46:40.830089
```

```
# yadi aap check karna chahate hai ki kon kon se module availabe hain
to do it :
help("modules")
```

```
Please wait a moment while I gather a list of all available modules...

3204bda914b7f2c6f497__mypyc cloudpickle         marshal
slugify
IPython             cmath               math                smart_open
OpenSSL             cmd                 matplotlib          smmap
PIL                 code                matplotlib_inline   smtplib
PyQt5               codecs              mccabe              sndhdr
__future__          codeop              mdit_py_plugins     sniffio
__hello__           collections         mdurl
snowballstemmer
__phello__          colorama            menuinst            socket
_abc                colorcet            mimetypes
socketserver
_aix_support        colorsys            mistune             socks
_argon2_cffi_bindings comm                mkl
sockshandler
_ast                commctrl            mkl_fft
sortedcontainers
_asyncio            compileall          mkl_random          soupsieve
_bisect             concurrent          mmap                sphinx
_black_version      conda               mmapfile            sphinxify
_blake2             conda_build         mmsystem
sphinxthread
_brotli             conda_content_trust modulefinder        spyder
_bz2                conda_env           more_itertools
spyder_kernels
_cffi_backend       conda_index         mpmath              sqlalchemy
_codecs             conda_libmamba_solver msgpack             sqlite3
_codecs_cn          conda_pack          msilib
sre_compile
_codecs_hk          conda_package_handling msvcrt
sre_constants
_codecs_iso2022     conda_package_streaming multidict
sre_parse
_codecs_jp          conda_token         multipledispatch    ssl
_codecs_kr          conf                multiprocessing     sspi
_codecs_tw          configparser        mypy                sspicon
_collections        constantly          mypy_extensions     stack_data
_collections_abc    contextlib          mypyc               stat
_compat_pickle      contextvars         nacl                statistics
_compression        contourpy           navigator_updater
statsmodels
_contextvars        cookiecutter        nb_conda_kernels    streamlit
_csv                copy                nbclient            string
```

```
_ctypes              copyreg              nbconvert            stringprep
_ctypes_test         cpuinfo              nbformat             struct
_datetime            crypt                nest_asyncio         subprocess
_decimal             cryptography         netbios              sunau
_distutils_hack      cssselect            netrc                sympy
_elementtree         csv                  networkx             symtable
_functools           ctypes               nltk                 sys
_hashlib             curl                 nntplib              sysconfig
_heapq               curses               notebook             tables
_imp                 cwp                  notebook_shim        tabnanny
_io                  cycler               nt                   tabulate
_json                cytoolz              ntpath               tarfile
_locale              dask                 ntsecuritycon        tblib
_lsprof              dask_expr            nturl2path           telnetlib
_lzma                dataclasses          numba                tempfile
_markupbase          datashader           numbergen            tenacity
_md5                 datetime             numbers              terminado
_msi                 dateutil             numexpr              test
_multibytecodec      dbi                  numpy
test_pycosat
_multiprocessing     dbm                  numpydoc
text_unidecode
_nsis                dde                  odbc
textdistance
_opcode              debugpy              opcode               textwrap
_operator            decimal              openpyxl             this
_osx_support         decorator            operator             threading
_overlapped          defusedxml           optparse
threadpoolctl
_pickle              diff_match_patch     os
three_merge
_plotly_future_      difflib              overrides            tifffile
_plotly_utils        dill                 packaging            time
_py_abc              dis                  pandas               timeit
_pydatetime          distributed          pandocfilters        timer
_pydecimal           distro               panel                tinycss2
_pyio                docstring_to_markdown param                 tkinter
_pylong              doctest              paramiko             tldextract
_pytest              docutils             parsel               tlz
_queue               dotenv               parso                token
_random              email                partd                tokenize
_ruamel_yaml         encodings            pathlib              toml
_sha1                ensurepip            pathspec             tomli
_sha2                enum                 patsy                tomlkit
_sha3                erfa                 pdb                  tomllib
_signal              errno                perfmon              toolz
_sitebuiltins        et_xmlfile           pexpect              tornado
_socket              executing            pickle               tqdm
_sqlite3             fastjsonschema       pickleshare          trace
```

```
_sre                 faulthandler         pickletools          traceback
_ssl                 filecmp              pip
tracemalloc
_stat                fileinput            pipes                traitlets
_statistics          filelock             pkce                 truststore
_string              flake8               pkg_resources        tty
_strptime            flask                pkginfo              turtle
_struct              fnmatch              pkgutil              turtledemo
_symtable            fontTools            platform             twisted
_system_path         fractions            platformdirs         typeguard
_testbuffer          frozendict           plistlib             typer
_testcapi            frozenlist           plotly               types
_testclinic          fsspec               pluggy               typing
_testconsole         ftplib               ply
typing_extensions
_testimportmultiple  functools            poplib               tzdata
_testinternalcapi    gc                   posixpath            uc_micro
_testmultiphase      genericpath          pprint               ujson
_testsinglephase     gensim               profile
unicodedata
_thread              getopt               prometheus_client
unicodedata2
_threading_local     getpass              prompt_toolkit       unidecode
_tkinter             gettext              protego              unittest
_tokenize            git                  pstats               urllib
_tracemalloc         gitdb                psutil               urllib3
_typing              glob                 pty                  uu
_uuid                graphlib             ptyprocess           uuid
_warnings            greenlet             pure_eval            venv
_weakref             gzip                 py                   w3lib
_weakrefset          h11                  py_compile           warnings
_win32sysloader      h5py                 pyarrow              watchdog
_winapi              hashlib              pyasn1               wave
_winxptheme          heapdict             pyasn1_modules       wcwidth
_wmi                 heapq                pyclbr               weakref
_xxinterpchannels    hmac                 pycodestyle          webbrowser
_xxsubinterpreters   holoviews            pycosat
webencodings
_yaml                html                 pycparser            websocket
_zoneinfo            http                 pyct                 werkzeug
abc                  httpcore             pycurl
whatthepatch
adodbapi             httpx                pydantic             wheel
aext_assistant       hvplot               pydantic_core
widgetsnbextension
aext_assistant_server hyperlink             pydantic_settings    win2kras
aext_core            idlelib              pydeck               win32api
aext_core_server     idna                 pydispatch
win32clipboard
```

```
aext_panels          imagecodecs          pydoc                win32com
aext_panels_server   imageio              pydoc_data           win32con
aext_project_filebrowser_server imagesize           pydocstyle
win32console
aext_share_notebook imaplib              pyexpat              win32cred
aext_share_notebook_server imblearn             pyflakes
win32crypt
aext_shared          imghdr               pygments
win32cryptcon
aext_toolbox         importlib            pylab
win32ctypes
afxres               importlib_metadata   pylint               win32event
aifc                 importlib_resources pylint_venv
win32evtlog
aiobotocore          incremental          pyls_spyder
win32evtlogutil
aiohappyeyeballs     inflect              pylsp                win32file
aiohttp              inflection           pylsp_black          win32gui
aioitertools         iniconfig            pylsp_jsonrpc
win32gui_struct
aiosignal            inspect              pyodbc               win32help
alabaster            intake               pyparsing            win32inet
alembic              intervaltree         pytest
win32inetcon
altair               io                   pythoncom            win32job
anaconda_anon_usage ipaddress            pythonjsonlogger     win32lz
anaconda_catalogs    ipykernel            pytoolconfig         win32net
anaconda_cli_base    ipykernel_launcher   pytz
win32netcon
anaconda_cloud_auth ipython_genutils     pyviz_comms          win32pdh
anaconda_navigator   ipywidgets           pywin
win32pdhquery
anaconda_project     isapi                pywin32_bootstrap
win32pdhutil
annotated_types      isort                pywin32_testutil     win32pipe
antigravity          isympy               pywintypes           win32print
anyio                itemadapter          pywt
win32process
appdirs              itemloaders          qdarkstyle
win32profile
archspec             itertools            qstylizer            win32ras
argon2               itsdangerous         qtawesome
win32rcparser
argparse             jedi                 qtconsole
win32security
array                jellyfish            qtpy
win32service
arrow                jinja2               queue
win32serviceutil
```

```
ast                   jmespath              queuelib
win32timezone
astroid               joblib                quopri                win32trace
astropy               json                  random
win32traceutil
astropy_iers_data     json5                 rasutil
win32transaction
asttokens             jsonpatch             re                    win32ts
async_lru             jsonpointer           readchar              win32ui
asyncio               jsonschema            referencing           win32uiole
atexit                jsonschema_specifications regcheck
win32verstamp
atomicwrites          jupyter               regex                 win32wnet
attr                  jupyter_client        regutil
win_inet_pton
attrs                 jupyter_console       repo_cli              winerror
audioop               jupyter_core          reprlib
winioctlcon
autocommand           jupyter_events        requests              winnt
automat               jupyter_lsp           requests_file         winperf
autopep8              jupyter_server        requests_toolbelt     winpty
babel                 jupyter_server_terminals rfc3339_validator
winreg
backports             jupyterlab            rfc3986_validator     winsound
base64                jupyterlab_plotly     rich                  winxpgui
bcrypt                jupyterlab_pygments   rlcompleter           winxptheme
bdb                   jupyterlab_server     rope                  wrapt
binaryornot           jupyterlab_widgets    rpds                  wsgiref
binascii              jwt                   rtree                 xarray
binstar_client        keyring               ruamel_yaml           xdrlib
bisect                keyword               runpy                 xlwings
black                 kiwisolver            s3fs                  xlwingsjs
blackd                lazy_loader           sched                 xml
bleach                lazy_object_proxy     scipy                 xmlrpc
blib2to3              lckr_jupyterlab_variableinspector scrapy
xxlimited
blinker               lib2to3               seaborn
xxlimited_35
bokeh                 libarchive            secrets               xxsubtype
boltons               libmambapy            select
xyzservices
botocore              lief                  selectors             yaml
bottleneck            linecache             semver                yapf
brotli                linkify_it            send2trash
yapf_third_party
bs4                   llvmlite              sentry_sdk            yapftests
builtins              lmdb                  service_identity      yarl
bz2                   locale                servicemanager        zict
cProfile              locket                setuptools            zipapp
```

```
cachetools          logging             shellingham         zipfile
calendar            lxml                shelve              zipimport
certifi             lz4                 shlex               zipp
cffi                lzma                shutil              zlib
cgi                 mailbox             signal              zmq
cgitb               mailcap             sipbuild            zoneinfo
chardet             mako                site                zope
charset_normalizer  markdown            six                 zstandard
chunk               markdown_it         skimage
click               markupsafe          sklearn

Enter any module name to get more help.  Or, type "modules spam" to
search
for modules whose name or summary contain the string "spam".
```

Loops in python

- While Loop
- For Loop

```
#while loop example -> program to print table
#program -> sum of all digits of a given number
#program -> keep accepting number from users till he/she enters a 0
and then find the average

# print table
number=int(input("enter any number "))
i=1
while(i<11):
    print(number*i)
    i+=1

enter any number  15

15
30
45
60
75
90
105
120
135
150
```

while loop with else

```
x=1
while x<3 : # jaise hi condition false ho jayega to else bala run ho jayega but dhyan rahe ki ye apne aap khatam hona chahaiye
    print(x)    #forcefully nahi jaise ki break statement kahi use kar dye loop ke andar to else condition nahi chalega
    x+=1
else:
    print("limit crossed")
```

```
1
2
limit crossed
```

guessing game

step : generate a random integer between 1 and 100

```
import random
jackpot=random.randint(1,100)

guess=int(input("guess karo ek number"))
counter=1
while guess!=jackpot:
    if guess < jackpot :
        print("galat ! guess heigher number")
    else :
        print("galat ! guess lower number ")
    guess=int(input("guess karo phir se ek number"))
    counter+=1

else :
    print("correct guess")
    print("attempts = ",counter)
```

```
guess karo ek number 55

galat ! guess lower number

guess karo phir se ek number 50

galat ! guess lower number

guess karo phir se ek number 30

galat ! guess lower number

guess karo phir se ek number 26

galat ! guess lower number
```

```
guess karo phir se ek number 20

galat ! guess lower number

guess karo phir se ek number 8

galat ! guess heigher number

guess karo phir se ek number 15

galat ! guess lower number

guess karo phir se ek number 10

galat ! guess heigher number

guess karo phir se ek number 12

galat ! guess heigher number

guess karo phir se ek number 13

galat ! guess heigher number

guess karo phir se ek number 14

correct guess
attempts =  11
```

for loop

```
for i in range(1,11): # range me pahla number included and last bala number excluded hota hai
    print(i)

1
2
3
4
5
6
7
8
9
10

import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
```

```
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```python
for i in range(1,11,2): #3rd parameter step size (or jump) ko dikhata hai
    print(i)
```

```
1
3
5
7
9
```

```python
#printing in reverse order
for i in range(10,0,-1):
    print(i)
```

```
10
9
8
7
6
5
4
3
2
1
```

```python
for i in "delhi":
    print(i)
```

```
d
e
l
h
i
```

program

- the current population of a town is 10000. the population of the town is increasing at the rate of 10% per year .you have to write a program to find out the population at the end of each of the last 10 years

```
population=10000
for i in range(10,0,-1):

    print(i," year : ",population)
    population=(population/1.1)

"""let population in 9th year is x
    x + 10%of x = 10000
    x+0.1x=10000
    1.1x=10000
    x=10000/1.1"""
```

Sequence sum

1/1! + 2/2! +3/3!+....

```
n=int(input("enter n : "))
result=0
fact=1
for i in range(1,n+1):
    fact=fact*i
    result+=i/fact
print(result)
```

```
enter n :  15
```

```
2.71828182845823
```

Nested loop

```
# example ->unique pair
for i in range(1,5):
    for j in range(1,5):
        print(i,j)
```

```
1 1
1 2
1 3
1 4
2 1
2 2
```

```
2 3
2 4
3 1
3 2
3 3
3 4
4 1
4 2
4 3
4 4
```

Pattern printing

****

```
n=int(input("enter the size of row "))
for i in range (1,n+1):
    for j in range (1,i+1):
        print("*",end=" ")
    print()
```

```
enter the size of row  5

*
* *
* * *
* * * *
* * * * *
```

pattern 2

1 121 12321 1234321

```
n=int(input("enter the size of row "))
for i in range (1,n+1):
    for j in range (1,i+1):
        print(j,end=" ")
    for k in range(i-1,0,-1):
        print(k,end=" ")
    print()
```

```
enter the size of row  5

1
1 2 1
1 2 3 2 1
```

```
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

Loop control statement

- Break --> suppose kuch search kar rahe hain and wo 1 se 100 ke bich me hai suppose 58 number pe hai to mujhe jaise hi wo mil jaye to hum aage search nahi karenge 58 pe break kar denge it can be an example of database
- Continue
- Pass

```
for i in range(1,10):
    if i==5:
        break
    print(i)
```

```
1
2
3
4
```

ques

- print series of prime number

```
lower=int(input("enter lower limit "))
upper=int(input("enter upper limit "))
for i in range(lower,upper+1):
    for j in range(2,i):
        if i%j==0:
            break
    else:
        print(i)
```

```
enter lower limit  10
enter upper limit  50
```

```
11
13
17
19
23
29
31
37
41
```

```
43
47
```

continue

- current iteration ko skip kar deta hai , suppose aap koi product search kar rahe ho but uss product ko skip karna chahate ho jiska ki rating 3 se low hai to hum simply ek loop laga sakte hain and condition laga sakte hain ki "if(rating<3) continue" to ye 3 se kam rating bale product ko skip kar deta hai

```
for i in range(1,10):
    if i==5:
        continue
    print(i)
```

```
1
2
3
4
6
7
8
9
```

pass

- yadi aap kuch likhna chahate hain but kuch der ke lye skip karna chahate hain to pass ka use karte hain but agar sirf loop likh ke chor dete hain to error throw karta hai

```
for i in range(1,10):
```

```
  Cell In[48], line 2

    ^
SyntaxError: incomplete input
```

```
for i in range(1,10):
    pass
```

string

Strings are basically sequence of charecters in python specifically , strings are a sequence of unicode characters(not same as ascii) ascii me charecter ka size 1 byte but unicode me 2 byte qki problem is arise ki hum sirf english ke alphabet hi use nahi karenge balki iske alawa bhi bahut

sare language hain and emojies hain jise ki hum use kar rahe honge. .Creating String .Accessing String .Adding Char to Strings .Editiing Strings .Deleting Strings .Operation On Strings .String function

creating String

```
s1='hello'
s2="hello"
s3='''hello''' # for multiline string
s4="""hello""" # for multiline string
print(s1)
print(s2)
print(s3)
print(s4)


# all the above  methode are true for initializing a string
```

```
hello
hello
hello
hello
```

```
print('it's raining outside') # ye samajh hi nahi paa raha hai ki kaha se suru and kaha pe end ho raha hai
                #agar kabhi aaisi need pad jaye to proper tarike se apne string ko print kara sake iske lye hi alag alag method ka use karte hain
```

```
  Cell In[56], line 1
    print('it's raining outside')
                               ^
SyntaxError: unterminated string literal (detected at line 1)
```

```
print("it's raining today")
```

```
it's raining today
```

type conversion method

```
s=str(123)
print(s)
print(type(s))
```

```
123
<class 'str'>
```

accessing Substring from a string

```
# Positive indexing -> provide an index to each character starting
from zero and continued incremented by one
s="hello world"
print(s[0])
print(s[3])

h
l

print(s[73]) # error qki humare pass 73 tak index hai hi nahi

---------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
Cell In[75], line 1
----> 1 print(s[73])

IndexError: string index out of range
```

negative indexing

- python provide the features of negative indexing in which last charcter is indexed with -1 and 2nd last is indexed with -2 and so on...(move right to left)

```
s="hello world"
print(s[-1])
print(s[-5])

d
w

print(s[-15]) # erroe because of index out of range

---------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
Cell In[81], line 1
----> 1 print(s[-15])

IndexError: string index out of range
```

slicing

```
s="hello world"
print(s[0:5]) # 1st parameter ki kaha se suru karna hai && 2nd parameter ki kaha tak jana hai 2nd parameter is always excluded
```

```
hello
```

```
print(s[2:8])
```

```
llo wo
```

```
print(s[3:]) # hum 2nd parameter ko skip kar sakte hain wo last index ko consider kar lega
```

```
lo world
```

```
print(s[:7]) #hum 1st parameter ko bhi skip kar sakte hain wo simply 0 index se suru ho jayega
```

```
hello w
```

```
print(s[:]) #hum dono parameter ko skip kar sakte hain wo simply suru se end tak print kar dega
```

```
hello world
```

#concept of step size-> agar hume ek charecter ke baad ek charecter print karana ho to ya phir ek ke baad do character ko skip karna ho and so on to hum step size ka use karte hain as 3rd parameter and by default step size ka value 1 hota hai agar bhul rahe ho to ise aache se samajhne ke lye number line bala concept yaad karo

```
print(s[0:8:2]) # 3rd parameter as 2 means ek ke baad ek print karenge "hello"=>[0,1,2,3,4] i.e 2-0=4,4-2=2 -- 3dr parameter
                #basically jo index print karana hai continuously uske bich ka difference hota hai
```

```
hlow
```

```
print(s[0::3])
```

```
hlwl
```

```
print(s[0:6:-1]) # kuch bhi print nahi hoga qki 0 to 6 , jana hai left to right direction hai but step size negative(right to left ka hai)


print(s[6:0:-1]) # abb dono direction same hai
```

```
w olle
```

```
print(s[6:0:-2])
```

```
wol
```

working with negative index

```
print(s[-5:-1])
```

```
worl
```

```
print(s[-5:])
```

```
world
```

```
print(s[-1:-6:-1]) # reverse order me with the help of negative index kuch print karana
```

```
dlrow
```

reversing the array

```
print(s[::-1]) #reverse the string
```

```
dlrow olleh
```

```
print(s[-1:-12:-1])
```

```
dlrow olleh
```

Editing and deleting in string

```
s="hello World"
s[0]="p"
print(s)  # error qki  in python string is immutable hum jo string ek baar bana dye hain use change nahi kar sakte hain
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[129], line 2
      1 s="hello World"
----> 2 s[0]="p"
      3 print(s)

TypeError: 'str' object does not support item assignment
```

```
# deleting the string
s="jay ram"
del s      #string s delete ho jayega and abb future me s string ko
use nahi kar payenge and one importent thing ki
             #string jab bhi hoga complete delete hoga aadha adhura
string delete nahi hota hai
print(s)
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[131], line 4
      2 s="jay ram"
      3 del s
----> 4 print(s)

NameError: name 's' is not defined
```

Operations on String

- Arithmetic Operations (only + and * not - and /)
- Relational operations
- logical operations
- loops on Strings
- Membership Operations

Arithmetic Operator

```
print("jay" + " " + "ram")
```

```
jay ram
```

```
print("jay " * 12)
```

```
jay jay jay jay jay jay jay jay jay jay jay jay
```

```
print("*" * 50)
```

```
**************************************************
```

Relational Operator

```
"delhi"!="delhi"
```

```
False
```

```
"mumbal">"pune"
```

```
False
```

```
print("pune">"Pune")
```

```
True
```

Logical Operator

```
 # agar appke string me character hai to wo python me true consider kya jata hai aur agar khali hai to false

"" and "world"  # ek bhi false to output false and false ko string me khali string se denote karte hain so output is khali string
```

```
''
```

```
"" or "world" # ek bhi true to output is true
```

```
'world'
```

```
"hello" or "world" # output hello qki jaise hi dekha pahla bala true hai and aage or operator hai to next bale ke pass jane ki naad hi nahi hai
```

```
'hello'
```

```
"hello" and "world" #output world qki jab first string is true and the and operator too ye next string ko evaluate karne gya and output is "world"
```

```
'world'
```

```
not ""
```

```
True
```

```
not "hello"
```

```
False
```

Loops

```
for i in "hello":  # string in python is iterable
    print(i)
```

```
h
e
l
l
o
```

```
for i in "delhi":
    print("pune")
```

```
pune
pune
pune
pune
pune
```

Membership oerator

```
"d" in "delhi"
```

```
True
```

```
"d" not in "delhi"
```

```
False
```

common functions

- len
- max
- min
- sorted

```
len("hello world") # string ka length find karega
```

```
11
```

```
max("hello world") # maximum value nikal ke dega according to ascii
```

```
'w'
```

```
min("hello world") # minimum value nikal ke dega according to ascii
```

```
' '
```

```
sorted("hello world") #sort kar dega according to ascii value ye string nahi balki list return karta hai
```

```
[' ', 'd', 'e', 'h', 'l', 'l', 'l', 'o', 'o', 'r', 'w']
```

```
sorted("hello world",reverse=True) #reverse order me sort karke dega
```

```
['w', 'r', 'o', 'o', 'l', 'l', 'l', 'h', 'e', 'd', ' ']
```

Capitalize/Title/Upper/lower/Swapcase

```
s="hello world"
s.capitalize()  # first letter ko capital kar deta hai
```

```
'Hello world'
```

```
s.title() # har word ka pahla letter capital kar dega
```

```
'Hello World'
```

```
print(s.upper())   #sabko capital letter me convert kar dega
print(s)     #ye sare function orignal string ko change nahi kar raha hai
```

```
HELLO WORLD
hello world
```

```
s.lower()  # sabko small letter me convert kar dega
```

```
'hello world'
```

```
"HeLlO WorLD".swapcase()  #jo small letter me hai usko capital and jo capital hai usko small letter me swap kar dega
```

```
'hElLo wORld'
```

count/find/index

```
"my name is jay".count("a") #pure string me a kitne baar aaya hai uska count bata dega
```

```
2
```

```
"my name is jay , this is a book".find("is") # 'is' ka 'i' first time kon se index pe hai
```

```
8
```

```
"my name is jay , this is a book".find("x")  # jo chiz hai hi nahi string me uske lye -1 return kar dega
```

```
-1
```

```
"my name is jay , this is a book".index("is") # ye bhi similar find ke jaisa hi kaam karta but different is this ki agar hummindex ke help se

                    #kuch aaisa find karna chahate hain jo ki available hi nahi to ye error throw karta hai jabki find -1 return karta hai
```

```
8
```

```
"my name is jay , this is a book".index("ram")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[26], line 1
----> 1 "my name is jay , this is a book".index("ram")

ValueError: substring not found
```

endswith/startswith

```
"my name is jay , this is a book".endswith("ook") # ye batata hi ki mera jo string hai wo isse ("ook") se end ho raha hai ki nahi similar
                                                  #startwith kaam karta hai
```

```
True
```

```
"my name is jay , this is a book".endswith("ooky")
```

```
False
```

```
"my name is jay , this is a book".startswith("my")
```

```
True
```

```
"my name is jay , this is a book".startswith("by")
```

```
False
```

format

```
name="jay"
gender="male"

print("hi my name is {} and i am {}".format(name,gender))  # pahle curly bracket ke andar formet me jo pahle likha hi wo and next bale me
                                                           # next value chala jayega
```

```
hi my name is jay and i am male
```

```
name="jay"
gender="male"
```

```
"hi my name is {} and i am {}".format(gender,name)
```

```
'hi my name is male and i am jay'
```

```
name="jay"
gender="male"

"hi my name is {1} and i am {0}".format(gender,name) # curly bracket ke andar  index bhi pass kar sakte hain formet ke
                                    #andar jo pass karte hain uska indexing 0 se suru hota and aage badhte jata hai
```

```
'hi my name is jay and i am male'
```

```
name="jay"
gender="male"

"hi my name is {} and i am {}".format("jay","male")
```

```
'hi my name is jay and i am male'
```

```
name="jay"
gender="male"

"hi my name is {1} and i am {0}".format("jay","male")
```

```
'hi my name is male and i am jay'
```

isalnum/isalpha/isdigit/isidentifier

```
"jay123".isalnum()  # 'isalnum' check karta hai ki humara string alpha numeric hai ki nahi matlab humara jo string bana hai wo sirf
                        # alphabet ya number ya dono se mil kar bana hi ki nahi
```

```
True
```

```
"jay".isalnum()
```

```
True
```

```
"123".isalnum()
```

```
True
```

```
"jay123@#%".isalnum()  #isme alphabet and number ke alawa bhi character hai
```

```
False
```

isalpha

```
"jay".isalpha()  # sirf alphabet hai ki nahi
```

```
True
```

```
"jay123".isalpha()
```

```
False
```

isdigit

```
"123".isdigit() # sirf digit hai ki nahi
```

```
True
```

```
"jay123".isdigit()
```

```
False
```

isidentifier

```
"jay123".isidentifier()  #valid identifier hai ki nahi matlab ki jo variable nominclature ke rule hain usko follow karta hai ki nahi
```

```
True
```

```
"123jay".isidentifier()
```

```
False
```

Split/Join

```
"hi my name is jay".split()  # ye word by word split kar dega and python list me convert kar dega
```

```
['hi', 'my', 'name', 'is', 'jay']
```

```
"hi my name is jay".split("i")  # yadi hum koi specific value pass karte hain to uske basis pe split kar dega for ex : jab jab "i" aayega

                                #string tab tab split kar dega and wo value eleminate ho jayega matlab "i" list me kahi nahi dikhega
```

```
['h', ' my name ', 's jay']
```

```
s='Data science mentorship program'
for i in s.split():
    print(i)
```

```
Data
science
```

```
mentorship
program
```

join

```
" ".join(['hi', 'my', 'name', 'is', 'jay']) # ye join kar dega suru me ek blank string hain jisme ki ek space hai to ek space har word ke baad de dega
```

```
'hi my name is jay'
```

```
"-".join(['hi', 'my', 'name', 'is', 'jay']) # ye abb hippen ke sath join ho jayega
```

```
'hi-my-name-is-jay'
```

replace

```
'hi my name is jay'.replace("jay","ram") # string me jaha bhi "jay" hoga usko "ram" se replace kar dega agar "jay" string me hoga
                                         #hi nahi to kuch bhi chnage nahi karega
```

```
'hi my name is ram'
```

strip

```
"jay                    ".strip() #ye suru aur end ke white space ko khatam kar deta hai suppose aap data base me koi name
                                  #add kar rahe ho aur galti se white space add ho gya to bbar baar data base se name fatch karte white
                                     #space bhi dikhega so strip function ka use karte hain
```

```
'jay'
```

```
"          jay            ".strip()
```

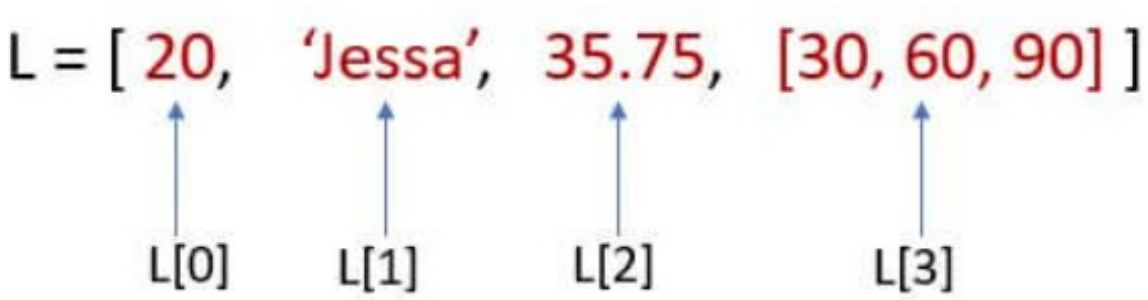
```
'jay'
```

lecture 4

Lists

What are Lists

List is a data type where you can store multiple items under 1 name. More technically, lists act like dynamic arrays which means you can add more items on the fly.



- Why Lists are required in programming?

Array Vs Lists

- Fixed Vs Dynamic Size-> list are dynamic sized (ye vector ke tarah kaam karta hai jaise usme capacity increase hota hai waise hi list me bhi hota hai)
- Convenience -> Hetrogeneous
- Speed of Execution of list is slow then array
- more Memory occupies by list then array

How lists are stored in memory

```
l=[1,2,3]

print(id(l))
print(id(l[0]))
print(id(l[1]))
print(id(l[2]))
print(id(1))
print(id(2))
print(id(3))


2134559477312
140720820857272
140720820857304
140720820857336
```

```
140720820857272
140720820857304
140720820857336
```

Characterstics of a List

- Ordered matters -> kiss order me element store hai wo matter karta hai
- Changeble/Mutable
- Hetrogeneous
- Can have duplicates
- are dynamic
- can be nested
- items can be accessed
- can contain any kind of objects in python

creating list

```
# Empty

print([])

# 1D -> Homo
print([1,2,3,4,5])

# 2D  -> list ke andar list isse nested bhi kah sakte hain
print([1,2,3,4,[5,6]])

# 3D
print([[[1,2],[3,4]],[[5,6],[7,8]]])

# Hetrogenous
print([1,True,5.6,5+6j,"hello"])

# Using Type conversion
print(list("hello"))
```

```
[]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, [5, 6]]
[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]
[1, True, 5.6, (5+6j), 'hello']
['h', 'e', 'l', 'l', 'o']
```

Accessing Items from a List

```
#indexing--> positive and negative dono
```

```
l=[1,2,3,4,5]
  #positive indexing
print(l[0])  # index aukat se jyada denge to out of bound ka error aa
jayega

   # negative indexing
print(l[-3])
```

```
1
3
```

```
# 2D INDEXING
l=[1,2,3,4,[5,6]]
print(l[4][1])
```

```
6
```

```
print(l[4][0])
```

```
5
```

```
#3D INDEXING
l=[[[1,2],[3,4]],[[5,6],[7,8]]]
print(l[0][0][1])
```

```
2
```

slicing

- similar of string

```
l=[1,2,3,4,5,6,7]
print(l[1:6])
```

```
[2, 3, 4, 5, 6]
```

```
print(l[::-1])
```

```
[7, 6, 5, 4, 3, 2, 1]
```

Adding Items to a List

```
# append => list ke last me jake ek element ko add kar deta hai
l=[1,2,3,4,5]
l.append("a")
l.append(True)
l.append([6,7,8])  # ye pure list ko ek item ke tarah consider karega
print(l)
```

```
[1, 2, 3, 4, 5, 'a', True, [6, 7, 8]]
```

```
#extend => ye ek se jyada element ko add karta hai last me
l=[1,2,3,4,5]
```

```
l.extend([6,7,8])
l.extend("ram") # ram ke har character ko extract karega and ek list banayega and then uske item ko add kar dega
print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 'r', 'a', 'm']
```

```
# insert => apne desired location pe element ko add kar dega
#insert(para1,para2) => para1--> kis index pe add karna hai && para2 --> kya add karna hai
l=[1,2,3,4,5]
l.insert(2,"jay")
print(l)
```

```
[1, 2, 'jay', 3, 4, 5]
```

Editing items in a list

```
# editing possible hai qki list mutable hai

# hum single element ko change kar sakte hain
l=[1,2,3,4,5]
l[2]="ram"
print(l)
```

```
[1, 2, 'ram', 4, 5]
```

```
l=[1,2,3,4,5]
l[2]=["ram",True]
print(l)
```

```
[1, 2, ['ram', True], 4, 5]
```

```
# hum multiple element ko bhi add kar sakte hain
l=[1,2,3,4,5]
l[1:4]="ramujhhbd"  # ye bhi possdible hai
print(l)
```

```
[1, 'r', 'a', 'm', 'u', 'j', 'h', 'h', 'b', 'd', 5]
```

```
l=[1,2,3,4,5]
l[1:4]=[123]
print(l)
```

```
[1, 123, 5]
```

```
l=[1,2,3,4,5]
l[1:4]=[200,300,400]
print(l)
```

```
[1, 200, 300, 400, 5]
```

```
l=[1,2,3,4,5]
l[1:4]=[200,300,400,500,6000]
print(l)
```

```
[1, 200, 300, 400, 500, 6000, 5]
```

Deleting From a list

```
#del
l=[1,2,3,4,5]
print(l)
del(l) # complete list ko hi delete kar dega
```

```
[1, 2, 3, 4, 5]
```

```
print(l)  # error qki l delete ho chuka hai
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[106], line 1
----> 1 print(l)

NameError: name 'l' is not defined
```

```
# kisi particular element ko delete karna
l=[1,2,3,4,5]
print(l)
del(l[2])  # 2nd index ko delete karna
print(l)
```

```
[1, 2, 3, 4, 5]
[1, 2, 4, 5]
```

```
# with the help of slicing
l=[1,2,3,4,5]
print(l)
del(l[1:3])  # 1st and 2nd index ko delete kar dega => indexing ke help se delete
print(l)
```

```
[1, 2, 3, 4, 5]
[1, 4, 5]
```

```
# remove => direct element ko hi delete kar dega  indexing ki koi need nahi hai
```

```
l=[1,2,3,4,5]
l.remove(5)
print(l)
```

```
[1, 2, 3, 4]
```

```
l=[1,5,5,6,5,5,9,7,6,3]
l.remove(l[9])
print(l)
```

```
[1, 5, 5, 6, 5, 5, 9, 7, 6]
```

```
l=[1,5,5,6,5,5,9,7,6,3]
l.remove(l[5])   # 5th index pe jayega and then waha ke element ko fetch karega and then uss element ke first occurrence ko remove kar dega
print(l)
```

```
[1, 5, 6, 5, 5, 9, 7, 6, 3]
```

```
#pop =>
l=[1,2,3,4,5]
l.pop(0)  # jis index ko delete karna hai wo provide kar do agar index provide nahi karoge to last item ko delete kar dega
print(l)
```

```
[2, 3, 4, 5]
```

```
l=[1,2,3,4,5]
l.pop()  # jis index ko delete karna hai wo provide kar do. agar index provide nahi karoge to last item ko delete kar dega
print(l)
```

```
[1, 2, 3, 4]
```

```
# clear
l=[1,2,3,4,5]
l.clear()  # ye list ke sare item ko delete kar deta hai aur list ko blank bana deta hai
print(l)
```

```
[]
```

Operations on Lists

- Arithmetic
- Membership
- Loop

```
#Arithmetic(+,*)
l1=[1,2,3,4,5]
l2=[6,7,8,9]
```

```
    #Concatination/Merge
print(l1+l2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print(l1*3)
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
# membership operator
l1=[1,2,3,4,5]
l2=[6,7,8,9]
print(5 in l1)
```

```
True
```

```
l1=[1,2,3,4,5]
l2=[6,7,8,9]
print(5 not in l2)
```

```
True
```

```
l1=[1,2,3,4,5]
l2=[6,7,8,9,[4,5]]
print(5 in l1)
print(5 in l2)
print([4,5] in l2)
print(5 in l2[4])
```

```
True
False
True
True
```

```
#loops
l1=[1,2,3,4,5]
l2=[6,7,8,9,[4,5]]
for i in l1 :
    print(i)
```

```
1
2
3
4
5
```

```
for i in l2:
    print(i)
```

```
6
7
8
```

```
9
[4, 5]
```

```
for i in range(4,5):
    for j in range(0,2):
        print(l2[i][j])
```

```
4
5
```

```
l3=[[[1,2],[3,4]],[[5,6],[7,8]]]
for i in l3:
    print(i)
```

```
[[1, 2], [3, 4]]
[[5, 6], [7, 8]]
```

List Functions

sum/len/min/max/sorted

```
l=[2,1,5,7,0]
print(sum(l))  # sare element ka sum find kar dega
print(len(l))  # list ka length find kar dega
print(min(l))  # min aur max function tabhi kaam karega jab ki element homogenious hoga
print(max(l))
print(sorted(l))  # sort karta hai by default ascending order me
print(sorted(l,reverse="True"))  # descending order me sort kar dega
```

```
15
5
0
7
[0, 1, 2, 5, 7]
[7, 5, 2, 1, 0]
```

count

- list me koi element kitne baar aaya hai wo count karke bata dega

```
l=[2,1,5,7,0,5,7,15]
print(l.count(5))
print(l.count(19))
```

```
2
0
```

index

- kisi particular element ke index ko batata hai agar koi element multiple time aata hai to uska first occurrence batata hai

```
l=[2,1,5,7,0,5,7,15]
print(l.index(5))
```

```
2
```

```
l=[2,1,5,7,0]
l.reverse() #permanently list ko reverse kar dega lekin uper jo function padhe hain wo permanent change nahi
print(l)      # karta hai sirf ek list provide kar deta hai
```

```
[0, 7, 5, 1, 2]
```

sort

- ye permanent sort kar deta hai and ye original list me change karta hai

```
# sort (vs sorted)
l=[2,1,5,7,0]
print(l)
print(sorted(l))
print(l)
l.sort()
print(l)
```

```
[2, 1, 5, 7, 0]
[0, 1, 2, 5, 7]
[2, 1, 5, 7, 0]
[0, 1, 2, 5, 7]
```

copy

- ek shallow copy banata hai matlab memory me ek new address pe same list ban jata hai

```
l=[2,1,5,7,0]
print(l)
print(id(l))
l1=l.copy()
print(l1)
print(id(l1))
```

```
[2, 1, 5, 7, 0]
2378207398784
[2, 1, 5, 7, 0]
2378206817152
```

List Comprehension

List Comprehension provides a concise way of creating lists.(shortcut tarika hai list create karne ka)

newlist = [expression for item in iterable if condition == True]

```
newlist = [expression for item in iterable if condition == True]
```

Advantages of List Comprehension

- More time-efficient and space-efficient than loops.
- Require fewer lines of code.
- Transforms iterative statement into a formula.

```
# add 1 to 10 numbers to a list

# normal tarika

l=[]
for i in range (1,11):
    l.append(i)

print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
# bindas tarika

l=[i for i in  range(1,11)]
print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
#scalar multiplication on a vector
v=[2,3,4]
s=-3
print(v*-3)  # ye ek blank list return karega

# humari requirement fullfil nahi ho paa rahi hai  upar ke tarike se
mujhe each element ko -3 se multiply karna tha
# but niche jo method apply kye hain usse kaam ho jayega but ye hai
normal zindigi
x=[]
for i in v:
    x.append(i*s)

print(x)

#bindas zindigi
```

```
z=[s*i for i in v]
print(z)
```

```
[]
[-6, -9, -12]
[-6, -9, -12]
```

```
# add squares
l=[1,2,3,4,5]
[i**2 for i in l]
```

```
[1, 4, 9, 16, 25]
```

```
# print all the numbers divisible by 5 in the range of  1 to 50

[i for i in range(1,51) if i%5==0]
```

```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

```
# find languages which start with letter p
languages = ['java','python','php','c','javascript']
[language for language in languages if language.startswith("p")]
```

```
['python', 'php']
```

```
# Nested if with List Comprehension
basket = ['apple','guava','cherry','banana']
my_fruits = ['apple','kiwi','grapes','banana']

# add new list from my_fruits and items if the fruit exists in basket
and also starts with 'a'

[fruit for fruit in my_fruits if fruit in basket if
fruit.startswith('a')]
```

```
['apple']
```

```
# Print a (3,3) matrix using list comprehension -> Nested List
comprehension
[[i*j for i in range(1,4)] for j in range(1,4)]
```

```
[[1, 2, 3], [2, 4, 6], [3, 6, 9]]
```

```
# cartesian products -> List comprehension on 2 lists together
L1 = [1,2,3,4]
L2 = [5,6,7,8]

[i*j for i in L1 for j in L2]
```

```
[5, 6, 7, 8, 10, 12, 14, 16, 15, 18, 21, 24, 20, 24, 28, 32]
```

2 ways to traverse a list

- itemwise
- indexwise

```
#itemwise loop chalana
l=[1,2,3,4]
for i in l :
    print(i)
```

```
1
2
3
4
```

```
#index wise loop chalana
l=[1,2,3,4]
for i in range(0,len(l)):
    print(i)
```

```
0
1
2
3
```

```
l=[1,2,3,4]
for i in range(0,len(l)):
    print(l[i])
```

```
1
2
3
4
```

Zip

The zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together.

If the passed iterators have different lengths, the iterator with the least items decides the length of the new iterator.

```
# zip basically do list ko respective item wise pair karta hai and
agar do list ka size different hai to chote bale ke according pair
# hoga matlab paired list me utna hi item hoga jitna ki small list me
hai

# Write a program to add items of 2 lists indexwise

L1 = [1,2,3,4]
```

```
L2 = [-1,-2,-3,-4]
zip(L1,L2)
```

```
<zip at 0x2207b9cbe40>
```

```
print(zip(L1,L2))
```

```
<zip object at 0x000001FE62000380>
```

```
L1 = [1,2,3,4]
L2 = [-1,-2,-3,-4]
list(zip(L1,L2))
```

```
[(1, -1), (2, -2), (3, -3), (4, -4)]
```

```
l1 = [1,2,3,4]
l2=[2,3,5,6,7,8]
zip(l1,l2)
```

```
<zip at 0x1fe62009540>
```

```
list(zip(l1,l2))
```

```
[(1, 2), (2, 3), (3, 5), (4, 6)]
```

```
L1 = [1,2,3,4]
L2 = [-1,-2,-3,-4]
list(zip(L1,L2))
[i+j for i,j in zip(L1,L2)]
```

```
[0, 0, 0, 0]
```

```
#can contain any kind of objects in python
l=[1,2,print,type,input,'jay']
print(l)
```

```
[1, 2, <built-in function print>, <class 'type'>, <bound method
Kernel.raw_input of <ipykernel.ipkernel.IPythonKernel object at
0x000001FE5DBD3E30>>, 'jay']
```

Disadvantages of Python Lists

- Slow
- Risky usage
- eats up more memory

```
a=[1,2,3]
b=a
print(a)
```

```
print(b)
a.append(4)
print(a)
print(b)
 # ye islye ho raha hai qki list are mutable => b=a.copy() karenge to ye nahi hoga
b=a.copy()
a.append(8)
print(a)
print(b)
```

```
[1, 2, 3]
[1, 2, 3]
[1, 2, 3, 4]
[1, 2, 3, 4]
[1, 2, 3, 4, 8]
[1, 2, 3, 4]
```

List Programs

```
# Create 2 lists from a given list where
# 1st list will contain all the odd numbers from the original list and
# the 2nd one will contain all the even numbers

L = [1,2,3,4,5,6]

l1=[i for i in L if i%2!=0 ]
l2=[i for i in L if i%2==0 ]
print(l1)
print(l2)
```

```
[1, 3, 5]
[2, 4, 6]
```

```
# m-2
l1=[]
l2=[]
for i in L:
    if i%2!=0:
        l1.append(i)
    else:
        l2.append(i)

print(l1)
print(l2)
```

```
[1, 3, 5]
[2, 4, 6]
```

```
# How to take list as input from user

L=[]
num=1

while True:
    num=int(input('Enter list item(enter any character to complete) :
'))
    if num!=-1:
        L.append(num)
    else:
        break

L
```

```
Enter list item(enter any character to complete) :  1
Enter list item(enter any character to complete) :  2
Enter list item(enter any character to complete) :  3
Enter list item(enter any character to complete) :  4
Enter list item(enter any character to complete) :  5
Enter list item(enter any character to complete) :  -1
```

```
[1, 2, 3, 4, 5]
```

```
# Write a program to merge 2 list without using the + operator
L1 = [1,2,3,4]
L2 = [5,6,7,8]

for i in L2:
    L1.append(i)

print(L1)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
# Write a program to replace an item with a different item if found in
the list
L = [1,2,3,4,5,3]
# replace 3 with 300
x=int(input('enter the replacebale item : '))
y=int(input('Enter the new value : '))


for i in range(0,len(L)):
    if L[i]==x:
        L[i]=y

print(L)
```

```
enter the replacebale item :  3
Enter the new value :  300

[1, 2, 300, 4, 5, 300]
```

```
# Write a program that can convert a 2D list to 1D list
L=[[1,2,3],[4,5,6],[7,8,9]]
l=[]
for i in L:
    for j in i:
        l.append(j)

print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
# Write a program to remove duplicate items from a list

L = [1,2,1,2,3,4,5,3,4]
l1=[]
for i in L:
    if i not in l1:
        l1.append(i)

print(l1)
```

```
[1, 2, 3, 4, 5]
```

```
# Write a program to check if a list is in ascending order or not

L=[1,2,3,4,5,6]

for i in range(0,len(L)-1):
    flag=False
    if L[i+1]>=L[i]:
        flag=True
    else:
        break


if flag==True:
    print("yes")
else:
    print('no')
```

```
yes
```

lecture 5

```
# Tuples

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

In short, a tuple is an immutable list. A tuple can not be changed in any way once it is created.

Characterstics

- Ordered
- Unchangeble
- Allows duplicate
```

Plan of attack

- Creating a Tuple
- Accessing items
- Editing items
- Adding items
- Deleting items
- Operations on Tuples
- Tuple Functions

```
# This is formatted as code
```

Creating Tuples

```
#empty
t1=()
print(t1)

()

#create a tuple with a single item
t2=(2)
t3=("hello")
print(t2)
print(t3)
print(type(t2))
print(type(t3))# banaye top tuple hain but type int aa raha hai ye ek choti se problem hai single item tuple ke sath ise duer
              #karne ke lye last me ek comma add kar dete hain

2
hello
<class 'int'>
<class 'str'>
```

```
t2=(2,)
t3=("hello",)
print(t2)
print(t3)
print(type(t2))
print(type(t3))
```

```
(2,)
('hello',)
<class 'tuple'>
<class 'tuple'>
```

```
#homogeneous tuple
t3=(1,2,3,4,5)
print(t3)
```

```
(1, 2, 3, 4, 5)
```

```
#hetrogeneous tuple
t4=(1,"hii",2.5,True,[2,"jay"])
print(t4)
```

```
(1, 'hii', 2.5, True, [2, 'jay'])
```

```
# tuple
t5 = (1,2,3,(4,5))
print(t5)
```

```
(1, 2, 3, (4, 5))
```

```
#using type conversion
t6=tuple("hello")
print(t6)
```

```
('h', 'e', 'l', 'l', 'o')
```

Accessing Items

- Indexing
- Slicing

```
# isme bhi +ve and -ve index ka concept hai
print(t3)
print(t3[0])
print(t3[-1])
```

```
(1, 2, 3, 4, 5)
1
5
```

```
t5[-1][0]
```

```
4

#slicing=> completely similer as list and string
print(t3[::-1])

(5, 4, 3, 2, 1)
```

Editing items

```
print(t3)
t3[0]=100
# immutable just like string

(1, 2, 3, 4, 5)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[72], line 2
      1 print(t3)
----> 2 t3[0]=100

TypeError: 'tuple' object does not support item assignment
```

Adding items

```
print(t3)
#not possible because of immutable

(1, 2, 3, 4, 5)
```

Deleting items

```
print(t3)
del t3
print(t3) #qki t3 delete ho chuka hai

(1, 2, 3, 4, 5)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[80], line 3
      1 print(t3)
```

```
      2 del t3
----> 3 print(t3)

NameError: name 't3' is not defined
```

```
print(t4)
del t4[3] # kisi paricular element ko delete nahi kar sakte hain but complete tuple ka deletion possible hai
```

```
(1, 'hii', 2.5, True, [2, 'jay'])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[86], line 2
      1 print(t4)
----> 2 del t4[3]

TypeError: 'tuple' object doesn't support item deletion
```

```
t=(1,2,3,4,5)
t[-1:-4:-1]
```

```
(5, 4, 3)
```

operations on Tuples

```
# + and *
t1 = (1,2,3,4)
t2 = (5,6,7,8)

print(t1 + t2)

print(t1*3)

# membership
8 in t1

# iteration
for i in t1:
  print(i)
```

```
(1, 2, 3, 4, 5, 6, 7, 8)
(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
1
2
3
4
```

```
1 in t1
```

```
True
```

Tuple function

```
# len/sum/min/max/sorted => sab kuch list ke tarah hi
t = (1,2,3,4)
print(len(t))

print(sum(t))

print(min(t))

print(max(t))

print(sorted(t))

print(sorted(t,reverse=True))
```

```
4
10
1
4
[1, 2, 3, 4]
[4, 3, 2, 1]
```

```
# count

t = (1,2,3,4,5)

t.count(50)
```

```
0
```

```
# index
t.index(4)
```

```
3
```

Difference between Lists and Tuples

- Syntax=> tuple me small bracket use hota hai jabki list me square
- Mutability => tuples are immutable
- Speed => tuple are faster (immutable generally mutable se fast hota hai)
- Memory =>tuple kam memory leta hai list ke compare me(immutable generally mutable se kam memeory leta hai)
- Built in functionality =>list me jyada inbuilt function hota hai compare to tuple
- Error prone =>list me jyada dikkat hota hai error find karne me

- Usability =>aaise application jo sirf read only hai waha hum tuple ka use karenge and application jo user friendly hai jaise todo to waha hum list ka use karte hain

```
import time

L = list(range(100000000))
T = tuple(range(100000000))

start = time.time()
for i in L:
  i*5
print('List time',time.time()-start)

start = time.time()
for i in T:
  i*5
print('Tuple time',time.time()-start)
```

```
List time 15.388324975967407
Tuple time 15.264079809188843
```

```
import sys

L = list(range(1000))
T = tuple(range(1000))

print('List size',sys.getsizeof(L))
print('Tuple size',sys.getsizeof(T))
```

```
List size 8056
Tuple size 8040
```

```
a = [1,2,3]
b = a

a.append(4)
print(a)
print(b)
# list me dikkat hai ye
```

```
[1, 2, 3, 4]
[1, 2, 3, 4]
```

```
a = (1,2,3)
b = a

a = a + (4,)
print(a)
print(b)
```

```
(1, 2, 3, 4)
(1, 2, 3)
```

why use tuple

```
# This is formatted as code
```

Special Syntax

```
# tuple unpacking
a,b,c = (1,2,3)
print(a,b,c) # tino variable ko ek ek value mil jayega
print(type(a))
```

```
1 2 3
<class 'int'>
```

```
a,b = (1,2,3)
print(a,b) # tin value hai but do hi variable so error aayega
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[131], line 1
----> 1 a,b = (1,2,3)
      2 print(a,b)

ValueError: too many values to unpack (expected 2)
```

```
#swap karne ka ninja tarika
a=1
b=2
a,b=b,a
print(a,b)
```

```
2 1
```

```
a,b,*others=(1,2,3,4,5)
print(a,b)
print(others)  # a,b me first 2 elements chala jayega baki sab others me list formet me store ho jayega
```

```
1 2
[3, 4, 5]
```

```
# zipping tuples
a=(1,2,3,4,5)
b=(6,7,8)
zip(a,b)
```

```
<zip at 0x2207ed86800>
```

```
a=(1,2,3,4,5)
b=(6,7,8)
```

```
print(tuple(zip(a,b)))
print(list(zip(a,b)))
```

```
((1, 6), (2, 7), (3, 8))
[(1, 6), (2, 7), (3, 8)]
```

tuple comprehensive bhi possible hai and simply tuple list ka bhai hai sab kuch list jaisa hi hai

Sets

- similar as maths

A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed).

However, a set itself is mutable. We can add or remove items from it.

Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.

Characterstics:

- Unordered => {1,2,3}={3,2,1} (ye aur kisi data type me possible nahi tha)
- Mutable => sets are mutable
- No Duplicates => duplicate element nahi ho sakta hai
- Can't contain mutable data types => set khud mutable hota hai lekin uske andar kisi mutable data types ko nahi rakh sakte ho (so 2D sets are not possible)

creating sets

```
# empty
s={}
print(s)
print(type(s)) #by default dictionary type hota hai qki dono same syntax share karta hai
```

```
{}
<class 'dict'>
```

```
s=set()
print(s)
print(type(s))
```

```
set()
<class 'set'>
```

```python
# 1D and 2D
s1={1,2,3}
print(s1)
```

```
{1, 2, 3}
```

```python
s2={1,2,3,{3,4}}
print(s2) #not possible 2D sets
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[155], line 1
----> 1 s2={1,2,3,{3,4}}
      2 print(s2)

TypeError: unhashable type: 'set'
```

```python
#homo and hetro
s3={1,2,"jay",4.5,True} # basically python True ko 1 consider karta hai and already 1 available hai set me so True print nahi hua hai output me
print(s3)
```

```
{1, 2, 'jay', 4.5}
```

```python
s4={1,"jay",4.5,(1,2,3)} #kabhi kabhi output ka order change bhi ho jata hai ye islye hota hsai qki internally hashing ka algo
print(s4)                  # chal raha hota hai and ispe humara koi control nahi hota hai
```

```
{1, 'jay', 4.5, (1, 2, 3)}
```

```python
#using type conversion
s5=set([1,2,3])
print(s5)
```

```
{1, 2, 3}
```

```python
# duplicates not allowed
s6 = {1,1,2,2,3,3,6,7}
print(s6)
```

```
{1, 2, 3, 6, 7}
```

```python
# set can't have mutable items
s6 = {1,2,[3,4]}
print(s6)
```

```
---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call
last)
Cell In[173], line 2
      1 # set can't have mutable items
----> 2 s6 = {1,2,[3,4]}
      3 print(s6)

TypeError: unhashable type: 'list'
```

```
s1={1,2,3}
s2={3,2,1}
print(s1==s2)
```

```
True
```

Accessing Items

- sets indexing ka koi concept nahi hota hai

```
s1={1,2,3,4}
s1[4]  # no concept of indexing(neither positive not negatoive) and
slicing

### sidhi baat ki Accessing element is not allowed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
Cell In[179], line 2
      1 s1={1,2,3,4}
----> 2 s1[4]

TypeError: 'set' object is not subscriptable
```

Editing Items => bhai jab indexing hi kaam nahi karta hai to Editing kya ghanta karoge so editing is also not allowed

```
s1={1,2,3,4}
s1[2]=100
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
Cell In[182], line 2
      1 s1={1,2,3,4}
----> 2 s1[2]=100
```

```
TypeError: 'set' object does not support item assignment
```

Adding Element

- ye easily kar sakte hain

```
#add => kaha add hoga koi fix nahi hai qki hashing ke through element
add hota hai
s={1,2,3,4}
s.add(5)
print(s)
```

```
{1, 2, 3, 4, 5}
```

```
#update  => ek sath multiple element ko add kar sakte hain
s={1,2,3,4}
s.update([5,6,7])
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7}
```

deleting item

```
#del
s1={1,2,3,4,5}
print(s1)
del s1
print(s1) # delete ho chuka hai to print kaha se hoga
```

```
{1, 2, 3, 4, 5}
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[190], line 5
      3 print(s1)
      4 del s1
----> 5 print(s1)

NameError: name 's1' is not defined
```

```
s1={1,2,3,4,5}
print(s1)
del s1[3] #not allowed qki indexing ka jab koi concept hi nahi hai to
kisi particular ko index ke help se kaise delete karoge
```

```
{1, 2, 3, 4, 5}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[194], line 3
      1 s1={1,2,3,4,5}
      2 print(s1)
----> 3 del s1[3]

TypeError: 'set' object doesn't support item deletion
```

```
#discard =>kisi particular element ko deletee karna
s1={1,2,3,4,5}
print(s1)
s1.discard(4) #jis element ko delete karna hai wo as a parameter pass kar do
print(s1)
```

```
{1, 2, 3, 4, 5}
{1, 2, 3, 5}
```

```
s1.discard(34)  #koi error throw nahi karega jab hum kisi aaise element ko delete karna chahenge jo ki available hi nahi hai
print(s1)
```

```
{1, 2, 3, 5}
```

```
#remove => ye bhi element ko delete kar dega difference bas itna ki jab hum wo value as a parameter pass
s1={1,2,3,4,5}                    # karenge jo ki availabe hi nahi hai to ye error dega jabki discard error nahi deta hai
s1.remove(5)
print(s1)
```

```
{1, 2, 3, 4}
```

```
s1.remove(12)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[7], line 1
----> 1 s1.remove(12)

KeyError: 12
```

```
#pop => ye rendomly kisi ek element ko delete kar deta hai
s1={1,2,3,4,5}
s1.pop()
```

```
1
```

```
s1={1,2,3,4,5}
s1.pop()
print(s1)
```

```
{2, 3, 4, 5}
```

```
#clear => ye set ke sare element ko delete kar deta hai
s1={1,2,3,4,5}
s.clear()
print(s)
```

```
set()
```

Set Functions

```
# len/sum/min/max/sorted
s = {3,1,4,5,2,7}
print(len(s))
print(sum(s))
print(min(s))
print(max(s))
print(sorted(s))
print(sorted(s,reverse=True))
```

```
6
22
1
7
[1, 2, 3, 4, 5, 7]
[7, 5, 4, 3, 2, 1]
```

```
#union/update
s1={1,2,3,4,5}
s2={4,5,6,7,8}
print(s1.union(s2))
print(s1)
print(s2)
#update
print()
print(s1.update(s2)) #permanent chane kar deta hai s1 ko s2 ka content
s1 ke sath aa jata hai and s1 update ho jata hai
print(s1)
print(s2)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5}
```

```
{4, 5, 6, 7, 8}

None
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5, 6, 7, 8}
```

```
# intersection/intersection_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

print(s1.intersection(s2))
print(s1)
print(s2)
print()
s1.intersection_update(s2)  # s1 ko permanent change kar dega
print(s1)
print(s2)
```

```
{4, 5}
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}

{4, 5}
{4, 5, 6, 7, 8}
```

```
# difference/difference_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

print(s1.difference(s2))
print(s1)
print(s2)
print()

s1.difference_update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3}
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}

{1, 2, 3}
{4, 5, 6, 7, 8}
```

```
# symmetric_difference/symmetric_difference_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

print(s1.symmetric_difference(s2))
print(s1)
```

```
print(s2)
print()
s1.symmetric_difference_update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}

{1, 2, 3, 6, 7, 8}
{4, 5, 6, 7, 8}
```

```
#isdisjoint/issubset/issuperset

s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}
s1.isdisjoint(s2)  #isdisjoint=>koi bhi common nahi hona chahaiye tab
true agar ek bhi common to false
```

```
False
```

```
#issubset  => check karta hai ki subset hai ki nahi
s1={1,2,3,4,5}
s2={3,4,5}
print(s1.issubset(s2))
print(s2.issubset(s1))
```

```
False
True
```

```
#issuperset  => check karta hai ki superset hai ki nahi
s1={1,2,3,4,5}
s2={3,4,5}
print(s1.issuperset(s2))
print(s2.issuperset(s1))
```

```
True
False
```

```
#copy => ek shallow copy create karta hai
s1 = {1,2,3}
s2 = s1.copy()

print(s1)
print(s2)
```

```
{1, 2, 3}
{1, 2, 3}
```

Set Operations

```
#Union(|)
#intersection(&)
#Difference(-)
#Symmetric Difference(^)
#Membership Test
#Iteration

#Union(|)
s1={1,2,3,4,5}
s2={4,5,6,7,8}
s1|s2
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
#intersection(&)
s1={1,2,3,4,5}
s2={4,5,6,7,8}
s1&s2
```

```
{4, 5}
```

```
s1={1,2,3,4,5}
s2={4,5,6,7,8}
print(s1-s2)
print(s2-s1)
```

```
{1, 2, 3}
{8, 6, 7}
```

```
s1={1,2,3,4,5}
s2={4,5,6,7,8}
s1^s2
```

```
{1, 2, 3, 6, 7, 8}
```

```
s1={1,2,3,4,5}
s2={4,5,6,7,8}
1 not in s1
```

```
False
```

```
for i in s1:
    print(i)
```

```
1
2
3
4
5
```

Frozenset

Frozen set is just an immutable version of a Python set object

```
fs=frozenset([1,2,3])
fs
```

```
frozenset({1, 2, 3})
```

```
#union
fs1=frozenset([1,2,3])
fs2=frozenset([3,4,5])
fs1|fs2
```

```
frozenset({1, 2, 3, 4, 5})
```

```
# what works and what does not
# works -> all read functions
# does't work -> write operations

#2d frozenset is possible
fs=frozenset([1,2,frozenset([3,4])])
fs
```

```
frozenset({1, 2, frozenset({3, 4})})
```

Set comprehension

```
#example
{i for i in range(1,11)}
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
{i for i in range(1,11) if i>5}
```

```
{6, 7, 8, 9, 10}
```

```
{i**2 for i in range(1,11) if i>5}
```

```
{36, 49, 64, 81, 100}
```

Dictionary

Dictionary in Python is a collection of keys & values, used to store data values like a map, which, unlike other data types which hold only a single value as an element.

In some languages it is known as map or assosiative arrays.

dict = { 'name' : 'nitish' , 'age' : 33 , 'gender' : 'male' }

Characterstics:

- Mutable
- Indexing has no meaning
- keys can't be duplicated
- keys can't be mutable items

create dictionary

```
# empty dictionary

# 1D dictionary

# with mixed keys

# 2D dictionary -> JSON

# using sequence and dict function

# duplicate keys

# mutable items as keys

# empty dictionary
d={}
d
```

```
{}
```

```
# 1D dictionary
dict = { 'name' : 'nitish' ,  'gender' : 'male' }
dict
```

```
{'name': 'nitish', 'gender': 'male'}
```

```
# with mixed keys
dict = { 'name' : 'nitish' , 'age' : 33 , 'gender' : 'male' }
dict
```

```
{'name': 'nitish', 'age': 33, 'gender': 'male'}
```

```
dict={(1,2,3):1,"hello":"world"}
dict
```

```
{(1, 2, 3): 1, 'hello': 'world'}
```

```
# 2D dictionary -> JSON
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
```

```
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
s
```

```
{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'maths': 67, 'english': 34}}
```

```
# using sequence and dict function
d4 = dict([('name','nitish'),('age',32),(3,3)])
print(d4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[110], line 2
      1 # using sequence and dict function
----> 2 d4 = dict([('name','nitish'),('age',32),(3,3)])
      3 print(d4)

TypeError: 'dict' object is not callable
```

```
# duplicate keys => not allowed agar hai to jo key value pair sabse last me hai wo output ho jayega
d5 = {'name':'nitish','name':'rahul'}
d5
```

```
{'name': 'rahul'}
```

```
# mutable items as keys  =>means list are not allowed as key
d6 = {'name':'nitish',(1,2,3):2}
print(d6)
```

```
{'name': 'nitish', (1, 2, 3): 2}
```

```
d6 = {'name':'nitish',[1,2,3]:2}
print(d6)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[116], line 1
----> 1 d6 = {'name':'nitish',[1,2,3]:2}
      2 print(d6)
```

```
TypeError: unhashable type: 'list'
```

Accessing items

```
my_dict={"name":"jack","age":26}
my_dict[0]                  # indexing not allowed
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[120], line 2
      1 my_dict={"name":"jack","age":26}
----> 2 my_dict[0]

KeyError: 0
```

```
#[]
my_dict={"name":"jack","age":26}
my_dict["name"]     # hum square bracket me key pass karke data access kar sakte hain
```

```
'jack'
```

```
my_dict={"name":"jack","age":26}
my_dict["name","age"]     # multiple key pass nahi kar sakte hain
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[124], line 2
      1 my_dict={"name":"jack","age":26}
----> 2 my_dict["name","age"]

KeyError: ('name', 'age')
```

```
#get
my_dict={"name":"jack","age":26}
my_dict.get("name")
```

```
'jack'
```

Adding key-value pair

```
my_dict={"name":"jack","age":26}
print(my_dict)
```

```
my_dict["gender"]="male"
print(my_dict)
my_dict["weight"]=72
print(my_dict)
```

```
{'name': 'jack', 'age': 26}
{'name': 'jack', 'age': 26, 'gender': 'male'}
{'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
```

remove key-value pair

```
#pop
#popitem
#del
#clear

#pop
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
d.pop("age")          # jis bhi key ko delete karna hai wo key pass
kar do
print(d)
```

```
{'name': 'jack', 'gender': 'male', 'weight': 72}
```

```
#popitem => last bale key value pair ko delete karta hai
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
d.popitem()

print(d)
```

```
{'name': 'jack', 'age': 26, 'gender': 'male'}
```

```
#del
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
del d
print(d) # complete d hi delete ho gya
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[152], line 4
      2 d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
      3 del d
----> 4 print(d)

NameError: name 'd' is not defined
```

```
#kisi specific ko bhi del ke help se delete kar sakte hain bass key pass kar do
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
del d["name"]
print(d)
```

```
{'age': 26, 'gender': 'male', 'weight': 72}
```

```
#clear
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
d.clear()
print(d) # pura khali ho jayega
```

```
{}
```

```
# accessing in 2D
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
s["subjects"]["maths"]
```

```
67
```

```
#Adding new in 2D
s["subjects"]["computer"]=87
print(s)
```

```
{'name': 'nitish', 'college': 'bit', 'sem': 4, 'subjects': {'dsa': 50, 'maths': 67, 'english': 34, 'computer': 87}}
```

```
# deleting in 2D => similar as
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
del s["subjects"]["maths"]
print(s)
```

```
{'name': 'nitish', 'college': 'bit', 'sem': 4, 'subjects': {'dsa': 50,
'english': 34}}

#Editing key value
s

{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'english': 34}}

s["sem"]=5
s

{'name': 'nitish',
 'college': 'bit',
 'sem': 5,
 'subjects': {'dsa': 50, 'english': 34}}

s["subjects"]["dsa"]=88
s

{'name': 'nitish',
 'college': 'bit',
 'sem': 5,
 'subjects': {'dsa': 88, 'english': 34}}
```

Dictionary Operation

- membership
- Iteration

```
# memebership
print(s)
print("nitish" in s)  # false qki hum value se kuch kaam nahi karte
hain sab kuch key se karte hain
print("name" in s)

{'name': 'nitish', 'college': 'bit', 'sem': 5, 'subjects': {'dsa': 88,
'english': 34}}
False
True

#iteration
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
for i in d:
    print(i)   # sara key mil jayega matlab i me key iterate kar raha
hai
```

```
name
age
gender
weight

#thoda dimag laga ke value bhi print kara sakte hain
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
for i in d:
    print(i,":",d[i])

name : jack
age : 26
gender : male
weight : 72
```

dictionary function

```
#len/sorted
d={'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}
len(d)

4

max(d) #ye key find karke dega ascii ke according

'weight'

 min(d)

'age'

sorted(d)

['age', 'gender', 'name', 'weight']

sorted(d,reverse=True)

['weight', 'name', 'gender', 'age']

#item/keys/values
print(d)

{'name': 'jack', 'age': 26, 'gender': 'male', 'weight': 72}

#items
print(d.items()) # sare key value ko tuples ke form me print kara deta
hai

dict_items([('name', 'jack'), ('age', 26), ('gender', 'male'),
('weight', 72)])
```

```
#keys
print(d.keys())  # sare keys ko print kara deta hai
```

```
dict_keys(['name', 'age', 'gender', 'weight'])
```

```
#value
print(d.values())  # sare ke sare value print ho jayega
```

```
dict_values(['jack', 26, 'male', 72])
```

```
#update
d1={1:2,2:3,4:5}
d2={4:7,6:8}
d1.update(d2)  # given dictionary ko kisi dusri dictionary se uodate kar sakte ho and ye permanent change hota hai
print(d1)
```

```
{1: 2, 2: 3, 4: 7, 6: 8}
```

Dictionary Comprehension

> Add blockquote

```
{ key: value for vars in iterable }
```

```
## print 1st 10 numbers and their squares
{i:i**2 for i in range(1,11)}
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

```
# using existing dict
distances = {'delhi':1000,'mumbai':2000,'bangalore':3000}  # distance km me hai ise miles me convert karna hai
{key:value*0.62 for (key,value) in distances.items()}
```

```
{'delhi': 620.0, 'mumbai': 1240.0, 'bangalore': 1860.0}
```

```
#using zip
days = ["Sunday",
"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
temp_C = [30.5,32.6,31.8,33.4,29.8,30.2,29.9]

{i:j for (i,j) in zip(days,temp_C)}
```

```
{'Sunday': 30.5,
 'Monday': 32.6,
 'Tuesday': 31.8,
 'Wednesday': 33.4,
 'Thursday': 29.8,
```

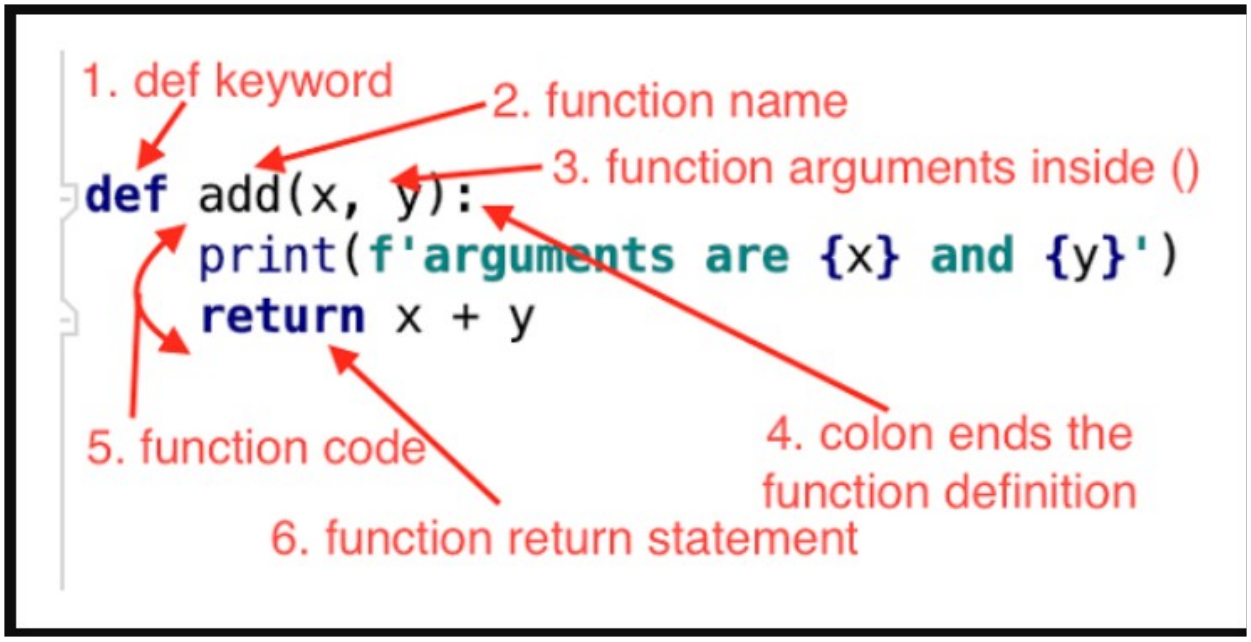
```
 'Friday': 30.2,
 'Saturday': 29.9}

#using if condition
products = {'phone':10,'laptop':0,'charger':32,'tablet':0}
{key:value for (key,value) in products.items() if value>0}

{'phone': 10, 'charger': 32}

# Nested Comprehension
# print tables of number from 2 to 4
{i:{j:i*j for j in range(1,11) }for i in range(2,5) }

{2: {1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18, 10:
20},
 3: {1: 3, 2: 6, 3: 9, 4: 12, 5: 15, 6: 18, 7: 21, 8: 24, 9: 27, 10:
30},
 4: {1: 4, 2: 8, 3: 12, 4: 16, 5: 20, 6: 24, 7: 28, 8: 32, 9: 36, 10:
40}}
```

#required output { 2:{1:2,2:4,3:6,4:8,...}, 3:{1:3,2:6,3:9,4:12}, 4:{1:4,2:8,3:12,4:16} }

functions

Let's create a function(with docstring)

for syntax of python just click the link



```python
#creating a function
def is_even(num):
    """
     ye doc string hai isme generally ye likhte hain ki function karta kya hai
     ex:this function returns if a given number is odd or even
     input- any valid integer
     output-odd/even
     created on-28th jan 2025
    """
    if type(num)==int:

        if num%2==0:
            return "even"
        else:
            return "odd"

    else:
        return "pagal hai kya"




#call the function
#syntax: function_name(input)
for i in range(1,11):
    x=is_even(i)  #calling
    print(i," is ",x)
```

```
1  is  odd
2  is  even
3  is  odd
4  is  even
5  is  odd
6  is  even
7  is  odd
8  is  even
9  is  odd
10  is  even
```

```python
#fatching the documentation
print(is_even.__doc__)  #=> bracket ke andar likho--> (function_Name dot double Underscore doc double underscore)
```

```
     ye doc string hai isme generally ye likhte hain ki function karta kya hai
     ex:this function returns if a given number is odd oe even
     input- any valid integer
     output-odd/even
```

```
    created on-28th jan 2025


print(print.__doc__)

Prints the values to a stream, or to sys.stdout by default.

  sep
    string inserted between values, default a space.
  end
    string appended after the last value, default a newline.
  file
    a file-like object (stream); defaults to the current sys.stdout.
  flush
    whether to forcibly flush the stream.
```

Arguement Vs Parameter

jab hum function ko call karte time value pass karte hain i.e.arguement and function jo accept karta hai i.e parameter

Types of Arguments

- Default Argument
- Positional Argument
- Keyword Argument

```
#Default Argument => yadi a ko value nn mile to assume kar lo 1 mil gya qki default set kar dye hain
def power(a=1,b=1):
    return a**b
```

1. `def power(a=1,b=1):` - This line defines a function named `power` that takes two parameters, `a` and `b`.
2. Both parameters have default values of `1` - this means if the function is called without specifying these arguments, they will automatically be assigned the value `1`.
3. The comment above the function explains the purpose of default arguments in Hindi/English: "Default Argument => if a value is not provided, assume it's 1 because defaults have been set"

```
power(2,3)

8

power(2)

2
```

Positional Argument

- pahla arguement pahle parameter ko and dusra argument dusre parameter ko milta hai and so on.... `Positional Arguments`
- You pass arguments in order, and they get assigned to parameters by position.

```
def abc(a,b):
    return a**b

print(abc(2,3))
```

```
8
```

```
#Keyword Argument => explicitely a and b ko value provide karna qki jaruri nahi ki aapko position yaad rahe ki kon sa
power(b=3,a=2)            # parameter  kis order me hai islye keyword (name of parameter) ke help  se value pass kar do
```

```
8
```

The example `power(b=3,a=2)` demonstrates using keyword arguments to call a function named `power` where parameter `b` is assigned the value 3 and parameter `a` is assigned the value 2, regardless of their original order in the function definition.

`*args and **kwargs`

`*args and **kwargs` are special Python keywords that are used to pass the variable length of arguments to a function

```
# *args
# allows us to pass a variable number of non-keyword arguments to a function.
def multiply(*args):      #=> ye abb kitnee bhi argument ko haldle kar sakta hai (jaruri nahi ki args hi likhe kuch bhi name likh sakte hai sirf name
    product=1                           # ke aage single star hona chahaiye but generally hum args likhte hain)
                           #=> internally args me  jitna bhi argument aata hai sabko mila ke ek tuples create kar deta hai

    for i in args:
        product=product*i
    print("proof of ki tuples banta hai")
    print(args)
    print(type(args))
    return product

multiply(2,4,5)
```

```
proof of ki tuples banta hai
(2, 4, 5)
<class 'tuple'>

40

def add(*args):
    sum = 0
    for i in args:
        sum = sum+i
    print(sum)

add(1,2,3)

6

# **kwargs
# **kwargs allows us to pass any number of keyword arguments.
# Keyword arguments mean that they contain a key-value pair, like a
Python dictionary.

def display(**kwargs):   #kwargs dictionary ban jayege and kwargs hi
name ho ye bhi compulsary nahi hai kuch bhi name ho sakta hai bass
name ke pahle
    for (key,value) in kwargs.items():
# double star hona chahaiye
        print(key,"->",value)


display(india="delhi",srilanka="colombo",nepal="kathmandu")

india -> delhi
srilanka -> colombo
nepal -> kathmandu
```

Points to remember while using `*args and **kwargs`

- order of the arguments matter(normal -> `*args` -> `**kwargs`) yadi tino tarah ke arguement use kar rahe hain to pahle priority : normal,*args*,kwargs
- The words “args” and “kwargs” are only a convention, you can use any name of your choice

How Functions are executed in memory?

```
#ek baar simply python tutor pe function ko execute karke dekho feel
aa jayega
#agar koi puch le what is the life sapn of a variable of a function =>
to jitnaa function ka life span hai(call hone se return tak)
```

Without return Statement

```
#default return type none hota hai
def is_even(num):
    """
     ye doc string hai isme generally ye likhte hain ki function karta
kya hai
     ex:this function returns if a given number is odd oe even
     input- any valid integer
     output-odd/even
     created on-28th jan 2025
    """
    if type(num)==int:

        if num%2==0:
            print( "even")
        else:
            print( "odd")

    else:
        print( "pagal hai kya")

is_even(7)
```

```
odd
```

```
print(is_even(7)) # yaha jab call hoga to uske baad usko jo value
milega wo none milega
```

```
odd
None
```

```
l=[1,2,3]
print(l.append(4)) #none islye qki append function kuch bhi return
nahi karta hai so by default return type is none
```

```
None
```

```
l
```

```
[1, 2, 3, 4]
```

Variable Scope

```
def g(y):
    print(x)
    print(x+1)
x = 5
g(x)
print(x)  # function ke andar ka variable local and main function ke
andar ka variable globle(basically main function ka koi concept hai
```

```
nahi
        # python me but samajhne ke lye) and main function ke variable
ko local bala use kar sakta hai lekin local bale ko main use nahi kar
sakta hai
```

```
5
6
5
```

```
def f(y):
    x = 1
    x += 1
    print(x)
    print(y)
x = 5
f(x)
print(x) # dono function differently behave karta hai
```

```
2
5
5
```

```
def h(y):
    x += 1
x = 5
h(x)
print(x) #yadi function ke andar variable availabe nahi hai to wo sirf
global bale ko use kar sakta hai usme koi update nahi kar sakta hai
```

```
---------------------------------------------------------------------
-----
UnboundLocalError                         Traceback (most recent call
last)
Cell In[38], line 4
      2     x += 1
      3 x = 5
----> 4 h(x)
      5 print(x) #yadi function ke andar variable availabe nahi hai to
wo sirf global bale ko use kar sakta hai usme koi update nahi kar
sakta hai

Cell In[38], line 2, in h(y)
      1 def h(y):
----> 2     x += 1

UnboundLocalError: cannot access local variable 'x' where it is not
associated with a value
```

```
def h(y):
    global x  #abb error nahi dega ye batane ka terika hai ki hum
global me change kar rahe hain butt ye good practice nahi hai
```

```
    x += 1
x = 5
h(x)
print(x)
```

```
6
```

```
def f(x):
   x = x + 1
   print('in f(x): x =', x)
   return x

x = 3
z = f(x)
print('in main program scope: z =', z)
print('in main program scope: x =', x)
```

```
in f(x): x = 4
in main program scope: z = 4
in main program scope: x = 3
```

Nested Function

```
def f():
    def g():
        print("inside function g")
    g()
    print("inside function f")



f()
```

```
inside function g
inside function f
```

```
g() #error qki direct g ko call kar hi nahi sakte hain
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[54], line 1
----> 1 g() #error qki direct g ko call kar hi nahi sakte hain

NameError: name 'g' is not defined
```

```
def f():
    def g():
```

```
        print("inside function g")
        f()
    g()
    print("inside function f")

f() # yaha recursion ho gya or can say infinite loop chal gya
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
```

```
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g
inside function g

---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
Cell In[31], line 1
----> 1 f()

Cell In[29], line 5, in f()
      3     print("inside function g")
      4     f()
----> 5 g()
      6 print("inside function f")

Cell In[29], line 4, in f.<locals>.g()
      2 def g():
      3     print("inside function g")
----> 4     f()

Cell In[29], line 5, in f()
      3     print("inside function g")
      4     f()
----> 5 g()
      6 print("inside function f")

Cell In[29], line 4, in f.<locals>.g()
      2 def g():
      3     print("inside function g")
----> 4     f()
```

```
    [... skipping similar frames: f at line 5 (1485 times),
f.<locals>.g at line 4 (1484 times)]

Cell In[29], line 4, in f.<locals>.g()
      2 def g():
      3     print("inside function g")
----> 4     f()

Cell In[29], line 5, in f()
      3     print("inside function g")
      4     f()
----> 5 g()
      6 print("inside function f")

Cell In[29], line 3, in f.<locals>.g()
      2 def g():
----> 3     print("inside function g")
      4     f()

File ~\anaconda3\Lib\site-packages\ipykernel\iostream.py:649, in
OutStream.write(self, string)
    646     msg = "I/O operation on closed file"
    647     raise ValueError(msg)
--> 649 is_child = not self._is_master_process()
    650 # only touch the buffer in the IO thread to avoid races
    651 with self._buffer_lock:

RecursionError: maximum recursion depth exceeded
```

```python
#give output
def g(x):
    def h():
        x = 'abc'
        #print(x)
    x = x + 1
    print('in g(x): x =', x)
    h()
    return x

x = 3
z = g(x)
print(z)
```

```
in g(x): x = 4
4
```

```python
def g(x):
    def h(x):
        x = x+1
        print("in h(x): x = ", x)
```

```
    x = x + 1
    print('in g(x): x = ', x)
    h(x)
    return x

x = 3
z = g(x)
print('in main program scope: x = ', x)
print('in main program scope: z = ', z)
```

```
in g(x): x =  4
in h(x): x =  5
in main program scope: x =  3
in main program scope: z =  4
```

Functions are 1st class citizens

- o bhai function data type ke jaisa kaam karta hai python me ye wo sab kuch kar sakta hai jo ki list, tuples wager karta hai

```
# function in python is like a data type ye bilkul data type ke jaisa
hi kaam karta hai
def square(num):
    return num**2

print(type(square))
print(id(square))
```

```
<class 'function'>
2499084830496
```

```
#reassign
x=square
print(id(x))
print(x(3))
```

```
2499084830496
9
```

```
#deleting a function
del square

square(3) #qki delete ho chuka hai
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[50], line 1
----> 1 square(3)

NameError: name 'square' is not defined
```

```
def square(num):
    return num**2

#storing
L=[1,2,3,4,square]
print(L[-1])
print(L[-1](3))
```

```
<function square at 0x0000015FDB1BF9C0>
9
```

```
# question is ki agar function datatype ke jaisa kaam kar raha hai to
to ye mutable hai ki immutable
s={square}  # set mutable ko allow nahi karta hai to agar ye code run
kar gya matlab function immutable data type ke tarah kaam karta hai
s             # yes run kar gya matlab function immutable data type ke
tarah kaam karta hai
```

```
{<function __main__.square(num)>}
```

```
#returning a function => agar aap chaho to ek function ke through ek
function ko bhi return kar sakte ho
def f():
    def x(a, b):
        return a+b
    return x

val = f()(3,4)
print(val)
```

```
7
```

```
# function as argument

def func_a():
    print('inside func_a')

def func_b(z):
    print('inside func_d')
    return z()

print(func_b(func_a))
```

```
inside func_d
inside func_a
None
```

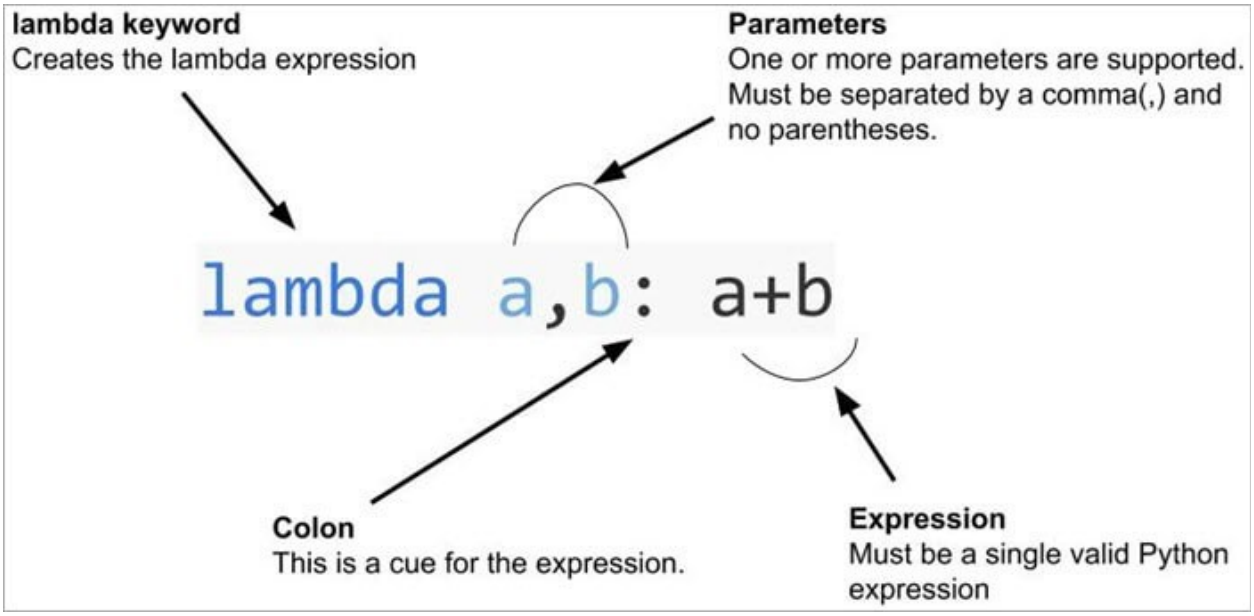
Benefits of using a Function

- Code Modularity
- Code Readibility
- Code Reusability

Lambda Function

A lambda function is a small anonymous(matlab ki iska koi name nahi hota hai) function.

A lambda function can take any number of arguments, but can only have one expression.




```
# x-> x^2
a=lambda x : x**2
a(2)
```

```
4
```

```
#x,y -> x+y
a=lambda x,y:x+y
a(5,2)
```

```
7
```

Diff between lambda vs Normal Function

- No name
- lambda has no return value(infact,returns a function)
- lambda is written in 1 line
- not reusable

Then why use lambda functions? **They are used with HOF**(Heigher order function)

```
# check if a string has 'a'
a=lambda s :'a' in s
a("hello") # ye hello basically s ko milega
```

```
False
```

```
# odd or even
a=lambda x : 'even' if x%2==0 else 'odd'
a(7)
```

```
'odd'
```

heigher order function

- aaisa function jo khud ek function ko return kare heigher order function kahlata hai yaa aaisa function jo ki input me kisis dusre function ko receive kare

```
#Example

def square(x):
  return x**2

#HOF
def transform(f,L):
  output = []
  for i in L:
    output.append(f(i))

  print(output)

L=[1,2,3,4,5]
transform(square,L)
```

```
[1, 4, 9, 16, 25]
```

```
def square(x):
  return x**2

def cube(x):
  return x**3

# HOF
def transform(f,L):
  output = []
  for i in L:
    output.append(f(i))

  print(output)

L = [1,2,3,4,5]

transform(lambda x:x**3,L)
transform(lambda x:x**2,L)
```

```
[1, 8, 27, 64, 125]
[1, 4, 9, 16, 25]
```

Map

- ye ek lambda function and ek list accept karta hai matlab isse do arguement chahaiye

```
# square the item of a list
map(lambda x:x**2,[1,2,3,4]) #abhi sirf address dikhega but isse list
me type cast karke dekh sakte hain
```

```
<map at 0x245de13c100>
```

```
list(map(lambda x:x**2,[1,2,3,4]))
```

```
[1, 4, 9, 16]
```

```
#Odd/even labeling of list items
list(map(lambda x : "even" if x%2==0 else "odd",[1,2,3,4]))
```

```
['odd', 'even', 'odd', 'even']
```

```
# fetch names from a list of dict

users = [
    {
        'name':'Rahul',
        'age':45,
        'gender':'male'
    },
    {
        'name':'Nitish',
        'age':33,
        'gender':'male'
    },
    {
        'name':'Ankita',
        'age':50,
        'gender':'female'
    }
]

print(list(map(lambda users:users['gender'],users)))
print(list(map(lambda users:users['name'],users)))
```

```
['male', 'male', 'female']
['Rahul', 'Nitish', 'Ankita']
```

Filter

- list me se kuch find karke dega

```
# numbers greater than 5
L = [3,4,5,6,7]
```

```
list(filter(lambda x:x>5,L))
```

```
[6, 7]
```

```
# fetch fruits starting with 'a'
fruits = ['apple','guava','cherry']
list(filter(lambda x:x.startswith('a'),fruits))
```

```
['apple']
```

Reduce

- ise use karne ke lye ek module ka use karna hoga that is "functoolos"
- ye reduce kar deta hai
- ye bhi lambda function and list as arguement leta hai

```
#sum of all item
import functools

functools.reduce(lambda x,y : x+y ,[1,2,3,4,5]) # do do ko uthathe
jayega and add karte jayega
```

```
15
```

```
#find min
functools.reduce(lambda x,y:x if x<y else y,[23,11,45,10,1])
```

```
1
```

```
#find max
functools.reduce(lambda x,y:x if x>y else y,[23,11,45,10,1])
```

```
45
```

OOPs (Object Oriented Programming)

- everything in python is object

```
l=[1,2,3]
l.upper() #qki list me koi upper function nahi hota hai
```

```
---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
```

```
Cell In[5], line 2
      1 l=[1,2,3]
----> 2 l.upper()

AttributeError: 'list' object has no attribute 'upper'
```

```
s="hello"
s.append("x")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[8], line 2
      1 s="hello"
----> 2 s.append("x")

AttributeError: 'str' object has no attribute 'append'
```

oop ka use karke programmer khud ka apna data type bana sakta hai

object and class

- python me do tarah ka class hota hai (i) inbuilt (ii) user defined

```
l=[1,2,3]
print(type(l))
```

```
<class 'list'>
```

```
# jo bhi data type (like int ,list ,float,set etc) hai wo sab kuch python me ek class hai lekin jab bhi in data types ka ek variable banate hain to
#usse ek us class ka object samajhta hai
# class is basically a blueprint jo ye batata hai ki object ko kya kya rules follow karna hai (ex :) cars are class but maruti car is an object)

# class ke andar data and function loikha jata hai [ class :=> a. Data or property , b. Function or behaviour]

# syntax to create an object

# objectname = classname()
```

```
# Object literals
l=[1,2,3]  # yaha l to ek object hai but isme to koi class name nahi hai ( islye qki ye ek data type hai to python ek help kar dya hai ki baar baar
             # class nn likhna pade (isi ko object literals bola jata hai) halanki humwaise bhi bana sakte hain nuche ke example ko dekho)

l=list()  #ye ek list hai and and object creation ke proper syntax ko bhi follow kar raha hai
l
```

```
[]
```

```
s=str()
s
```

```
''
```

```
# class ka name generally Pascal case me likhte hain
```

ATM

```
class Atm:
    # kon kon sa data use hoga => pin , balance

    #constructor => jo bhi data banaoge wo sab constructor ke andar banana hai and jo bhi data ka name likhenge usko self. ke sath likhenge
    def __init__(self): # constructor apne aap call ho jata hai object banate hi
        self.pin=''
        self.balance=0
        #print("mai to execute ho gya")
        self.menu()  # islye taki jaise hi ek object create ho menu function reun ho jaye
        # class me chahe marji jitna bhi code likh lo wo execute nahi hoga unless or untill jabtak ek bhi iss class ka object create nahi kye hai

    def menu(self):
        user_input=input("""
        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. anything else to exit
        """)
```

```
        if user_input=='1':
            #create pin
            self.create_pin()
        elif user_input=='2':
            #change pin
            self.change_pin()

        elif user_input=='3':
            #check balance
            self.check_balance()

        elif user_input=='4':
            #
            self.withdraw()

        else:
            exit()

    def create_pin(self):
        user_pin=input("enter your pin")
        self.pin=user_pin

        user_balance=int(input("enter balance")) #qki abhi humare pass
koi machanism hai nahi paise phase karne ke lye to just mang lye user
se hi
        self.balance=user_balance

        print("pin created successfully")
        self.menu()


    def change_pin(self):
        old_pin=input("enter  old pin")

        if old_pin==self.pin:
            #let him change the pin
            new_pin=input("enter new pin")
            self.pin=new_pin
            print("pin change successfully")
            self.menu()


        else:
            print("golmaal hai bhai golmaal hai !! nahi karne de sakta
re baba")
            self.menu()

    def check_balance(self):
        user_pin=input("enter your pin")
        if user_pin==self.pin:
```

```
            print("balance is  : ",self.balance)
            self.menu()

        else:
            print("chala jaa dikh mat jana dubara")
            self.menu()

    def withdraw(self):
        user_pin=input("enter the pin")
        if user_pin==self.pin:
            #allow to withdraw
            amount=int(input("enter the amount : "))
            if amount<=self.balance:
                self.balance=self.balance-amount
                print("withdrawl successful balance",self.balance)
            else:
                print("sale gareeb")

        else:
            print("bhagg jaa sale chor")
        self.menu()









obj2=Atm()
```

```
        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. anything else to exit
         1
enter your pin 1234
enter balance 10000

pin created successfully


        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
```

```
        5. anything else to exit
         4

print(type(obj))

<class '__main__.Atm'>

obj3=Atm()


        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. anything else to exit
         1
enter your pin 1234
enter balance 10000

pin created successfully


        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. anything else to exit
         4
enter the pin 1234
enter the amount :  10000000

sale gareeb


        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. anything else to exit
         4
enter the pin 1234
enter the amount :  5000

withdrawl successful balance 5000


        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
```

```
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. anything else to exit
         7
```

Methods and Functions

- class ke andar ke function ko method kahate hain

```
l=[1,2,3]
len(l)       # function -> bcos it is outside the list class
l.append()  # method-> bcos it is inside the list class

3
```

Magic Method Or Dunder method

- special method jo ki **name** kuch aaisa dikhta hai jiske pass ek alag tarah ka super power hota hai jaise ki constructor python me approx 500 magic method hain and iska help karke hum log apna data type banate hain

```
# true work of constuctor is to write configuration related code
because construtor is a function jiska ki command user ke pass nahi
hota hai
#wo sare task jo ki aap user ko puche bina karna chahate ho

# just an intresting example ki suppose god is a programmer and earth
is the class and human is the objects the what is the constructor the
#simply can se jiska ki control human ke pass nn ho ex: death...

# other language me constructor name and class name dono same hota hai
but python me constructor __init__ isi name se hota hai and python me
sirf
#ek hi constructor hota hai
```

self

- Golden rule of oops
- aapke class ke andar jo kuch bhi hai usko sirf class ka object hi access kar sakta hai iska matlab ki agar class ke andar ka koi function kisi dusre function ko call nahi nahi kar sakta hai but ye to ek badi problem ho jayegi ye need pad sakta hai ki ek function ko dusre function se baat karna pade but ye to oops ka concept allow hi nahi karta isi problem ko dur karta hai self islye jab bhi kisi ek function ko kisi dusre function ya data ko use karna hota hai to self ka use karte hain to ye self kon hai ye bhi ek object hai chalo check karte hain islye oops me jo bhi function banate hain

usko ek default parameter mil jata hai jo ki self hota hai . jaise hi aap ek object banate hain to by default wo object self ko mil jata hai as a function parameter.simply self is nothing but the current object and maje ki baat ki jaruri nahi hai ki self hi likho iske jagah pe salman_khan bhi likh do its ok but its a convention ki sab log yahi use karte hain to hum bhi yahi karenge , ek baat aur ki self humesa current object ko point karta hai matlab ki agar object-1 ka location xyz hai to jab tak hum object-1 ke sath kaam karte rahenge to self xyz ko hi point karega but jaise hi hum object-2 ke sath kaam karna suru karenge to object-2 ka jo addree hoga let us say abc to iss time self abc ko hi point karega

```
class Atm:

    def __init__(self):
        print(id(self))
        print(type(self))
        self.pin=''
        self.balance=0



obj=Atm()
```

```
2163878705344
<class '__main__.Atm'>
```

```
print(id(obj)) #observe ki obj ka address and self ka address dono same hai  iska matlab ki aapka jo object hai usi ko salf bulaya jata hai
```

```
2163878705344
```

```
# creating own data types
#creating fraction datatypes
class Fraction:
    #hum aaisa constructor bana rahe hai jo ki kuch input accept karega ise paramiterized constructor kahate hain
    def __init__(self,x,y):
        self.num=x
        self.den=y

    def __str__(self): #ye kuch return karta hai
        return '{}/{}'.format(self.num,self.den) #ye ek string formet hai first value first bracket me and second second bracket me chala jayega

    #lets write the logic for add
    def __add__(self,other): # __add__ do parameter leta hai but name kuch bhi ho sakta hai but ye convention hai (jab do object ko add kar rahe hainn to pahla bala self kahlayega and second bala other)
        new_num=self.num*other.den + other.num*self.den   #logic for
```

```
numenator after adding to fraction
        new_den=self.den*other.den

        return '{}/{}'.format(new_num,new_den)  #ye magic function bhi
kuch return karta hai

    def __sub__(self,other):  #magic function for subtraction
        new_num=self.num*other.den - other.num*self.den
        new_den=self.den*other.den

        return '{}/{}'.format(new_num,new_den)

    def __mul__(self,other): # magic function for multiplication
        new_num=self.num*other.num
        new_den=self.den*other.den

        return '{}/{}'.format(new_num,new_den)

    def __truediv__(self,other): #magic function for division
        new_num=self.num*other.den
        new_den=self.den*other.num

        return '{}/{}'.format(new_num,new_den)

    #hum kuch non magic method bhi add kar sakte hain
    def convert_to_decimal(self):
        return self.num/self.den




fr1=Fraction() # yaha error aayega qki object create karte time koi
parameter pass nahi kye hain
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[101], line 1
----> 1 fr1=Fraction()

TypeError: Fraction.__init__() missing 2 required positional
arguments: 'x' and 'y'
```

```
fr2=Fraction(3,4)  #abb error nahi aa raha hai   qki hum parameter
pass kar rahe hain

print(type(fr2))
```

```
<class '__main__.Fraction'>
```

```
print(fr2) # jab bhi aap apne object ko print karan chahqate hai to wo kuch iss tarah ka output deta hai basiclly object ke address ko print kar deta
            #hai but hum ek proper way me dekhna chahate hain to iske lye hum magic method ka use karte hain i.e __str__ to jab bhi hum object ko
              #print karana chahate hain to ye direct isi magic function ke andar aata hai and iske andaar ke code ko execute kar deta hai
```

```
<__main__.Fraction object at 0x000001F7D0160950>
```

```
print(fr2) #now __str__ define karne ke baad required output mil gya
```

```
3/4
```

```
fr3=Fraction(1,2)
```

```
print(fr3)
```

```
1/2
```

```
print(fr2+fr3) #abhi ye add nahi hoga qki do object ko likhne ka koi logic nahi likhe hain , to chalo add karne ka logic likhte hain and abb fir ek
                  #new magic function ka entry hoga i.e __add__
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[71], line 1
----> 1 print(fr2+fr3)

TypeError: unsupported operand type(s) for +: 'Fraction' and 'Fraction'
```

```
s1={1,2,3}
s2={3,4,5}
s1+s2   # jisne bhi set banane ka logic likha hoga usne uske andar do set ko add karne ka logiic nahi likha hoga
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[73], line 3
      1 s1={1,2,3}
      2 s2={3,4,5}
----> 3 s1+s2

TypeError: unsupported operand type(s) for +: 'set' and 'set'
```

```
fr1=Fraction(3,4)
fr2=Fraction(1,2)

print(fr1,fr2)
```

```
3/4 1/2
```

```
print(fr1+fr2)
```

```
10/8
```

```
print(fr1-fr2) # isko run karenge to kya hoga obviously error aayega qki minus karne ka code nahi likhe hain
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[83], line 1
----> 1 print(fr1-fr2)

TypeError: unsupported operand type(s) for -: 'Fraction' and 'Fraction'
```

```
# lets write the code for minus and phir ek new magic function enter karega i.e __sub__
print(fr1-fr2)
```

```
2/8
```

```
# lets write the code for multiplication and phir ek new magic function enter karega i.e __mul__
print(fr1*fr2)
```

```
3/8
```

```
# lets write the code for division and phir ek new magic function enter karega i.e __truediv__
print(fr1/fr2)
```

```
6/4
```

```
fr1.convert_to_decimal()
```

```
0.75
```

```
# ye  jo libraries hain like numpy,pandas etc ye sab libraries inhi sab concept ko mila ke bana hai
```

OOP Part-2

Write OOP classes to handle the following scenarios:

- A user can create and view 2D coordinates
- A user can find out the distance between 2 coordinates
- A user can find find the distance of a coordinate from origin
- A user can check if a point lies on a given line
- A user can find the distance between a given 2D point and a given line

```
class Point:
    def __init__(self,x,y):
        self.x_cod=x
        self.y_cod=y

    def __str__(self):
        return "<{},{}>".format(self.x_cod,self.y_cod)

    def euclidean_distance(self,other): #self is(x1,y1) And
other(x2,y2)
        return ((self.x_cod-other.x_cod)**2 + (self.y_cod-
other.y_cod)**2)**.5

    def distance_from_origin(self):
        # return self.euclidean_distance(Point(0,0))  #aage self
likhne ka matlab ye bhi hota hai ki hum as argument salf ko bhej bhi
rahe hain
        return (self.x_cod**2 + self.y_cod**2)**0.5

class Line:
    def __init__(self,A,B,C):
        self.A=A
        self.B=B
        self.C=C

    def __str__(self):
        return "{}x + {}y + {} = 0".format(self.A,self.B,self.C)

    def point_on_line(line,point):   #hum self ke jagah sirf line
likhe hain iss line ka class ke name bale line se koi matlab nahi hai
        if line.A*point.x_cod + line.B*point.y_cod + line.C == 0:
#yahi formula to hota hai check karne ka ki koi point line pe hai ki
nahi point ke co-ordinate ko line ke
            return "lies on the line"
#equation me put kar do agar zero ke equal ho raha hai to lies kar
raha hai otherwise nahi

        else:
            return "does not lies on the line"
```

```
    def shortest_distance(line,point):
        return abs(line.A*point.x_cod + line.B*point.y_cod +
line.C)/(line.A**2 + line.B**2)**0.5 #d = [|Ax1 + By1 + C|]/ √(A2 +
B2)
```

```
p1=Point(0,0)
p2=Point(3,4)
```

```
print(p1) #now use __str__  magic function so that user can print the
point
```

```
<0,0>
```

```
p1.euclidean_distance(p2)  # p1 ek object hai to ye data and function
jo ki class me hai usse call kar ssakta hai so call the method
                           #euclidean_distance so p1 self ko and p2
other ko mil jayega
```

```
5.0
```

```
p2.distance_from_origin()  # aaise jab bhi call karte hain ti p2 self
automatically ban jata hai
```

```
5.0
```

```
l1=Line(2,3,4)
```

```
print(l1)
```

```
2x + 3y + 4 = 0
```

```
l1 = Line(1,1,-2)
p1=Point(1,1)
print(l1)
print(p1)
```

```
1x + 1y + -2 = 0
<1,1>
```

```
l1.point_on_line(p1)
```

```
'lies on the line'
```

```
l1.shortest_distance(p1)
```

```
0.0
```

```
# hw ki isi me add karo ki do line segment aapas me intersect karti
hai ki nahi
```

How objects access attributes(data)

```
class Person:

  def __init__(self,name_input,country_input):
    self.name = name_input
    self.country = country_input

  def greet(self):
    if self.country == 'india':
      print('Namaste',self.name)
    else:
      print('Hello',self.name)

#how to access attributes
p=Person("jay","india")

p.name
```

```
'jay'
```

```
#how to access methods
p.greet()
```

```
Namaste jay
```

```
#what if i try to access non existent attributes
p.gender  # ofcourse error aayega
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call
last)
Cell In[121], line 2
      1 #what if i try to access non existent attributes
----> 2 p.gender

AttributeError: 'Person' object has no attribute 'gender'
```

```
### Attribute creation from outside of the class
p.gender="male"  #jaruri nahi ki sare ke sare attributes class ke
andar hi bane usse hum class ke bahar se bhi bana sakte hain

p.gender
```

```
'male'
```

Reference Variables

- Reference variables hold the objects
- We can create objects without reference variable as well
- An object can have multiple reference variables
- Assigning a new reference variable to an existing object does not create a new object

```
# object without reference
# object without a reference
class Person:

  def __init__(self):
    self.name = 'nitish'
    self.gender = 'male'

Person()  # yaha to sirf class name likhne se bhi object ban gya jabki abhi tak (p1=person()) aaisa kuch likhte the to
         #basically abhi tak hum aaisa kuch bol reahe the ki p is an object of personnclass but its not a correst sentennce
           #technally hota ye hai ki class ko call karte hi ek object bana memory me aur hum p me uska reference store kar rahe
         #hain p is just a variable name jiske pass memory address hai object ka
         # bina referance variable ke agar object banayenge to object to banega but memory me gum ho jayega ,
         #taki uss object ko use kar paaye so reference variable banate hain
p=Person()
q=p # jaise aap ye karoge to p and q both are the variable name pointing to the same object

print(id(p))
print(id(q))  #you can see both addresses are same
```

```
2137430812352
2137430812352
```

finally baat ye hai ki p koi object nahi hai ye ek bass variable hai jiske passs ki address hai object ka so p is a reference variaable

```
print(p.name)
print(q.name)
```

```
nitish
nitish
```

```
q.name="jay"
print(p.name)
print(q.name)
```

```
jay
jay
```

Pass by reference

```
class Person:

  def __init__(self,name,gender):
    self.name = name

    self.gender = gender
# this is outside the classs so it's a normal function
def greet(person):  # hum ek object ko bhi pass kar sakte hain  and
koi function return me koi object bhi de sakta hai
    print("hi  my name is ",person.name," and i am a ", person.gender)
    print(id(person))
    person.name="ankit"
    print(person.name)

p=Person("jay","male")
greet(p)   # ye p object(i.e referance variable ) normal function ke
peroson valriable ko milega and ye uske according print kard ega
print(id(p))
```

```
hi  my name is  jay  and i am a  male
3064749696944
ankit
3064749696944
```

object ki mutability

- by default python me obbject mutalbe hoti(just like list , set and dictionary) hai niche ke example deko address same hai matlab ki same location pe change hui hai but hum ise immutable bana sakte hain

```
class Person:

  def __init__(self,name,gender):
    self.name = name

    self.gender = gender
# this is outside the classs so it's a normal function
def greet(person):  # hum ek object ko bhi pass kar sakte hain  and
koi function return me koi object bhi de sakta hai
    person.name="ankit"
    return person

p=Person("jay","male")
```

```
print(id(p))
p1=greet(p)   # ye p object(i.e referance variable ) normal function ke peroson valriable ko milega and ye uske according print kard ega
print(id(p1))
```

```
3064732572880
3064732572880
```

Encapsulation

```
# instance var->python tutor # aaisa variable jiska value alag alag object ke lye alag alag hota hai but normally
class Person:                    #socho ek variable ka ek hi value hota hai

  def __init__(self,name_input,country_input):
    self.name = name_input
    self.country = country_input

p1 = Person('nitish','india')
p2 = Person('steve','australia')

p2.name
```

```
'steve'
```

```
class Atm:



    # constructor(special function)->superpower ->
    def __init__(self):
        print(id(self))
        self.pin = ''
        self.__balance = 0
        #self.menu()

    # creating a getter function
    def get_balance(self):
        return self.__balance

    #creating a setter

    def sel_balance(self,new_value):
        self.__balance=new_value
```

```
    def get_balance(self):
        return self.__balance

    def set_balance(self,new_value):
        if type(new_value) == int:
          self.__balance = new_value
        else:
          print('beta bahot maarenge')

    def menu(self):
        user_input = input("""
        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. Anything else to exit
        """)

        if user_input == '1':
          self.create_pin()
        elif user_input == '2':
          self.change_pin()
        elif user_input == '3':
          self.check_balance()
        elif user_input == '4':
          self.withdraw()
        else:
          exit()

    def create_pin(self):
        user_pin = input('enter your pin')
        self.pin = user_pin

        user_balance = int(input('enter balance'))
        self.__balance = user_balance

        print('pin created successfully')

    def change_pin(self):
        old_pin = input('enter old pin')

        if old_pin == self.pin:


            # let him change the pin
            new_pin = input('enter new pin')
            self.pin = new_pin
            print('pin change successful')
```

```
        else:

            print('nai karne de sakta re baba')

    def check_balance(self):
        user_pin = input('enter your pin')
        if user_pin == self.pin:
          print('your balance is ',self.__balance)
        else:
          print('chal nikal yahan se')

    def withdraw(self):
        user_pin = input('enter the pin')
        if user_pin == self.pin:
          # allow to withdraw
            amount = int(input('enter the amount'))
            if amount <= self.__balance:
                self.__balance = self.__balance - amount
                print('withdrawl successful.balance
is',self.__balance)
            else:
                print('abe garib')
        else:
          print('sale chor')

obj = Atm()
```

```
2842412990432
```

```
# ek kahani :) suppose aap ek senior programmer ho and generally
senior programmer ka kaam hota hai ki class create karna and junior
program ka
#kaam hota hai ki uss class ka use karke kuch kaam karna abb suppose
junior programmer kisi variable ko accsess karke kuch change kar dya
like upar Atm
#bale example me
obj.create_pin()

obj.balance="hehehe"
```

```
enter your pin 1234
enter balance 100000
```

```
pin created successfully
```

```
# abb yaha upar me kya hua junior programmer ne andar ke variable ko
access karke usne balance ke agah pe kuch string hehehe likh dya now
#abb balane me basically ek string store ho gya hai agar abb hum
```

```
woithdraw function ko call kare
obj.withdraw()

enter the pin 1234
enter the amount 5000

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 3
      1 # abb yaha upar me kya hua junior programmer ne andar ke variable ko access karke usne balance ke agah pe kuch string hehehe likh dya now
      2 #abb balane me basically ek string store ho gya hai agar abb hum woithdraw function ko call kare
----> 3 obj.withdraw()

Cell In[25], line 72, in Atm.withdraw(self)
     69 if user_pin == self.pin:
     70   # allow to withdraw
     71   amount = int(input('enter the amount'))
---> 72   if amount <= self.balance:
     73     self.balance = self.balance - amount
     74     print('withdrawl successful.balance is',self.balance)

TypeError: '<=' not supported between instances of 'int' and 'str'

# ye code to fat gya qki balance string me junior ne store kar dya hai , to problem ye hai koi koi another banda aapke class ke variable ko bahar me
#access kar paa raha haiye ek bahut hi dangrous chiz hai ki hum bahar se andar kuch change kar de rahe hain jisse ki problem create ho sakta hai to
#iss problem ko dur karne ke lye concept aaya private variable ka so hum sare ke sare variable ko private bana denge taki bahar ka log ise access nn
#kar sake to hum kisi bhi variable ko private python me double underscore(__) ke help se banate hain other language me private keyword hi hota hai

#phir se khelte hain
obj1=Atm()

2842436773904

#abb suppose kisi tarah junior programmer ko pata chal gya ki balance ko __balance me badal dya hai and junior ko ye pata nahi hai ki ye private
#karne ka tarika hai , aur use laga ki variable ka name change ho gya hai and usne phir se ye karnama suru kya, pahle to ye batao ki ye jo
```

```
niche code
#likha hai wo fatega ki nahi let us see---
obj1.create_pin()

obj1.__balance="hehehe"

enter your pin 1234
enter balance 10000

pin created successfully

obj1.withdraw()

enter the pin 1234
enter the amount 2000

withdrawl successful.balance is 8000

# to yaha code fata nahi balki perfect output aaya qki jab junior
programmer me ye kaand kya to basically hua ye ki __balance name se ek
alag
#variable ban gya memory me jo ki possible hai hum bahar se bhi ek
variable bana sakte hain islye code run to kar gya , but humare
initial variable
#kuch change nahi aaya qki jaise hi hum ek variable ko private banate
hain to uska name change hoke memory me store hota hai to yaha kya hua
#basically __balance convert ho gya _Atm__balance me . to abb jo bhi
new change hoga wo sab kuch alag variable me hoga original bale me
nahi

class Person:

  def __init__(self):
    self.name = "jay"


p1 = Person()
p1.__name="ram"

#ek baar upar ke code ko python tutor me iss code ko run karke dekho
and visualize karo

# abb phir se kahani suru karo suppose junior programmer sala had
harami aadmi hai usko kisi tarah ye pata chal gya internally wo
#variable kiss tarah se store hua hai and phir se karnama suru kya
obj2=Atm()

2842417695920

obj2.create_pin()

obj2._Atm__balance="hehehe"
```

```
enter your pin 1234
enter balance 10000

pin created successfully

obj2.withdraw()

enter the pin 1234
enter the amount 2000

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[75], line 1
----> 1 obj2.withdraw()

Cell In[57], line 72, in Atm.withdraw(self)
     69 if user_pin == self.pin:
     70   # allow to withdraw
     71   amount = int(input('enter the amount'))
---> 72   if amount <= self.__balance:
     73     self.__balance = self.__balance - amount
     74     print('withdrawl successful.balance is',self.__balance)

TypeError: '<=' not supported between instances of 'int' and 'str'

# bc phir code fata to yaha python me ye concept hai ki  "there is
nothing truly private in python". jabki kisi another language me to
hum private \
#variable ko bahar me access hi nahi kar sakte hai to ek question
ubhar ke aata hai ki aakhir q the reason is ki python ek language hai
ye aapko sari
# subidha provide karata hai abb senior junior ki ladai hai to isme
python ki kya galti hai .. ye flexibility provide karata hai ki agar
kabhi kisi
#senario me need pade to hum private ko bhi change kar sakte hai abb
tum chutyapa karoge code ke sath to isme python ki kya galti hai
# and one thing ki python bolta hai ki "python is a language made for
a\gentle man" naki chutiya logo ke lye jo ki code ke sath baith ke
chutiyapa kare

# hum ofcourse kisi method ko bhi private kar sakte hai agar need pade
to same tarike se

## abb phir se kahani suru karte hai sale uss junior programmer ko
bhagaya aur kisi dusre ko laya abb hoga kya suppose iss programmer ko
laga ki private
# variable ko access karna hai and phir usme kuch update karna hai to
wo gya senior ke pass and ye baat bataya to senior ko laga ki baat to
sahi hai
```

```
# but senior private variable ko public nahi karna chahata the but junior ko access bhi dena chahata hai private variable ka

# too yaha pe concept aaya getter and setter function ka :) getter -- data ko get kar payega && setter -- new data se update kar payega

# basically getter setter hai kuch nahi bass do function bana do jisme se ek value paa sake and ek change kar sake ye class ke andar banayenge
# qki class ke method to sare value ko use kar hi sakte hain to hum method ke help se access kar sakte hain uss variable ko and method ko easily hum
#bahar me access kar sakte hain

obj=Atm()

2842436767472

obj.get_balance()

0

obj.set_balance(1000)

obj.get_balance()

1000

# now abb hum value dekh bhi paa rahe hain and update bhi kar paa rahe hain

# abb what iff sala new bala programmer bhi pagla jaye aur ye bhi purane bale ke tarah kaam karne lage

obj5=Atm()

2842417574832


obj5.create_pin()
obj5.set_balance("hehehe")

enter your pin 1234
enter balance 100000

pin created successfully

obj5.withdraw()

enter the pin 1234
enter the amount 5000
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
Cell In[113], line 1
----> 1 obj5.withdraw()

Cell In[103], line 89, in Atm.withdraw(self)
     86 if user_pin == self.pin:
     87   # allow to withdraw
     88     amount = int(input('enter the amount'))
---> 89     if amount <= self.__balance:
     90         self.__balance = self.__balance - amount
     91         print('withdrawl successful.balance
is',self.__balance)

TypeError: '<=' not supported between instances of 'int' and 'str'
```

```
# phir se code fata to abb basically hum ye kar sakte hai ki hum apne
setter function me check laga denge ki jo bhi new value set hoga
#wo sirf int hi ho sakta hai
```

```
obj6=Atm()
```

```
2842415914624
```

```
obj6.create_pin()
```

```
enter your pin 1234
enter balance 1999
```

```
pin created successfully
```

```
obj6.set_balance("hehehe")
```

```
beta bahot maarenge
```

```
## sare baato ko recall karo and then abb encapsulation ko feel karo
```

collection of objects

- basically mean ki hum object ko list , tuples ki tarah se use kar sakte hain

```
# list of object
# list of objects
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

p1 = Person('nitish','male')
```

```
p2 = Person('ankit','male')
p3 = Person('ankita','female')

L = [p1,p2,p3]

print(L) # since isme __str__ magic method nahi hai to tino ka address dikhega
```

```
[<__main__.Person object at 0x00000295CE6F9700>, <__main__.Person object at 0x00000295CE6FA150>, <__main__.Person object at 0x00000295CE6FAC90>]
```

```
for i in L:
    print(i)
```

```
<__main__.Person object at 0x00000295CE6F9700>
<__main__.Person object at 0x00000295CE6FA150>
<__main__.Person object at 0x00000295CE6FAC90>
```

```
for i in L:
    print(i.name,i.gender)
```

```
nitish male
ankit male
ankita female
```

```
# dict of objects

class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

p1 = Person('nitish','male')
p2 = Person('ankit','male')
p3 = Person('ankita','female')

d = {'p1':p1,'p2':p2,'p3':p3}

for i in d:
  print(i)
```

```
p1
p2
p3
```

```
for i in d:
  print(d[i])
```

```
<__main__.Person object at 0x00000295CE6FD850>
<__main__.Person object at 0x00000295CCB845C0>
<__main__.Person object at 0x00000295CE702B40>

for i in d:
  print(d[i].name)

nitish
ankit
ankita

for i in d:
  print(d[i].gender)

male
male
female

for i in d:
  print(d[i].name,d[i].gender)

nitish male
ankit male
ankita female

for i in d:
  print(i, "=" ,d[i].name,d[i].gender)

p1 = nitish male
p2 = ankit male
p3 = ankita female
```

Static Variables(Vs Instance variables)

- static aur class variable dono same hota hai

```
class Atm:

    __counter=1



  # constructor(special function)->superpower ->
  def __init__(self):
    print(id(self))
    self.pin = ''
    self.__balance = 0
      # abb yaha static ko samajhte hain suppose karo ki hume need hai
ki har customer ka ek unique id generate ho
    #self.cid = 0
    #self.cid+=1
```

```
      # ye abb static counter bana rahe hain

    self.cid=Atm.__counter
    Atm.__counter = Atm.__counter + 1

  #abb kya kare agar phie se kaand suru ho jaye to islye stati
variable ko private bana do and  getter and setter bana do
  #ye ek bass utility method hai jisko use karne ke lye koi object
banane ki jaruri nahi hata hai
  @staticmethod # ye ek decorator hai jiska ki use karte hain static
variable ko find karne ke lye
  def get_counter(): # to kya hum yaha pe kisi self ko use kar rahe
hain nahi to andar se selg ko hata do and jab bhi iss gunction ko call
karoge
                       #to class ke name ke sath karna hai like "
Atm.get_counter " agar object ke name ke sath like c1.get_counter call
karoge
                       #to error aayega ki aap ek value as an
arguement pass kar rahe ofcourse wo c1 hai lekin ye function to koi
argument receive
                        #hi nahi karta hai islye ise class ke name ke
sath use karte hain
      return Atm.__counter




  def get_balance(self):
    return self.__balance

  def set_balance(self,new_value):
    if type(new_value) == int:
      self.__balance = new_value
    else:
      print('beta bahot maarenge')

  def __menu(self):
    user_input = input("""
    Hi how can I help you?
    1. Press 1 to create pin
    2. Press 2 to change pin
    3. Press 3 to check balance
    4. Press 4 to withdraw
    5. Anything else to exit
    """)

    if user_input == '1':
      self.create_pin()
```

```
    elif user_input == '2':
      self.change_pin()
    elif user_input == '3':
      self.check_balance()
    elif user_input == '4':
      self.withdraw()
    else:
      exit()

  def create_pin(self):
    user_pin = input('enter your pin')
    self.pin = user_pin

    user_balance = int(input('enter balance'))
    self.__balance = user_balance

    print('pin created successfully')

  def change_pin(self):
    old_pin = input('enter old pin')

    if old_pin == self.pin:
      # let him change the pin
      new_pin = input('enter new pin')
      self.pin = new_pin
      print('pin change successful')
    else:
      print('nai karne de sakta re baba')

  def check_balance(self):
    user_pin = input('enter your pin')
    if user_pin == self.pin:
      print('your balance is ',self.__balance)
    else:
      print('chal nikal yahan se')

  def withdraw(self):
    user_pin = input('enter the pin')
    if user_pin == self.pin:
      # allow to withdraw
      amount = int(input('enter the amount'))
      if amount <= self.__balance:
        self.__balance = self.__balance - amount
        print('withdrawl successful.balance is',self.__balance)
      else:
        print('abe garib')
    else:
      print('sale chor')
```

```
  File <string>:8
    def __init__(self):
                       ^
IndentationError: unindent does not match any outer indentation level


c1=Atm()

2842436837472

c2=Atm()

2842436858464

c3=Atm()

2842417575744

#suppose ki humne tin object banaya abb tino ka customer id print
karate hai abhi tak kaide se tino ka alag alag id hona chahasiye

c1.cid

1

c2.cid

1

c3.cid

1

# ye kya tino ka id to same hi aaya qki har customer ke lye
constructor suru se run ho jayega to jo value 1 hua hai wo phie se
zero and increment hoke 1
# har baar yahi hoga
### matlab ki using a instance variable you cant create a counter qki
har object ke lye instance variable alag hota hai but hume chahaiye ki
har baar jab
# object bane to ye ek hi kisi variable ko point karte rahe jaha ki
har baar change to ho but scratch se nahi to yaha pe concept aaya
static variable ka
# to satic variable kya hota hai static variable class ka variable
hota hai jaise ki instance variable  object ka variable hota hai
# instance variable ka value har object ke lya alag alag hota hai
jabki static variable ka value har object ke lye same hota hai

#kuch question batao :) suppose tum ek bank ka software bana rahe ho
i) toh customer ka name kon sa variable hona chahaiye => obcouse
instance qki
# sab customer(object) ka name different hona chahaiye.   ii) customer
ka balance kon sa variable hona chahaiye => obcouse instance qki
```

```
# sab customer(object) ka balance bhi different hona chahaiye.
iii)bank ka IFSC kon sa variable hona chahaiye => obcouse static qki
# sab customer(object) ka  IFSC  same hona chahaiye.

# static variable humesa class ke andar hota hai jabki instance
variable constructor ke andar

# jab bhi hum static variable ka use karte hain to class ke name ke
sath karte hain and jab bhi instance variable ka use karte hain to
object
#ke name ke sath karte hain

c4=Atm()

2842436856496

c5=Atm()

2842436831328

c6=Atm()

2842409003840

c4.cid

1

c5.cid

1

c6.cid

1
```

Class Relationships

- Aggregation
- Inheritance

Aggregation(Has-A relationship)

- means one class is the owner of the another class

```
# example
class Customer:
    def __init__(self,name,gender,address):
        self.name=name
        self.gender=gender
        self.address=address # address me bhi bahut sa part hota hai
islye alag se ek address class banate hain
```

```
    def print_address(self):
        print(self.address.city,self.address.pin,self.address.state)

class Address:

    def __init__(self,city,pin,state):
        self.city=city
        self.pin=pin
        self.state=state

add1=Address("gurgaon",122011,"haryana")
cust =Customer("nitish","male",add1) # address ek object hona chahaiye
tabhi pahle address  class ka object banaye hain

cust.print_address()

# you cannot access private varaible directly from another class agar
karna hai to getter function bana lo
# suppose hum city ko private member bana de to anotheer class isse
access nahi kar payega
```

```
gurgaon 122011 haryana
```

```
class Customer:
    def __init__(self,name,gender,address):
        self.name=name
        self.gender=gender
        self.address=address

    def print_address(self):
        print(self.address.__city,self.address.pin,self.address.state)

class Address:

    def __init__(self,city,pin,state):
        self.__city=city
        self.pin=pin
        self.state=state

add1=Address("gurgaon",122011,"haryana")
cust =Customer("nitish","male",add1)

cust.print_address()
```

```
---------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[2], line 20
     17 add1=Address("gurgaon",122011,"haryana")
```

```
     18 cust =Customer("nitish","male",add1) # address ek object hona chahaiye tabhi pahle address  class ka object banaye hain
---> 20 cust.print_address()

Cell In[2], line 8, in Customer.print_address(self)
      7 def print_address(self):
----> 8 print(self.address.__city,self.address.pin,self.address.state)

AttributeError: 'Address' object has no attribute '_Customer__city'
```

```
# error islye aa raha hai qki hum city jo ki private member hai usse access karne ka try kar rahe hain agar access karna hi hai to ise
#  getter function ke help se kar lo

class Customer:
    def __init__(self,name,gender,address):
        self.name=name
        self.gender=gender
        self.address=address

    def print_address(self):

print(self.address.get_city(),self.address.pin,self.address.state)

class Address:

    def __init__(self,city,pin,state):
        self.__city=city
        self.pin=pin
        self.state=state

    def get_city(self):
        return self.__city

add1=Address("gurgaon",122011,"haryana")
cust =Customer("nitish","male",add1)

cust.print_address()
```

```
gurgaon 122011 haryana
```

```
class Customer:

  def __init__(self,name,gender,address):
    self.name = name
    self.gender = gender
    self.address = address

  def print_address(self):
```

```
print(self.address._Address__city,self.address.pin,self.address.state)

  def edit_profile(self,new_name,new_city,new_pin,new_state):
    self.name = new_name
    self.address.edit_address(new_city,new_pin,new_state)

class Address:

  def __init__(self,city,pin,state):
      self.__city = city
      self.pin = pin
      self.state = state

  def get_city(self):
    return self.__city

  def edit_address(self,new_city,new_pin,new_state):
    self.__city = new_city
    self.pin = new_pin
    self.state = new_state

add1 = Address('gurgaon',122011,'haryana')
cust = Customer('nitish','male',add1)

cust.print_address()

cust.edit_profile('ankit','mumbai',111111,'maharastra')
cust.print_address()
```

```
gurgaon 122011 haryana
mumbai 111111 maharastra
```

Aggrigation class diagram

- notebook me dekho

Inheritance

- What is inheritance
- Example
- What gets inherited

suppose karo ki koi ek teaching ka website hai for example udemy waha kya hota hai there are two entity first one is student jo ki couse mee enroll karta hai and second one is teacher jo ki course ko create kara hai hai suppose hume udemy jaise ek website banana hai to humara approach kya hoga hum do class banayenge one is for student jisme ki -> login, register, enrol and review jaise functionality ko add karenge and second class banayenge for the rteacher jisme ki -> login , register, create , and reply jaise functionality ko add karenge to yaha pe ek baat observ karo ki hum coding ke DRY(do not repeate yourself) principle ko violet kar rahe hain qki login and register ka code dono class me likh rahe hain jo ki DRY principle ko follow nahi hone de raha hai to what i can do is hum alag se ek class bana sakte hain like name user jo ki parent class

hoga and isme hum functionality add karenge login and register ka and other two chiled class banayenge name like student and teacher and ye dono class parent class ko inherit karega

```
# inherit and it's benifit
#parent class
class User:
    def __init__(self):
        self.name="jay"

    def login(self):
        print("login")

#child class
class Student(User): # ye inherit karne ka tarika hai simply jis class ke content ko inherit karna hai usse current class ke name ke sath bracket me class ke name ke sath likh do abb ye user class ke bhi sare content ko access kar sakta hai
    def __init__(self):
        self.rollno=100

    def enroll(self):
        print("enrolled into the course")



u=User()
s=Student()

print(s.name)  # abhi tak ke logic ke hisaab se output aana to chahaiye but code fata q ?????  qki yaha method override ho raha hai and that is
#constructor  , basically hota kya hai jab hum student class ka object banaye to pahle ye apne original class ke pass gya ang coonstructor ko dhundh and
#usse execute karaya but jab ek baar constructor call ho gya to ye parent ke constructor ke pass gya hi nahi jiske karan name attributes execute hua hi
#nahi , abb agar student class me hu constructor banaye hi nahi to what will happen ki ye pahle apne class me constructor ko dhundhega agar  nahi mila
#to apne parent class ke pass jayega and usse run karayega letus see niche bala exaple

class User:
    def __init__(self):
        self.name="jay"

    def login(self):
        print("login")
```

```
#child class
class Student(User):

    def enroll(self):
        print("enrolled into the course")




u=User()
s=Student()

print(s.name)
```

```
jay
```

```
# abhi code nahi fata qki student class me koi constructor nahi hai
matlab ki aap parent ke constructor ko tabhi use kar sakte hain jabki
# aapke khud ka constructor available nahi ho
```

What gets inherited?

- Constructor
- Non Private Attributes
- Non Private Methods

```
#  constructor Example
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    pass

s=SmartPhone(20000, "Apple", 13)
s.buy()   # non private method hai parent class ka to ofcourse access
kar hi sakte hain
```

```
Inside phone constructor
Buying a phone
```

```
# agar child ke pass apna construct nahi hai to wo parent class ke
constructor ko execute kara dega
```

```python
# constructor example 2

class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

class SmartPhone(Phone):
    def __init__(self, os, ram):
        self.os = os
        self.ram = ram
        print ("Inside SmartPhone constructor")

s=SmartPhone("Android", 2)  # khud ka constructor hai to parent ka constructor call hi nahi hoga
s.brand  # ye error throw karega kyunki parent ka constructor call hi nahi hoga
```

```
Inside SmartPhone constructor
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[7], line 17
     14         print ("Inside SmartPhone constructor")
     16 s=SmartPhone("Android", 2)
---> 17 s.brand

AttributeError: 'SmartPhone' object has no attribute 'brand'
```

```python
# child can't access private members of the class

class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    #getter
    def show(self):
        print (self.__price)

class SmartPhone(Phone):
    def check(self):
        print(self.__price)

s=SmartPhone(20000, "Apple", 13)
```

```
print(s.brand)
s.check()   # check function ke andar parent class ka private member hai to ye access nahi kar payega haa but isko getter function ke help se access kar sakte hain
```

```
Inside phone constructor
Apple

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[15], line 20
     18 s=SmartPhone(20000, "Apple", 13)
     19 print(s.brand)
---> 20 s.check()

Cell In[15], line 16, in SmartPhone.check(self)
     15 def check(self):
---> 16     print(self.__price)

AttributeError: 'SmartPhone' object has no attribute '_SmartPhone__price'
```

```
# child can't access private members of the class

class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    #getter
    def show(self):
        print (self.__price)

class SmartPhone(Phone):
    def check(self):
        print(self.__price)

s=SmartPhone(20000, "Apple", 13)
s.show() # yaha show function non private hai
```

```
Inside phone constructor
20000
```

```
# child can't access private members of the class

class Phone:
    def __init__(self, price, brand, camera):
```

```
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    #getter
    def __show(self):
        print (self.__price)

class SmartPhone(Phone):
    def check(self):
        print(self.__price)

s=SmartPhone(20000, "Apple", 13)
s.show()  # abb show private hai to usse access nahi kar sakte hain
```

```
Inside phone constructor

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[17], line 19
     16         print(self.__price)
     18 s=SmartPhone(20000, "Apple", 13)
---> 19 s.show()

AttributeError: 'SmartPhone' object has no attribute 'show'
```

```
# what will be the output
class Parent:

    def __init__(self,num):
        self.__num=num

    def get_num(self):
        return self.__num

class Child(Parent):

    def show(self):
        print("This is in child class")

son=Child(100)
print(son.get_num())
son.show()
```

```
100
This is in child class
```

```python
# what will be the output
class Parent:

    def __init__(self,num):
        self.__num=num

    def get_num(self):
        return self.__num

class Child(Parent):

    def __init__(self,val,num):
        self.__val=val

    def get_val(self):
        return self.__val

son=Child(100,10)
print("Parent: Num:",son.get_num())
print("Child: Val:",son.get_val())      # ofcourse error aayega qki parent ka constructor kabhi call hoga hi nahi to get_num() execute hoga hi nahi
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[21], line 19
     16         return self.__val
     18 son=Child(100,10)
---> 19 print("Parent: Num:",son.get_num())
     20 print("Child: Val:",son.get_val())

Cell In[21], line 8, in Parent.get_num(self)
      7 def get_num(self):
----> 8     return self.__num

AttributeError: 'Child' object has no attribute '_Parent__num'
```

```python
# what will be the output
class A:
    def __init__(self):
        self.var1=100

    def display1(self,var1):
        print("class A :", self.var1)
        print("class A :", var1)
class B(A):

    def display2(self,var1):
```

```
        print("class B :", self.var1)

obj=B()
obj.display1(200)
```

```
class A : 100
class A : 200
```

```
# Method overriding
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    def buy(self):
        print ("Buying a smartphone")

s=SmartPhone(20000, "Apple", 13)

s.buy()  # agar parent aur child dono ke pass same method hai to child ka method call hoga this is calleed as method overriding
```

```
Inside phone constructor
Buying a smartphone
```

Super Keyword

- super keyword ke help se parent ke chizo ko call kae sakte ho

```
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    def buy(self):
        print ("Buying a smartphone")
        # syntax to call parent ka buy method
        super().buy()  # yaha se abb parent class ka bhi buy method call ho jayega
```

```
s=SmartPhone(20000, "Apple", 13)

s.buy()
```

```
Inside phone constructor
Buying a smartphone
Buying a phone
```

```
# using super outside the class
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    def buy(self):
        print ("Buying a smartphone")
        # syntax to call parent ka buy method
        super().buy()

s=SmartPhone(20000, "Apple", 13)

s.super().buy()    # super keyword ko class ke andar use kartee hain
bahar nahi wo bhi generally child class ke andar
```

```
Inside phone constructor

---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[41], line 20
     16         super().buy()
     18 s=SmartPhone(20000, "Apple", 13)
---> 20 s.super().buy()

AttributeError: 'SmartPhone' object has no attribute 'super'
```

```
# can super access parent ka data?

class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
```

```
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    def buy(self):
        print ("Buying a smartphone")
        # syntax to call parent ka buy method
        print(super().brand)   # super ke help se kabhi bhi hum
variable ko access nahi kar sakte hain only can access method

s=SmartPhone(20000, "Apple", 13)

s.buy()
```

```
Inside phone constructor
Buying a smartphone

---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[43], line 21
     17         print(super().brand)
     19 s=SmartPhone(20000, "Apple", 13)
---> 21 s.buy()

Cell In[43], line 17, in SmartPhone.buy(self)
     15 print ("Buying a smartphone")
     16 # syntax to call parent ka buy method
---> 17 print(super().brand)

AttributeError: 'super' object has no attribute 'brand'
```

in summery

- super cannot access variable
- super cannot be used outside the class
- super is used inside the class

```
# super -> constuctor
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

class SmartPhone(Phone):
```

```
    def __init__(self, price, brand, camera, os, ram):
        print('Inside smartphone constructor')
        super().__init__(price, brand, camera)  # yaha super keyword
ke help se parent ke constructor ko call kar dye and parent ka sara
variable initialize ho gya
        self.os = os
        self.ram = ram
        print ("Inside smartphone constructor")

s=SmartPhone(20000, "Samsung", 12, "Android", 2)

print(s.os)
print(s.brand)
```

```
Inside smartphone constructor
Inside phone constructor
Inside smartphone constructor
Android
Samsung
```

Inheritance in summary

- A class can inherit from another class.
- Inheritance improves code reusebility
- Constructor, attributes, methods get inherited to the child class
- The parent has no access to the child class
- Private properties of parent are not accessible directly in child class
- Child class can override the attributes or methods. This is called method overriding
- super() is an inbuilt function which is used to invoke the parent class methods and constructor

```
# pridict the output
class Parent:

    def __init__(self,num):
      self.__num=num

    def get_num(self):
      return self.__num

class Child(Parent):

    def __init__(self,num,val):
      super().__init__(num)
      self.__val=val
```

```
    def get_val(self):
      return self.__val

son=Child(100,200)
print(son.get_num())
print(son.get_val())
```

```
100
200
```

```
#pridict the output
class Parent:
    def __init__(self):
        self.num=100

class Child(Parent):

    def __init__(self):
        super().__init__()    #self hi son hai to ek parameter as son
pass ho jayeha constructor me
        self.var=200

    def show(self):
        print(self.num)
        print(self.var)

son=Child()
son.show()
```

```
100
200
```

```
#pridict the output
class Parent:
    def __init__(self):
        self.__num=100

    def show(self):
        print("Parent:",self.__num)

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.__var=10

    def show(self):
        print("Child:",self.__var)

obj=Child()
obj.show()
```

```
Child: 10
```

```
#pridict the output
class Parent:
    def __init__(self):
        self.__num=100

    def show(self):
        print("Parent:",self.__num)

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.__var=10

    def show(self):
        print("Child:",self.__var)

obj=Child()
obj.show()
```

```
Child: 10
```

Types of Inheritance

- Single Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Multiple Inheritance(Diamond Problem)
- Hybrid Inheritance

```
# single inheritance
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    pass

SmartPhone(1000,"Apple","13px").buy()
```

```
Inside phone constructor
Buying a phone
```

```python
# multilevel
class Product:
    def review(self):
        print ("Product customer review")

class Phone(Product):
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    pass

s=SmartPhone(20000, "Apple", 12)

s.buy()
s.review()
```

```
Inside phone constructor
Buying a phone
Product customer review
```

```python
# Hierarchical
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class SmartPhone(Phone):
    pass

class FeaturePhone(Phone):
    pass

SmartPhone(1000,"Apple","13px").buy()
FeaturePhone(10,"Lava","1px").buy()
```

```
Inside phone constructor
Buying a phone
Inside phone constructor
Buying a phone
```

```python
# Multiple
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class Product:
    def review(self):
        print ("Customer review")

class SmartPhone(Phone, Product):
    pass

s=SmartPhone(20000, "Apple", 12)

s.buy()  # phone  class ka mothod
s.review() # product class ka method
```

```
Inside phone constructor
Buying a phone
Customer review
```

```python
# the diamond problem
# https://stackoverflow.com/questions/56361048/what-is-the-diamond-problem-in-python-and-why-its-not-appear-in-python2
class Phone:
    def __init__(self, price, brand, camera):
        print ("Inside phone constructor")
        self.__price = price
        self.brand = brand
        self.camera = camera

    def buy(self):
        print ("Buying a phone")

class Product:
    def buy(self):
        print ("Product buy method")

# Method resolution order (MRO)
class SmartPhone(Phone,Product):  # jis class ka name pahle likhenge uska  method call ho jayega
    pass

s=SmartPhone(20000, "Apple", 12)
```

```
s.buy()   # Phone class ka buy method call ho jayega qki inheerrit karte time Phone class pahle likhe hain
```

```
Inside phone constructor
Buying a phone
```

```
#pridict the output
class A:

    def m1(self):
        return 20

class B(A):

    def m1(self):
        return 30

    def m2(self):
        return 40

class C(B):

    def m2(self):
        return 20
obj1=A()
obj2=B()
obj3=C()
print(obj1.m1() + obj3.m1()+ obj3.m2())
```

```
70
```

```
#pridict the output
class A:

    def m1(self):
        return 20

class B(A):

    def m1(self):
        val=super().m1()+30
        return val

class C(B):

    def m1(self):
        val=self.m1()+20  # yaha recursion ka problem aa jayega qki fuction overriding ke karan yahi function baar bssr execute karega
        return val
obj=C()
print(obj.m1())
```

Polymorphism

- Method Overriding
- Method Overloading
- Operator Overloading

```
# method overriding already  upar me discuss kar chuke hai ki same
name ka do function hai to kon execute hoga

# Method overloading : in a single class ek hi name ke do method hain
but dono ka behaviour different hai depending on the input

class Shape:

    def area(self,radius):
        return 3.14*radius*redius

    def area(self,l,b):
        return l*b


s = Shape()


print(s.area(3,4))
print(s.area(2))
```

```
12

---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[92], line 16
     12 s = Shape()
     15 print(s.area(3,4))
---> 16 print(s.area(2))

TypeError: Shape.area() missing 1 required positional argument: 'b'
```

```
 # error qki python me function overloading name ka koi concept nahi
hota hai agar same name ka do function create karenge to baad bale ke
 # according execute hoga qki python kahata hai ki hum same kaam with
the help of defaul argument ke help se kar sakte hain let seeee..

class Shape:

  def area(self,a,b=0):
    if b == 0:
      return 3.14*a*a
    else:
      return a*b
```

```
s = Shape()

print(s.area(2))
print(s.area(3,4))
```

```
12.56
12
```

```
#  operator overloading => same operator alag alag input pe alag alag
kaam karta hai

"hello" + "world"  # + -> act as concatination
```

```
'helloworld'
```

```
4+5  # + -> act as addition
```

```
9
```

```
[1,2,3]+[4,5,6]   # + -> act as merging
```

```
[1, 2, 3, 4, 5, 6]
```

Abstraction

Abstract class => wo class jiske andar at least ek abstract method hota hai -> Abstract Method :) waisa methode jiske andar kuch bhi code nahi hota hai basically do tarah ke method hote hain i) abstract method and ii)concrete method-> jiske andar ki kuch kuch code hota hai

```
from abc import ABC,abstractmethod
class BankApp(ABC):

    def database(self):
        print("connected to database")


    @abstractmethod
    def security(self):
        pass

class MobileApp(BankApp):

    def mobile_login(self):
        print("login into mobile")

    def security(self):    # jab tak ye security function  iss class
me  nahi banayenge to error aayega jab object banayenge basically
abstraction
        print("mobile security")

mob = MobileApp()
```

```
# 14 mint ka video hai ek baar dekh lo 2x pe
```

file handling

Some Theory

Types of data used for I/O:

- Text - '12345' as a sequence of unicode chars
- Binary - 12345 as a sequence of bytes of its binary equivalent

Hence there are 2 file types to deal with

- Text files - All program files are text files
- Binary Files - Images,music,video,exe files

How File I/O is done in most programming languages

- Open a file
- Read/Write data
- Close the file

Writing to a file

```
# case 1 - if the file is not present

f = open('sample.txt','w') # ye line likhte hi same directory me ek sample.txt name file ban jayega

f.write('hello world')   # sample.txt me hello world write ho jayega
```

```
11
```

```
f.close()    # file close kar dye abb jab tak phir se open nahi karenge tabtak phir se kuch write ya read nahi kar payenge

f.write('hello')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 f.write('hello')

ValueError: I/O operation on closed file.
```

```
# write multiline string
f = open('sample1.txt','w')
f.write('hello world')
```

```
f.write('\nhow are  you')
f.close()

# case 2 - if the file is already present

f = open('sample.txt','w') #suppose sample.txt me kuch likhna hai phir se
f.write('akshay kumar')
f.close()

# abhi jo purana data tha pahle sample.txt me i.e hello world wo erase ho jayega and usme akshaye kumar aaa jayega

# how exactly open() works  : basically file ROM me hota hai and as we call the open function to python jata hai and uss file ko lake RAM me load kar deta hai and load bhi buffer me hota hai jaha character by character
# data read hota hai and and jab close() function ko call kya jata hai to again ye ROM me store ho jata hai

# problem with w mode   :  ye purane data ko erase kar dete hai  so aviod it we use append mode

# introducing append mode

f = open('sample1.txt','a')
f.write('\n i am fine')
f.close()

# write lines

L = ['hello\n','hi\n','how are you\n',' i an fine']

f = open('sample.txt','w')
f.writelines(L)    # yadi multiple lines jo ki list me available hai usko likhna chahate hain yaha multiple times write ko call nahi karna padega

f.close()

# yadi hum write ke through kareng to
L = ['hello\n','hi\n','how are you\n',' i an fine']

f = open('sample.txt','w')
f.write(L)

f.close()

# reading from files
# using read()  : read -> ek baar me pure file ko read kar leta hai

f=open('sample.txt','r')
```

```
s=f.read()
print(s)
f.close()

# reading upto n chars
f=open('sample.txt','r')
s=f.read(10)     # 10 charecter hi read hoga
print(s)
f.close()

# readline()  -> read line by line

f=open('sample.txt','r')
print(f.readline())
print(f.readline())
f.close()

# readline automatically line change kar deta hai and print bhi line
change karta hai islye do line change ho gya and bich me ek extra
space aa gya hai

f=open('sample.txt','r')
print(f.readline(),end='')
print(f.readline(),end='')
f.close()

# readline tab use karte hain jab humare pass bahut bada file hota hai
and pure file ko ek saath memory me load nahi karna chahate hain

#reading entire using readline

f=open('sample.txt','r')

while True :
    data =f.readline()

    if data == '':
        break

    else:
        print(data,end='')

f.close()
```

Using Context Manager (With)

- It's a good idea to close a file after usage as it will free up the resources
- If we dont close it, garbage collector would close it
- with keyword closes the file as soon as the usage is over

```
# with is basically the replacement of f.closse()
```

```
# with

with open('sample1.txt','w') as f:
    f.write('salmon bhai')         # ye automatically file ko close
kar deta hai

f.write('hello')  # qki with automatically file ko close kar dya hai

# try f.raed() now

with open('sample.txt','r') as f:
    print(f.read())

with open('sample.txt','r') as f:
    print(f.readline())

# moving within a file -> 10 char then 10 char

with open('sample.txt','r') as f:
    print(f.read(10))

with open('sample.txt','r') as f:
    print(f.read(10))
    print(f.read(10))   # abb next 10 character print karega yahi
fayda hota hai buffer ka wo count rakhta hai ki kitne character ko
process kya gya hai

# benifit  :  to load a big file in memory

big_L = ['heello world' for i in range(1000)]

with open('big.txt','w') as f:
    f.writelines(big_L)

# ye ek badi file hai hum isee chunk me la la ke kaam kar sakte hain
and kaam ke baadd remove kar sakte hain phir next chumk ko laa sakte
hainn and
# so onnnn.... isse memory effeciently use hoga

with open('big.txt','r') as f:

    chunk_size=100
    while len(f.read(chunk_size)) >0:
        print(f.read(chunk_size),end='*')
        f.read(chunk_size)   # current chunk ko print kar rahe hai and
next chunk ko load kar rahe hain

# seek an tell function
with open('sample.txt','r') as f:
    print(f.read(10))
    print(f.tell())    # tell batata hai ki  cursor next kon sa
character par hoga
```

```
# seek  batata hai ki current cursor ko hum kahi bhi khich ke le jaa
sakte hain
with open('sample.txt','r') as f:
    print(f.read(10))
    print(f.tell())
    f.seek(12)           # ye 12th character pe le jayega direct
    print(f.read(10))
    print(f.tell())

# seek during write
with open('sample.txt','w') as f:
    f.write('Hello')
    f.seek(0)             # 0th index pe jayega and phir se uss jagah
pe X write kar dega and output hoga 'Xello'
    f.write('X')
```

Problems with working in text mode

- can't work with binary files like images
- not good for other data types like int/float/list/tuples

```
# working with binary file
with open('Thumbnel.png','r') as f:
    f.read()                  #image ek binary file hai to usse iss
tarah se hum read nahi kar sakte hain

with open('Thumbnel.png','rb') as f:    # rb-> read binary
    with open('Thumbnel_copy.png','wb') as wf:  # wb -> write binary
hum basically file ko read karke phir se ek another file me write kar
lye
        wf.write(f.read())

# working with other data types
with open('sample.txt','w') as f:
    f.write(5)             # hum sirf string hi write kar sakte hain
what if hume need ho kisi another data type ko write karne ki -->
chalo dekhte hain

with open('sample.txt','w') as f:
    f.write('5')        # isse string format me send kar do

with open('sample.txt','r') as f:
    print((f.read())+5)

with open('sample.txt','r') as f:
    print(int(f.read())+5)

# more complex data

d = {
    'name':'nitish',
```

```
    'age':33,
    'gender':'male'
}

with open('sample.txt','w') as f:
    f.write(d)

d = {
    'name':'nitish',
    'age':33,
    'gender':'male'
}

with open('sample.txt','w') as f:
    f.write(str(d))

with open('sample.txt','r') as f:
    print(f.read())

with open('sample.txt','r') as f:
    print(dict(f.read()))      #error qki string ko dictionary me
convert nahi kar sakte hain

## basically problem ye aa raha hai ki hum complex data type ko file
me store nahi kara sakte hain
# issi problem ko dur karne ke lye aaya next tope i. Serialization and
Deserialization
```

Serialization and Deserialization

- **Serialization** - process of converting python data types to JSON format
- **Deserialization** - process of converting JSON to python data types

What is JSON?

```
{
    "d": {
        "results": [
            {
                "__metadata": {
                    "type": "EmployeeDetails.Employee"
                },
                "UserID": "E12012",
                "RoleCode": "35"
            }
        ]
    }
}
```

```
# JSON -> javaScript on notation , kind of universal text type (or
data type) jisko ki sab language janta hai, basically key value pair
me data hoata hai

# serialization using json module
#list

import json

l=[1,2,3,4,5]

with open('demo.json','w') as f:
    json.dump(l,f)     #json.dump ek function hai

# dictionary
d = {
    'name':'nitish',
    'age':33,
    'gender':'male'
}
with open('demo.json','w') as f:   # f is nothing but a variable known
as file handler object
    json.dump(d,f,indent=4)    # indent=4 bass ek better format
provide karta hai 4 -> space hai 4 ke jagah kuch bhi le sakte hai
according to need

# deserialization

import json
```

```
with open('demo.json','r') as f:
    print(json.load(f))


with open('demo.json','r') as f:
    d=json.load(f)
    print(d)
    print(type(d))

# serializa and deserialize tuple
import json

t = (1,2,3,4,5)

with open('demo.json','w') as f:
  json.dump(t,f)   # ye ek list ke tarah hi store hota hai

d = {
    'student':'nitish',
     'marks':[23,14,34,45,56]
}

with open('demo.json','w') as f:
  json.dump(d,f)
```

Serializing and Deserializing custom objects

```
class Person:

  def __init__(self,fname,lname,age,gender):
    self.fname = fname
    self.lname = lname
    self.age = age
    self.gender = gender

# format to printed is
# -> Nitish Singh age -> 33 gender -> male

person = Person('Nitish','Singh',33,'male')

# custome object ko file me rakhna hai

import json
with open('demo.json','w') as f:
    json.dump(person,f)           # object ko iss method de dump
(write) nahi kara sakte qki normaly json me dict form me data store
hota
                                  # hai but yaha isse pata hi nahi
chal raha hai ki iss data ko dictio nary me kaise store karenge
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[8], line 3
      1 import json
      2 with open('demo.json','w') as f:
----> 3     json.dump(person,f)           # object ko iss method de
dump (write) nahi kara sakte qki normaly json me dict form me data
store hota hai but yaha isse pata hi nahi chal raha hai ki iss
      4                                     #data ko dictio nary me
kaise store karenge

File ~\anaconda3\Lib\json\__init__.py:179, in dump(obj, fp, skipkeys,
ensure_ascii, check_circular, allow_nan, cls, indent, separators,
default, sort_keys, **kw)
    173     iterable = cls(skipkeys=skipkeys,
ensure_ascii=ensure_ascii,
    174         check_circular=check_circular, allow_nan=allow_nan,
indent=indent,
    175         separators=separators,
    176         default=default, sort_keys=sort_keys,
**kw).iterencode(obj)
    177 # could accelerate with writelines in some versions of Python,
at
    178 # a debuggability cost
--> 179 for chunk in iterable:
    180     fp.write(chunk)

File ~\anaconda3\Lib\json\encoder.py:439, in
_make_iterencode.<locals>._iterencode(o, _current_indent_level)
    437         raise ValueError("Circular reference detected")
    438     markers[markerid] = o
--> 439 o = _default(o)
    440 yield from _iterencode(o, _current_indent_level)
    441 if markers is not None:

File ~\anaconda3\Lib\json\encoder.py:180, in JSONEncoder.default(self,
o)
    161 def default(self, o):
    162     """Implement this method in a subclass such that it
returns
    163     a serializable object for ``o``, or calls the base
implementation
    164     (to raise a ``TypeError``).
   (...)
    178
    179     """
--> 180     raise TypeError(f'Object of type {o.__class__.__name__} '
    181                     f'is not JSON serializable')
```

```
TypeError: Object of type Person is not JSON serializable
```

```
# as a string
import json

def show_object(person):
    if isinstance(person,Person):
        return '{} {} age - > {} gender -> {}
'.format(person.fname,person.lname,person.age,person.gender)

with open('demo.json','w') as f:
    json.dump(person,f,default=show_object)
```

```
import calendar

year = int(input('Enter year  : '))

month = int(input('Enter month  : '))

cal = calendar.month(year,month)

print(cal)
```

```
Enter year  :  2025
Enter month  :  5

      May 2025
Mo Tu We Th Fr Sa Su
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```