Python_Complete_Work

September 25, 2025

1 Complete Python Learning Journey: From Basics to Advanced

1.0.1 *By Awais Manzoor*

1.1 Overview

This notebook contains a comprehensive collection of Python codes and concepts, starting from basic variables and data types to advanced topics like object-oriented programming, file handling, and serialization. It also includes practical mini-projects to demonstrate real-world applications.

1.1.1 Contents

2 Chapter 1 variables and data types

[1]:
```python
print("Hello world")
```

```
Hello world
```

[4]:
```python
a = 8
b = 9
c = a+b
print("Sum is:", c)
```

```
Sum is: 17
```

[6]:
```python
num1 = int(input('Enter a number: '))
num2 = int(input('Enter another number: '))
sum = num1 + num2
print('The sum of', num1, 'and', num2, 'is', sum)
```

```
The sum of 3 and 4 is 7
```

[7]:
```python
num1 = int(input("enter your first number"))
num2 = int(input("enter second number"))
sum = num1 + num2
print("The sum of",num1, 'and', num2, 'is', sum)
```

```
The sum of 5 and 8 is 13
```

[8]:
```python
num1 = int(input("enter first number"))
num2 = int(input("enter other no "))
sum = num1 + num2
print(f"The sum of", num1, 'and', num2, 'is', sum)
```

```
The sum of 77 and 99 is 176
```

[9]:
```python
num1 = int(input('enter a number'))
num2 = int(input('ENTER ANOTHER NUMBER:'))
sum = num1 + num2
print('The sum of', num1, 'and', num2, 'is', sum)
```

```
The sum of 1000 and 5000 is 6000
```

[10]:
```python
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 2.0
print(num2, 'is of type', type(num2))

num3 = 1+2j
print(num3, 'is of type', type(num3))
```

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type <class 'complex'>
```

[11]:
```python
name = ("awais manzooor")
print(name.upper())
print(name.lower())
```

```
AWAIS MANZOOOR
awais manzooor
```

[ ]:
```python
name_1= "corporate"
name_2="law"
full_name = f"{name_1} {name_2}"
print(full_name.capitalize())
```

```
Corporate law
```

[13]:
```
print("Languages:\nPython\nC\njavaScript")
```

```
Languages:
Python
C
javaScript
```

[14]:
```
print("Python")
print("\tPython")
```

```
Python
        Python
```

[15]:
```
name = "awais manzooor"
print(name.title())
```

```
Awais Manzooor
```

[16]:
```
first_name = "Awais"
last_name = "Manzoor"
full_name = f"{first_name} {last_name}"
print(full_name)
```

```
Awais Manzoor
```

[17]:
```
first_name = "Awais"
last_name = "Manzoor"
full_name = f"{first_name} {last_name}"
print(full_name)
```

```
Awais Manzoor
```

[18]:
```
print("languages: \ntpython")
```

```
languages:
tpython
```

[19]:
```
print("languages: \tpython")
```

```
languages:      python
```

[20]:
```
print("languages: \npython")
```

```
languages:
python
```

[21]:
```
print("languages: \n\tpython\n\tcooding\n\tc++")
```

```
languages:
        python
        cooding
        c++
```

[22]:

```
fav_languages = ' python '                  # remove space right side.
fav_languages = ' python '                  # remove space left side.
fav_languages = ' python '                   # remove space both side.
print(fav_languages.rstrip())
print(fav_languages.lstrip())
print(fav_languages.strip())
```

```
 python
python
python
```

#***Float values***

[23]:

```
print(2.2 + 2.3)
print(2.2 - 2.3)
print(2.2 * 2.3)
print(2.2 / 2.3)
print(2.2 ** 2.3)
```

```
4.5
-0.09999999999999964
5.06
0.9565217391304349
6.131576709333357
```

3 Integers value.

[24]:

```
print(2 + 2)
print(2 - 2)
print(2 * 2)
print(2 ** 2)
print(2 / 2)
```

```
4
0
4
4
1.0
```

[25]:

```
celsius = float(input("Enter temperature in celsius: "))
fahrenheit = (celsius * 9/5) + 32
print(celsius,'celsius','=', fahrenheit,'fahrenheit')
```

```
6.0 celsius = 42.8 fahrenheit
```

3.1 Euclidean Distance

[26]:
```
# Euclidean Distance find out
# 1. The distance between two points in a 2D plane
x_1 = float(input("x_1 of x cordinate"))
y_1 = float(input("y_1 of y cordinate"))
x_2 = float(input("x_2 of x cordinate"))
y_2 = float(input("y_2 of y cordinate"))
d = (((x_2 - x_1)**2 + (y_2 - y_1)**2)**0.5)
print("The distance between two points is ", d)
```

```
The distance between two points is  2.8284271247461903
```

[27]:
```
help('modules')
```

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

test_sqlite3: testing with SQLite version 3.43.1

c:\Users\hp\AppData\Local\Programs\Python\Python312\Lib\pkgutil.py:78:
UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
  __import__(info.name)
c:\Users\hp\AppData\Local\Programs\Python\Python312\Lib\pkgutil.py:78:
UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  __import__(info.name)
c:\Users\hp\AppData\Local\Programs\Python\Python312\Lib\pkgutil.py:78:
UserWarning:
The dash_table package is deprecated. Please replace
`import dash_table` with `from dash import dash_table`

Also, if you're using any of the table format helpers (e.g. Group), replace
`from dash_table.Format import Group` with
`from dash.dash_table.Format import Group`
  __import__(info.name)
c:\Users\hp\AppData\Local\Programs\Python\Python312\Lib\site-
packages\fpdf\__init__.py:40: UserWarning:

You have both PyFPDF & fpdf2 installed. Both packages cannot be installed at the
same time as they share the same module namespace. To only keep fpdf2, run: pip
uninstall --yes pypdf && pip install --upgrade fpdf2
```

```
c:\Users\hp\AppData\Local\Programs\Python\Python312\Lib\site-
packages\pydantic\experimental\__init__.py:7: PydanticExperimentalWarning: This
module is experimental, its contents are subject to change and deprecation.

pygame 2.6.1 (SDL 2.28.4, Python 3.12.2)
Hello from the pygame community. https://www.pygame.org/contribute.html

WARNING Task(Task-4)
setuptools.config._validate_pyproject.formats:formats.py:<module>()- Could not
find an up-to-date installation of `packaging`. License expressions might not be
validated. To enforce validation, please install `packaging>=24.2`.
[2025-09-24 11:34:33,234] Could not find an up-to-date installation of
`packaging`. License expressions might not be validated. To enforce validation,
please install `packaging>=24.2`.

IPython             cohere              mdurl               sqlglot
PIL                 collections         mergedeep           sqlite3
__future__          colorama            mimetypes           squarify
__hello__           colorsys            missingno           sre_compile
__phello__          comm                mizani              sre_constants
_abc                commctrl            mkdocs              sre_parse
_aix_support        compileall          mkdocs_get_deps     ssl
_ast                concurrent          mmap                sspi
_asyncio            configparser        mmapfile            sspicon
_bisect             contextlib          mmsystem            stack_data
_blake2             contextvars         modulefinder        starlette
_brotli             contourpy           monotonic           start_pythonwin
_bz2                copy                mpl_toolkits        stat
_codecs             copyreg             msilib              statistics
_codecs_cn          crypt               msvcrt              statsmodels
_codecs_hk          csv                 multimethod         string
_codecs_iso2022     ctypes              multipart           stringprep
_codecs_jp          curses              multiprocessing     strsimpy
_codecs_kr          cv2                 narwhals            struct
_codecs_tw          cycler              nest_asyncio        subprocess
_collections        dacite              netbios             sunau
_collections_abc    dash                netrc               sweetviz
_compat_pickle      dash_bootstrap_components networkx            symtable
_compression        dash_core_components nntplib             sys
_contextvars        dash_daq            nt                  sysconfig
_csv                dash_html_components ntpath              tabnanny
_ctypes             dash_table          ntsecuritycon       tarfile
_ctypes_test        dataclasses         nturl2path          telnetlib
_datetime           datetime            numba               tempfile
_decimal            dateutil            numbers             test
_distutils_hack     dateutils           numpy               textwrap
_elementtree        dbi                 odbc                this
_functools          dbm                 opcode              threading
```

```
_hashlib            dde                 openai              threadpoolctl
_heapq              debugpy             openpyxl            tifffile
_imp                decimal             operator            tiktoken
_io                 decorator           optparse            time
_json               defusedxml          orjson              timeit
_locale             difflib             os                  timer
_lsprof             dis                 packaging           tk
_lzma               diskcache           pandas              tkinter
_markupbase         distro              pandas_profiling    token
_md5                doctest             parso               tokenize
_msi                docutils            past                tokenizers
_multibytecodec     dtale               pathlib             tomllib
_multiprocessing    duckdb              pathspec            tornado
_opcode             email               patsy               tqdm
_operator           emoji               pdb                 trace
_osx_support        encodings           perfmon             traceback
_overlapped         ensurepip           phik                tracemalloc
_pickle             enum                pickle              traitlets
_plotly_utils       errno               pickletools         ttkbootstrap
_py_abc             et_xmlfile          pip                 ttkcreator
_pydatetime         executing           pipes               tty
_pydecimal          fastapi             pkg_resources       turtle
_pyio               fastavro            pkginfo             turtledemo
_pylong             faulthandler        pkgutil             typeguard
_pyrsistent_version filecmp             platform            typer
_queue              fileinput           platformdirs        types
_quickjs            filelock            plistlib            typing
_random             flask               plotly              typing_extensions
_sha1               flask_compress      plotnine            typing_inspection
_sha2               fnmatch             polars              tzdata
_sha3               fontTools           pooch               unicodedata
_signal             fpdf                poplib              unittest
_sitebuiltins       fractions           posixpath           urllib
_socket             fsspec              pprint              urllib3
_sqlite3            ftplib              profile             uu
_sre                functools           prompt_toolkit      uuid
_ssl                future              pstats              uvicorn
_stat               gc                  psutil              venv
_statistics         genericpath         psygnal             visions
_string             geopandas           pty                 warnings
_strptime           getopt              pure_eval           wasmtime
_struct             getpass             puremagic           watchdog
_symtable           gettext             pvectorc            wave
_testbuffer         ghp_import          py_compile          wcwidth
_testcapi           glob                pyarrow             weakref
_testclinic         gpsd                pyasn1              webbrowser
_testconsole        graphlib            pyasn1_modules      werkzeug
_testimportmultiple greenlet            pyclbr              widgetsnbextension
```

```
_testinternalcapi   gw_dsl_parser       pydantic            win2kras
_testmultiphase     gzip                pydantic_core       win32api
_testsinglephase    h11                 pydoc               win32clipboard
_thread             hashlib             pydoc_data          win32com
_threading_local    heapq               pyexpat             win32con
_tkinter            hmac                pygame              win32console
_tokenize           html                pygments            win32cred
_tracemalloc        htmlmin             pygwalker           win32crypt
_typing             http                pygwalker_tools     win32cryptcon
_uuid               httpcore            pylab               win32event
_warnings           httpx               pymatting           win32evtlog
_weakref            httpx_sse           pyogrio             win32evtlogutil
_weakrefset         huggingface_hub     pyparsing           win32file
_win32sysloader     idlelib             pyperclip           win32gui
_winapi             idna                pyproj              win32gui_struct
_winxptheme         imagehash           pyrsistent          win32help
_wmi                imageio             python_multipart    win32inet
_xxinterpchannels   imaplib             pythoncom           win32inetcon
_xxsubinterpreters  imghdr              pytz                win32job
_yaml               importlib           pywin               win32lz
_zoneinfo           importlib_metadata  pywin32_bootstrap   win32net
abc                 importlib_resources pywin32_testutil    win32netcon
ac51d50a4f4b6d748b8c__mypyc inspect             pywintypes          win32pdh
adbc_driver_duckdb  io                  pywt                win32pdhquery
adodbapi            ipaddress           pyzstd              win32pdhutil
afxres              ipykernel           qrcode              win32pipe
aifc                ipykernel_launcher  queue               win32print
altair              ipylab              quickjs             win32process
analytics           ipywidgets          quopri              win32profile
annotated_types     isapi               random              win32ras
antigravity         itertools           rasutil             win32rcparser
anyio               itsdangerous        re                  win32security
anywidget           jedi                regcheck            win32service
appdirs             jinja2              regex               win32serviceutil
argparse            jiter               regutil             win32timezone
array               joblib              rembg               win32trace
arrow               json                reprlib             win32traceutil
ast                 jsonschema          requests            win32transaction
astor               jupyter             retrying            win32ts
asttokens           jupyter_client      rich                win32ui
asyncio             jupyter_core        rlcompleter         win32uiole
atexit              jupyterlab_widgets  rsa                 win32verstamp
attr                kagglehub           runpy               win32wnet
attrs               kaleido             sched               winerror
audioop             kanaries_track      scipy               winioctlcon
backoff             keyword             seaborn             winnt
base64              kiwisolver          secrets             winperf
bdb                 lazy_loader         select              winreg
```

```
binascii            lib2to3             selectors           winsound
bisect              libfuturize         servicemanager      winxpgui
blinker             libpasteurize       setuptools          winxptheme
brotli              lida                shapely             wordcloud
bs4                 linecache           shellingham         wsgiref
builtins            llmx                shelve              xarray
bz2                 llvmlite            shlex               xdrlib
cProfile            locale              shutil              xlrd
cachetools          logging             signal              xml
calendar            logistro            simplejson          xmlrpc
certifi             lz4                 site                xxsubtype
cgi                 lzma                six                 yaml
cgitb               mailbox             skimage             yaml_env_tag
charset_normalizer  mailcap             skimpy              ydata_profiling
choreographer       markdown            sklearn             zipapp
chunk               markdown_it         smtplib             zipfile
click               markupsafe          sndhdr              zipimport
cmath               marshal             sniffio             zipp
cmd                 math                socket              zlib
code                matplotlib          socketserver        zmq
codecs              matplotlib_inline   soupsieve           zoneinfo
codeop              matplotlib_venn     sqlalchemy

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

4 LISTS

[28]:
```
#1. Lists  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.
"""Characterstics of a List
Ordered
Changeble/Mutable
Hetrogeneous
Can have duplicates
are dynamic
can be nested
items can be accessed
can contain any kind of objects in python"""
```

[28]:
```
'Characterstics of a List\nOrdered\nChangeble/Mutable\nHetrogeneous\nCan have
duplicates\nare dynamic\ncan be nested\nitems can be accessed\ncan contain any
kind of objects in python'
```

[29]:
```
# append
L = [1,2,3,4,5]
L.append(True)
print(L)
```

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

[30]:
```
# extend
L = [1,2,3,4,5]
L.extend([6,7,8])
print(L)
```

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

[31]:
```
L = [1,2,3,4,5]

# editing with indexing
L[-1] = 500

# editing with slicing
L[1:4] = [200,300,400]

print(L)
```

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

[32]:
```
# del
L = [1,2,3,4,5]

# indexing
del L[-1]

# slicing
del L[1:3]
print(L)
```

```
[1, 4]
```

[33]:
```
# remove

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

L.remove(5)

print(L)
```

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

[34]:
```
# pop
L = [1,2,3,4,5]

L.pop()

print(L)
```

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

5 Zip

[35]:
```
"""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."""
```

[35]:
```
'Zip\nThe 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.\n\nIf the passed
iterators have different lengths, the iterator with the least items decides the
length of the new iterator.'
```

[36]:
```
countries = ['Pakistan', 'Japna', 'India',' America']
print(countries)
```

```
['Pakistan', 'Japna', 'India', ' America']
```

[37]:
```
# A list is a collection of items in a particular order
# a list usually contains more then one element e.g letters, digits, or names
# in python suqure brakets ([]) indicate a list and individual eklements in the␣
↪list
# are separated by commas
cities = ['Lahore', 'Karachi', 'Islamabad', 'Swat']
print(cities[0])
```

```
Lahore
```

[38]:
```
bicycles = ['trek', 'redline']
message = f"My favourite bike is: {bicycles[1].upper()}"
print(message)
print('trek')
```

```
My favourite bike is: REDLINE
trek
```

[ ]:
```
foods = ['pizza', 'sandwich', 'burger']
message = f"My fav_food is {foods[2].upper()}"
print(message)
```

```
My fav_food is BURGER
```

6 ASSIGNMENT 1

[41]:
```
Guest_list = ['Ali', 'Zohaib', 'Awais']
print(Guest_list)
```

```
['Ali', 'Zohaib', 'Awais']
```

[42]:
```
# Original guest list
guest_list = ["Ali", "Babar", "Amir"]

# Print the original invitations
for guest in guest_list:
 print(f"Dear {guest}, you are invited to dinner!")
```

```
Dear Ali, you are invited to dinner!
Dear Babar, you are invited to dinner!
Dear Amir, you are invited to dinner!
```

[43]:
```
guest_list = ["Shaheen", "Babar", "Amir"]
for guest in guest_list:
 cant_make_it = "Babar"
new_guest = "Fakhar Azam"
guest_list[guest_list.index(cant_make_it)] = new_guest

for guest in guest_list:
 print(f"Dear {guest}, you are invited to dinner!")
```

```
Dear Shaheen, you are invited to dinner!
Dear Fakhar Azam, you are invited to dinner!
Dear Amir, you are invited to dinner!
```

[44]:
```
# fav_ fooods any 5.
fav_foods = ['pizza','burger','sandwich','karhai','pulio']
print(fav_foods)
```

```
['pizza', 'burger', 'sandwich', 'karhai', 'pulio']
```

[ ]:
```
# 2nd item of lists
fav_foods = ['pizza','burger','sandwich','karhai','pulio']
print(fav_foods[1])
```

```
burger
```

[46]:
```
favorite_foods = ["pizza", "sushi", "tacos", "steak", "chicken"]
uppercase_foods = ('favorite _foods'.upper())
print(uppercase_foods)
```

```
FAVORITE _FOODS
```

[47]:
```
food = ("burger")
print(food.upper())
```

```
BURGER
```

[48]:
```
food = ["burger", "beryani", "raita"]
food.append(2)
print(food)
```

```
['burger', 'beryani', 'raita', 2]
```

[49]:
```
food = ["burger", "beryani", "raita"]
food.pop(2)
print(food)
```

```
['burger', 'beryani']
```

[50]:
```
food = ["burger", "beryani", "raita"]
food.remove("burger")
print(food)
```

```
['beryani', 'raita']
```

[51]:
```
# append a item
food = ["burger", "beryani", "raita"]
print(food.append("karhaia"))
print(food)
```

```
None
['burger', 'beryani', 'raita', 'karhaia']
```

[52]:
```
# append items
message = []
message.append('beryani')
message.append('chicken')
print(message)
```

```
['beryani', 'chicken']
```

[53]:
```
# inserting elements which mean insert a new element in a specific position .
message = ['beryani', 'raita']
message.insert(1, 'chickedn')
```

```
print(message)
```

```
['beryani', 'chickedn', 'raita']
```

[54]:
```
# removing a elements by using delete command
message = ['beryani', 'rita']
del message[0]
print(message)  # ['rita']
```

```
['rita']
```

[55]:
```
# using poped method by removing the last item in a list
message1 = ['beryani', 'rita']
message = message1.pop(0)
print(message)  # Output: ['beryani']
```

```
beryani
```

[56]:
```
message1 = ['beryani', 'raita', 'krhai']
message = message1.pop(1)
print(f"The own things which is my first was {message.title()}")
```

```
The own things which is my first was Raita
```

[57]:
```
# give position of pop
message1 = ['beryani', 'raita', 'krhai']
message = message1.pop()
print(f"The own things which is my first was {message.title()}")
```

```
The own things which is my first was Krhai
```

[58]:
```
# removing elements
message1 = ['beryani', 'raita', 'krhai']
message1.remove('beryani')
print(message1)
```

```
['raita', 'krhai']
```

[59]:
```
# removing the elements.
food = ["burger", "beryani", "raita"]
food.remove('burger')
print(food)
```

```
['beryani', 'raita']
```

[ ]:
```
# sort -> This method is used for converting elements in a alphabetical orders␣
 ↪and that can not convert to other positions
cars = ['bmw', 'audi', 'yahma']
```

```
cars.sort()
print(cars)
```

```
['audi', 'bmw', 'yahma']
```

[61]:
```
# also we can reverse this method by using( reverse = True)
cars = ['bmw', 'audi', 'yahma']
cars.sort(reverse=True)
print(cars)
```

```
['yahma', 'bmw', 'audi']
```

[62]:
```
# by alphabat order
cars = ['bmw', 'audi', 'toyota', 'subaru']
print("Here is the original list:")
print(cars)
```

```
Here is the original list:
['bmw', 'audi', 'toyota', 'subaru']
```

[63]:
```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print("\n Here is the sorted list:")
print(sorted(cars))
```

```

 Here is the sorted list:
['audi', 'bmw', 'subaru', 'toyota']
```

[64]:
```
# reverse order
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.reverse()
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
```

[65]:
```
# using length of cars
cars = ['bmw', 'audi', 'toyota', 'subaru']
len(cars)
```

[65]:
```
4
```

[68]:
```
# avoiding index errors
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars[-1])
```

```
subaru
```

[71]:
```
# len,
list = [3, 4, 5, 6]
length = len(list)
print(length)
```

```
4
```

[72]:
```
# max
list = [200, 300, 400]
print(max(list))
```

```
400
```

[73]:
```
# min
list = [2000,3000,4000]
print(min(list))
```

```
2000
```

[74]:
```
# range
list = ['word', 'type']
print(list[1])
```

```
type
```

[76]:
```
# copy
list = ['word', 'type']
print(list.copy())
```

```
['word', 'type']
```

[77]:
```
# clear
list = ['word', 'type']
print(list.clear())
```

```
None
```

[78]:
```
# popp
list = ['word', 'type']
print(list.pop(1))
```

```
type
```

[79]:
```
# append
list = [200, 300, 400]
list.append(500)
print(list)
```

```
[200, 300, 400, 500]
```

```
[80]: # insert
list = [200, 300, 400]
list.insert(1, 250)
print(list)
```

```
[200, 250, 300, 400]
```

```
[81]: # index
list = [200, 300, 400]
print(list.index(300))
```

```
1
```

```
[82]: # slicing
my_list = [1, 2, 3, 4, 5]
print(my_list[1:4])  # Output: [2, 3, 4]
print(my_list[:3])   # Output: [1, 2, 3]
print(my_list[2:])   # Output: [3, 4, 5]
```

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

```
[83]: # Use in to check if an element exists in the list.

my_list = [1, 2, 3]
print(2 in my_list)
print(5 in my_list)
```

```
True
False
```

7 Exercise.

```
[84]: # 01. create a list
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
print(fav_foods)
```

```
['beryani', 'chicken', 'fish', 'chawal', 'raita']
```

```
[85]: # 02.select a second item
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
second_item = fav_foods[1]
print(second_item)
```

```
chicken
```

[86]:
```
# 03.  # Convert each item to uppercase
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
uppercase_foods = [food.upper() for food in fav_foods]
print(uppercase_foods)
```

```
['BERYANI', 'CHICKEN', 'FISH', 'CHAWAL', 'RAITA']
```

[87]:
```
# 03.  # Convert each item to uppercase
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
uppercase_foods = [foods.upper() for foods in fav_foods]
print(uppercase_foods)
```

```
['BERYANI', 'CHICKEN', 'FISH', 'CHAWAL', 'RAITA']
```

[88]:
```
# 04.add a another food
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
fav_foods.append('pizza')
print(fav_foods)
```

```
['beryani', 'chicken', 'fish', 'chawal', 'raita', 'pizza']
```

[89]:
```
# 05. delete 1 element
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
fav_foods.pop(0)
print(fav_foods)
```

```
['chicken', 'fish', 'chawal', 'raita']
```

[90]:
```
# 06.using pop
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
not_liked = fav_foods.pop(2) # not liked if fish
print("not liked food:", not_liked)
print("update fav_foods are:", fav_foods)
```

```
not liked food: fish
update fav_foods are: ['beryani', 'chicken', 'chawal', 'raita']
```

[91]:
```
# 07. sort a list in reverse order (1 and 2 both correct)
"""fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
fav_foods.sort(reverse=True)
print(fav_foods)  # Output: ['raita', 'chawal', '"""
letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G']

# Sort the list in reverse order
letters.sort(reverse=True)
print("Letters sorted in reverse order:", letters)
```

```
Letters sorted in reverse order: ['G', 'F', 'E', 'D', 'C', 'B', 'A']
```

[92]:
```
# 08.len
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
print(len(fav_foods))  # Output: 5
```

```
5
```

[93]:
```
# 09.get last item using (+ , and _ )index
fav_foods = ['beryani', 'chicken', 'fish', 'chawal', 'raita']
print(fav_foods[-1])  # prints: raita
print(fav_foods[4])
```

```
raita
raita
```

[94]:
```
# 10. Reverse the list using slicing
letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
reversed_letters = letters[::-1]
print(reversed_letters)  # Output: ['G', 'F', 'E', 'D
```

```
['G', 'F', 'E', 'D', 'C', 'B', 'A']
```

8 for loops

[96]:
```
# use for loops
friends = ['zohaib', 'Awais', 'ammar']
for friend in friends:
    print(friend,'is a good friend')
print('Done!')
```

```
zohaib is a good friend
Awais is a good friend
ammar is a good friend
Done!
```

[97]:
```
# use for loops
invitees = ['sarah', 'sarwat', 'sabra']
for invitee in invitees:
    #print('you are invited to a dinner', invitee)
    print(f'You are invited to a dinner, {invitee.title()}')
```

```
You are invited to a dinner, Sarah
You are invited to a dinner, Sarwat
You are invited to a dinner, Sabra
```

[98]:
```
# use for loops
for invite in invitees:
    print(f"you are invited to my party {invite.title()}")
    print(f'incase you cant come {invite.title()}, please let me know\n')
print(f'you are the best friend ever')
```

```
you are invited to my party Sarah
incase you cant come Sarah, please let me know

you are invited to my party Sarwat
incase you cant come Sarwat, please let me know

you are invited to my party Sabra
incase you cant come Sabra, please let me know

you are the best friend ever
```

[99]:
```
# use for loop in range
for value in range(2,10,2):
   print(value)
```

```
2
4
6
8
```

[104]:
```
squares = []
for value in range(1,12):
    square = value ** 3
    print(squares)
    squares.append(square)
print(squares)
```

```
[]
[1]
[1, 8]
[1, 8, 27]
[1, 8, 27, 64]
[1, 8, 27, 64, 125]
[1, 8, 27, 64, 125, 216]
[1, 8, 27, 64, 125, 216, 343]
[1, 8, 27, 64, 125, 216, 343, 512]
[1, 8, 27, 64, 125, 216, 343, 512, 729]
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331]
```

[105]:
```
# making suqraes of numbers
squares = []
for value in range(1,22,3):
    square = value ** 2
    squares.append(square)
print(squares)
```

```
[1, 16, 49, 100, 169, 256, 361]
```

[106]:
```
# use squre function
squares = []
for value in range(1,11):
    squares.append(value**2)
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[107]:
```
# use squre function
squares = []
for value in range(1, 11):
        square = value ** 2
        print(squares)
        squares.append(square)
        print(squares)
```

```
[]
[1]
[1]
[1, 4]
[1, 4]
[1, 4, 9]
[1, 4, 9]
[1, 4, 9, 16]
[1, 4, 9, 16]
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25, 36]
[1, 4, 9, 16, 25, 36]
[1, 4, 9, 16, 25, 36, 49]
[1, 4, 9, 16, 25, 36, 49]
[1, 4, 9, 16, 25, 36, 49, 64]
[1, 4, 9, 16, 25, 36, 49, 64]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[108]:
```
count = 0
for intervar in[2,33,44,55,66,77]:
    count+=1
    print("Count:", count)
```

```
Count: 1
```

```
Count: 2
Count: 3
Count: 4
Count: 5
Count: 6
```

[109]:
```python
# use count function
count = 0
for intervar in[3, 41, 12, 9, 74, 15]:
    count = count + 1
    print('count: ', count)
```

```
count:  1
count:  2
count:  3
count:  4
count:  5
count:  6
```

[110]:
```python
# use multiply function
multiply = [value*2 for value in range(5)]
print(multiply)
```

```
[0, 2, 4, 6, 8]
```

[111]:
```python
# use squares with one line of code
squares      =       [value*2 for value in range(1,11)]
print(squares)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

[112]:
```python
counts = 0
counts = [count + 1 for count in range(len([3, 41, 12, 9, 74, 15]))]
for c in counts :
    print('count:',c)
```

```
count: 1
count: 2
count: 3
count: 4
count: 5
count: 6
```

[113]:
```python
# use i
for i in 'python':
    print(i)
# use j
for j in range(5):
    print(j)

AnimalList = ['Cat','Dog','Tiger','Cow']
for x in AnimalList:
    print(x)
```

```
p
y
t
h
o
n
0
1
2
3
4
Cat
Dog
Tiger
Cow
```

[114]:
```
# use zip function
a1 = ['python', 'java', 'csharp']
b1 = [1,2,3]

for i,j in zip(a1,b1):
    print(i,j)
```

```
python 1
java 2
csharp 3
```

[115]:
```
# Using else statement inside a for loop in Python
flowers = ['Jasmine','Lotus','Rose','Sunflower']
for x in flowers:
    print(x)
else:
    print('Done!')
```

```
Jasmine
Lotus
Rose
Sunflower
Done!
```

[116]:
```python
list1 = [5,10,15,20]
list2 = ['Tomatoes','Potatoes','Carrots','Cucumbers']

for x in list1:
    for y in list2:
        print(x,y)
```

```
5 Tomatoes
5 Potatoes
5 Carrots
5 Cucumbers
10 Tomatoes
10 Potatoes
10 Carrots
10 Cucumbers
15 Tomatoes
15 Potatoes
15 Carrots
15 Cucumbers
20 Tomatoes
20 Potatoes
20 Carrots
20 Cucumbers
```

[117]:
```python
current_pop = 2000
for i in range(4, 0, -1):
    current_pop = current_pop/1.1
    print(i, current_pop)
```

```
4 1818.181818181818
3 1652.8925619834708
2 1502.629601803155
1 1366.0269107301408
```

[118]:
```python
# papolation 10% decrease
current_pap = 10000
for i in range(10,0,-1):
    current_pap = current_pap/1.1
    print(i,current_pap)
```

```
10 9090.90909090909
9 8264.462809917353
8 7513.148009015775
7 6830.134553650703
6 6209.213230591548
5 5644.739300537771
4 5131.5811823070635
3 4665.07380209733
```

```
2 4240.976183724845
1 3855.4328942953134
```

[119]:
```
# using break function
vehicles = ['cars', 'toyta','audi']
for v in vehicles:
    if v=='toyta':
        break
    print(v)
```

```
cars
```

9 while loops

[120]:
```
# give a table of 2 using while loops
num1 = int(input("Enter a number"))
i = 1
while i<11:
    print(num1, "*", i, "=", num1 * i)
    i+=1
```

```
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

[121]:
```
num1 = int(input("Enter  a number "))
i = 1
while i <= 11:
    print(num1, "*",i,"=", num1*i)
    i +=1
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
```

```
5 * 10 = 50
5 * 11 = 55
```

[122]:
```
current_number = 1
while current_number <=5:
    print(current_number)
    current_number += 1
```

```
1
2
3
4
5
```

[123]:
```
# use a Flag by using while loops
prompt = "\n Tell me something , and i will repeat itr back to you"
prompt += "\n Enter 'quit', to end programm "
active = True # flag
while active:
    message = input(prompt)
    if message == 'quit':
        active = False
    else:
        print(message)
```

```
Hi
```

[124]:
```
prompt = "\n Tell me"
prompt += "\n Enter 'quit' to end programm"
message = " "
while message != 'quit':
    message = input(prompt)
    print(message)
```

```
Awais
quit
```

[125]:
```
# loop with else
i = 1
while i <= 4:
    print(i)
    i += 1
else:
    print("loop finished")
```

```
1
2
3
```

```
4
loop finished
```

10 Exercise

[126]:
```
# 01.
for i in range(1,21):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

[138]:
```
# 02. cubes
cubes = []
for i in range(1,11):
    cubes.append(i**3)
for cube in cubes:
    print(cube)
```

```
1
8
27
64
125
216
343
512
729
1000
```

[140]:
```
# 03. Cube Comprehension

cubes = [i**3 for i in range(1, 11)]
print(cubes)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

[141]:
```
# 04. fav pizza

favorite_pizzas = ["pepperoni", "hawaiian", "meat lovers"]
for pizza in favorite_pizzas:
     print(f'i like : {pizza} pizza')
print(f'i like pizza !')
```

```
i like : pepperoni pizza
i like : hawaiian pizza
i like : meat lovers pizza
i like pizza !
```

11 preration for loops

[142]:
```
magicians = ['alice', 'david', 'caroline']
for magician in magicians:
    print(f'{magician.title()}, that was a great trick!')
    print(f'i cant wait to see your next trick,{magician.title()}.\n')
print("Thank you, everyone. That was a great magic show!")
```

```
Alice, that was a great trick!
i cant wait to see your next trick,Alice.

David, that was a great trick!
i cant wait to see your next trick,David.

Caroline, that was a great trick!
i cant wait to see your next trick,Caroline.

Thank you, everyone. That was a great magic show!
```

[143]:
```
# using range function.
for value in range(1,11):
    print(value)
```

```
1
2
3
4
5
```

```
6
7
8
9
10
```

[1]:
```
# list of range
message = list(range(1,5))
print(message)
```

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

[2]:
```
# square of numbers
squares = []
for value in range(1, 11):
                        square = value ** 2
                        squares.append(square)
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[3]:
```
# min max
digit = [1,2,3,4,5,6,7,8,9]
print(min(digit))
print(max(digit))
print(sum(digit))
```

```
1
9
45
```

[4]:
```
# silicing a list
players = ['babar', 'shaheen', 'malik','rizwan']
print(players[0:3])
print(players[1:4])
print(players[:4])
print(players[2:])
```

```
['babar', 'shaheen', 'malik']
['shaheen', 'malik', 'rizwan']
['babar', 'shaheen', 'malik', 'rizwan']
['malik', 'rizwan']
```

[5]:
```
# looping through silicing a list
players = ['babar', 'shaheen', 'malik','rizwan']

print("Here are the first four players on my team:")
```

```
for player in players[:4]:
    print(player.title())
```

```
Here are the first four players on my team:
Babar
Shaheen
Malik
Rizwan
```

[6]:
```
players = ['babar', 'rizwan', 'shaheen', 'amir']
print(players[-3:])
```

```
['rizwan', 'shaheen', 'amir']
```

[7]:
```
# looping through slicing.
players = ['babar', 'rizwan', 'shaheen', 'amir']
print("Here are the first three playeers on my team")
for player in players[:3]:
    print(player.title())
```

```
Here are the first three playeers on my team
Babar
Rizwan
Shaheen
```

[8]:
```
my_food = ['pizza','Tea','etc']
friend_foods = my_food[:1]
print("My favourite foods are:")
print(my_food)
print("\nMy friends favourite foods are:")
print(friend_foods)
```

```
My favourite foods are:
['pizza', 'Tea', 'etc']

My friends favourite foods are:
['pizza']
```

[9]:
```
rows = int(input('Enter a number of rows'))
i = 1
for i in range(i,rows+1):
    for j in range(1,i+1):
        print(j, end="")
    for k in range(i-1,0,-1):
        print(k, end="")

print()
```

```
1121123211234321123454321
```

[11]:
```python
# code for triangle
a = int(input("Enter the first angle: "))
b = int(input("Enter the second angle: "))
c = int(input("Enter the third angle: "))
if a+b+c == 180:
    print("It can form a triangle")
else:
    print("It cannot form a triangle")
```

```
It can form a triangle
```

12 Rows pattern using loops

[16]:
```python
# rows pattern
rows = int(input("Enter the number of rows "))
for i in range(1, rows+1):
    for j in range(1, i+1):
        print('*', end='')
    print()
```

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

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

[17]:
```python
# rows pattern
rows = int(input("Enter the number of rows "))
for i in range(1, rows+1):
    for j in range(1, i+1):
        print(j, end='')
    for k in range(i-1, 0, -1):
        print(k, end='')
    print()
```

```
1
121
12321
1234321
123454321
12345654321
1234567654321
123456787654321
12345678987654321
1234567891010987654321
12345678910111110987654321
1234567891011121110987654321
```

13 Loop Control Statement

Break Continue Pass

[18]:
```python
# break
lower = int(input('enter lower range'))
upper = int(input('enter upper range'))

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

```
5
7
```

```
[19]: # Continue
      for i in range(1,10):
        if i == 5:
          continue
        print(i)
```

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

14 Tuples

- tuples canot change able
- its immutable
- tuples are faster than lists
- tuples are more memory efficient than lists

```
[3]: message = (9200,300)
     print(message[0]) #9200
     print(message[1]) #300
```

```
9200
300
```

```
[4]: # looping through tuples
     tup = (1, 2, 3, 4, 5)
     for x in tup:
         print(tup)

     dimensions = (200,50)
     print("Original dimensions")
     for dimension in dimensions:
         print(dimension)

     dimensions = (400,70)
     print("\nmodified dimension :")
     for dimension in dimensions:
         print(dimension)
```

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

```
(1, 2, 3, 4, 5)
Original dimensions
200
50

modified dimension :
400
70
```

[6]:
```
# loops in reverse order
i = 10
while i >= 1:
    print(i)
    i -=1
```

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

[7]:
```
# print a table of 3 or multiplication role
i = 1
while i <= 10:
    print(3 , '*', i, '=',  3* i)
    i += 1
```

```
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

[8]:
```
numbers = [1,4,16,64,81,100]

index = 0
while index < len(numbers):
    print(numbers[index])
    index += 1
```

```
1
4
16
64
81
100
```

[9]:
```
# use for loops
numbers = (1,22,44,55, 66, 36,50,77,36)
x = 36
index = 0
for value in numbers:
    index+=1
    if(value == x):
        print("Found at the index", index)
        break

    else:
        print("The end")
```

```
The end
The end
The end
The end
The end
Found at the index 6
```

[10]:
```
# use squre function
squares = []
for value in range(1, 11):
        square = value ** 2
        print(squares)
        squares.append(square)
        print(squares)
```

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

```
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25, 36]
[1, 4, 9, 16, 25, 36]
[1, 4, 9, 16, 25, 36, 49]
[1, 4, 9, 16, 25, 36, 49]
[1, 4, 9, 16, 25, 36, 49, 64]
[1, 4, 9, 16, 25, 36, 49, 64]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[ ]:
```
#  generate tables of 2 and yields following
print("Table of 2")
print("............")
for i in range(1, 13):
    product = 2 *i
    print(f"2  x {i} = {product}")

#
```

```
Table of 2
…
2  x 1 = 2
2  x 2 = 4
2  x 3 = 6
2  x 4 = 8
2  x 5 = 10
2  x 6 = 12
2  x 7 = 14
2  x 8 = 16
2  x 9 = 18
2  x 10 = 20
2  x 11 = 22
2  x 12 = 24
```

[12]:
```
# create a table of 10
print("Table of 10")
print("..........")
for i in range(1,11):
    product=i*10
    print(f"10 x {i} = {product}")

print("Done!")
```

```
Table of 10
…
10 x 1 = 10
10 x 2 = 20
```

```
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
Done!
```

[13]:
```python
# Table of 5.
print("Table of 5")
print("............")
for i in range(1, 13):
    product = 5 *i
    print(f"5  x {i} = {product}")
```

```
Table of 5
…
5  x 1 = 5
5  x 2 = 10
5  x 3 = 15
5  x 4 = 20
5  x 5 = 25
5  x 6 = 30
5  x 7 = 35
5  x 8 = 40
5  x 9 = 45
5  x 10 = 50
5  x 11 = 55
5  x 12 = 60
```

[14]:
```python
# using a integer input.
numbers = int(input("enter a number"))
for i in range(1,13):
    print(f" {numbers} x {i} = {numbers*i}")
```

```
 5 x 1 = 5
 5 x 2 = 10
 5 x 3 = 15
 5 x 4 = 20
 5 x 5 = 25
 5 x 6 = 30
 5 x 7 = 35
 5 x 8 = 40
 5 x 9 = 45
 5 x 10 = 50
 5 x 11 = 55
```

```
5 x 12 = 60
```

15 Dictionary

[15]:
```
# Create a list of 30 aliens
aliens = []
for alien_number in range(30):
    new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'}
    aliens.append(new_alien)

# Modify the first 3 aliens
for alien in aliens[:3]:
    if alien['color'] == 'green':
        alien['color'] = 'yellow'
        alien['speed'] = 'medium'
        alien['points'] = 10

# Print the first 5 aliens

for alien in aliens[:5]:
    print(alien)

    print("...")
```

```
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
…
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
…
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
…
{'color': 'green', 'points': 5, 'speed': 'slow'}
…
{'color': 'green', 'points': 5, 'speed': 'slow'}
…
```

[16]:
```
# Define the pizza order
pizza = {
    'crust': 'thick',
    'toppings': ['mushrooms', 'extra cheese']
}

# Summarize the order
print(f"You ordered a {pizza['crust']}-crust pizza ")
print("with the following toppings:")

# List each topping
for topping in pizza['toppings']:
    print(f"- {topping}")
```

```
You ordered a thick-crust pizza
with the following toppings:
- mushrooms
- extra cheese
```

[17]:
```
fav_languages = {
    'jen' : ['python', 'rust'],
    'sarah' : ['c', 'java'],
    'edward' : ['ruby', 'go'],
    'john' : ['python', 'swift'],
}

# use for loops
for name, languages in fav_languages.items():
    print(f"\n{name.title()}'s favorite languages are:")
    for language in languages:
        print(f"\t{language.title()}")
```

```

Jen's favorite languages are:
        Python
        Rust

Sarah's favorite languages are:
        C
        Java

Edward's favorite languages are:
        Ruby
        Go

John's favorite languages are:
        Python
        Swift
```

16 Tuples prepration.

[18]:
```
# 1. create a tuple
tup = (2,3,4,5)
print(tup)
```

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

[19]:
```python
# 2. diffrent types of tuples
tup1 = ( "Hello", 33, complex,)
print(type(tup1))
```

```
<class 'tuple'>
```

[20]:
```python
# get 4th elements of tuple
tup2 = (3,4,5,6,7)
print(tup2[4])
```

```
7
```

[21]:
```python
# exists elemnt in a list
my_tup3 = (1,2,3,4,5)
element = 3
if element in my_tup3:
    print(f"{element} exists in the tuple")
else:
    print(f"{element} does not exists tuple")
```

```
3 exists in the tuple
```

[22]:
```python
# convert a list into tuple.

my_list = [1,2,3,45,4]
my_tuple = tuple(my_list)
print(my_tuple)
```

```
(1, 2, 3, 45, 4)
```

[23]:
```python
# convert a list to atuple (alternative)
my_list = [1,2,3,45,4]
my_tuple= (*my_list,)
print(my_tuple)
```

```
(1, 2, 3, 45, 4)
```

[24]:
```python
# reverse a tuple
my_tuple = (1,2,3,4,5,56)
reverse_tuple = tuple(reversed(my_tuple))
print(reverse_tuple)
```

```
(56, 5, 4, 3, 2, 1)
```

[25]:
```python
# reverse a tuple
my_tuple = (1,2,3,4,5,56)
reverse_tuple = my_tuple[::-1]
print(reverse_tuple)
```

```
(56, 5, 4, 3, 2, 1)
```

[26]:
```
# copy elements of 44 and 55
tuple1 = (11, 22, 33, 44, 55, 66)
tuple2 = (tuple1[3], tuple1[4])
print(tuple2)
```

```
(44, 55)
```

17 Conditional satetment

[27]:
```
# conditional if elif else
a = 5
b = 4
c = 3

if a < b and a < c:
    print("a is smaller than b and c")
elif b < c:
        print("b is smaller than c")
else:
        print("c is the smallest")
```

```
c is the smallest
```

18 Module in python

[28]:
```
# math
import math
print(math.cos(90))

# keywords
import keyword
print(keyword.kwlist)

# random
import random
print(random.randint(1, 10))

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

```
-0.4480736161291701
['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']
10
2025-09-24 11:54:29.346105
```

```
[29]: # 10% decrease population
current_pap = 10000
for i in range(10,0,-1):
    print(i,current_pap)
    current_pap = current_pap/1.1
```

```
10 10000
9 9090.90909090909
8 8264.462809917353
7 7513.148009015775
6 6830.134553650703
5 6209.213230591548
4 5644.739300537771
3 5131.5811823070635
2 4665.07380209733
1 4240.976183724845
```

```
[30]: import datetime
print(datetime.datetime.now())
```

```
2025-09-24 11:54:34.426718
```

19 Nested loops

```
[31]: 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
```

20 Dictionary

```
[ ]: # 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"""
```

```
[36]: # empty dictionary
      d = {}
      d
      # 1D dictionary
      d1 = { 'name' : 'Awais' ,'gender' : 'male' }
      d1
      # with mixed keys
      d2 = {(1,2,3):1,'hello':'world'}
      d2
      # 2D dictionary -> JSON
      s = {
          'name':'Awais',
           'college':'Comsats',
           'sem':4,
           'subjects':{
               'dsa':50,
               'maths':67,
               'english':34
           }
      }
      s
      # using sequence and dict function
      d4 = dict([('name','nitish'),('age',32),(3,3)])
      d4
      # duplicate keys
      d5 = {'name':'Awais','name':'Ali'}
```

```
d5
# mutable items as keys
d6 = {'name':'Awais',(1,2,3):2}
print(d6)
```

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

[37]:
```
# Accessing items
my_dict = {'name': 'Awais', 'age': 26}
# []
my_dict['age']
# get
my_dict.get('age')

s['subjects']['maths']
```

[37]:
```
67
```

[38]:
```
# Removing key,value pairs
d = {'name': 'Awais ', 'age': 20, 3: 3, 'gender': 'male', 'weight': 50}
# pop
#d.pop(3)
#print(d)
# popitem
#d.popitem()
# d.popitem()
# print(d)
# del
#del d['name']
#print(d)
# clear
d.clear()
print(d)

del s['subjects']['maths']
s
```

```
{}
```

[38]:
```
{'name': 'Awais',
 'college': 'Comsats',
 'sem': 4,
 'subjects': {'dsa': 50, 'english': 34}}
```

[39]:
```
# Editing
s['subjects']['dsa'] = 80
s
```

```
print(s)

'name' in s
```

```
{'name': 'Awais', 'college': 'Comsats', 'sem': 4, 'subjects': {'dsa': 80,
'english': 34}}
```

```
[39]: True
```

```
[40]: # items/keys/values
print(d)

print(d.items())
print(d.keys())
print(d.values())
```

```
{}
dict_items([])
dict_keys([])
dict_values([])
```

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

```
[41]: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

```
[43]: # using existing dict
distances = {'Islamabad':1000,'Taunsa':2000,'Lahore':3000}
{key:value*0.62 for (key,value) in distances.items()}
```

```
[43]: {'Islamabad': 620.0, 'Taunsa': 1240.0, 'Lahore': 1860.0}
```

```
[44]: # 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)}
```

```
[44]: {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}}
```

```
[45]: # 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)}
```

```
[45]: {'Sunday': 30.5,
       'Monday': 32.6,
       'Tuesday': 31.8,
       'Wednesday': 33.4,
       'Thursday': 29.8,
       'Friday': 30.2,
       'Saturday': 29.9}
```

```
[46]: # update
      d1 = {1:2,3:4,4:5}
      d2 = {4:7,6:8}

      d1.update(d2)
      print(d1)
```

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

```
[47]: #Adding key-value pair
      d4['gender'] = 'male'
      d4
      d4['weight'] = 72
      d4

      s['subjects']['ds'] = 75
      s
```

```
[47]: {'name': 'Awais',
       'college': 'Comsats',
       'sem': 4,
       'subjects': {'dsa': 80, 'english': 34, 'ds': 75}}
```

```
[48]: alien_0 = {'color':'green', 'points': 5}
      print(alien_0['color'])
      print(alien_0['points'])
      new_points = alien_0['points']
      print(f'you just earned {new_points} points!')
```

```
green
5
you just earned 5 points!
```

```
[49]: alien_0 = {'color':'green', 'points': 5}
      print(alien_0['color'])

      alien_0['x position'] = 0
      alien_0['y position'] = 25
      print(alien_0)
```

```
green
{'color': 'green', 'points': 5, 'x position': 0, 'y position': 25}
```

[50]:
```
# Removing
alien_0 = {'color': 'green', 'points': 5}
print(alien_0)
del alien_0['points']
print(alien_0)
```

```
{'color': 'green', 'points': 5}
{'color': 'green'}
```

[51]:
```
# using get
alien_0 = {'color': 'green', 'speed': 'slow'}
point_value = alien_0.get('points', 90)
print(point_value)
```

```
90
```

[52]:
```
#for loop:
user_0 = {
 'username': 'efermi',
 'first': 'enrico',
 'last': 'fermi',
 }
for key, value in user_0.items():
 print(f"\nKey: {key}")
 print(f"Value: {value}")
```

```

Key: username
Value: efermi

Key: first
Value: enrico

Key: last
Value: fermi
```

[53]:
```
fav_languages = {
    'awais' : 'python',
    'ali' : 'java',
    'ahsan' : 'c++',
    'khan' : 'c#',
}

for name, language in fav_languages.items():
    print(f"{name.title()} likes {language.title()}")  # title() function is␣
↪used
```

```
Awais likes Python
Ali likes Java
Ahsan likes C++
Khan likes C#
```

[54]:
```
favorite_languages = {
    "jan" : "python",
    "sarah" : "c",
    "edward" : "rust",
    "phil" : "python",
}
```

[55]:
```
friends = ["phil","sarah"]
for name in favorite_languages.keys():
    print(f"Hi {name.title()}.")
    if name in friends:
        language = favorite_languages[name].title()
        print(f"\t{name.title()}, I see you love {language}")
```

```
Hi Jan.
Hi Sarah.
        Sarah, I see you love C
Hi Edward.
Hi Phil.
        Phil, I see you love Python
```

[56]:
```
users = {
    'aeinstein' : {
        'first' : 'Albert',
        'last' : 'Einstein',
        'location' : 'princenton',
    },
    'mcurie' : {
        'first' : 'Marie',
        'last' : 'Curie',
        'location' : 'paris',
    },
}

for username, user_info in users.items():
    print(f"Username: {username}")
    full_name = f"{user_info['first']}{user_info['last']}"
    location = user_info['location']
    print(f"Full Name: {full_name}")
    print(f"location {location}")
```

```
Username: aeinstein
Full Name: AlbertEinstein
location princenton
Username: mcurie
Full Name: MarieCurie
location paris
```

[57]:
```
users = {
    'aiensteian' : {
        'first' : 'Albert',
        'last': 'curie',
        'age' : 50,
    },
    'muerice' : {
        'first': 'Marie',
        'last': 'curie',
        'age' : 60,
    },
}
for username,user_info in users.items():
    print(f"username: {username}")
    fullname = f"{user_info['first']}{user_info['last']}"
    age = user_info['age']
    print(f"fullname{fullname}")
    print(f"age{age}")
```

```
username: aiensteian
fullnameAlbertcurie
age50
username: muerice
fullnameMariecurie
age60
```

[58]:
```
# user input
prompt = "if you share your name , we can personalize the message you see"
prompt += 'What is you first name '
name = input(prompt)
print(f"\nHello, {name}")
```

```

Hello, HI
```

21 Exercise

[60]:
```
# Pizza Toppings:
while True:
    topping = input("Enter a pizza topping (or 'quit' to stop): ")
    if topping == 'quit':
        break
    print(f"I'll add {topping} to your pizza.")
```

```
I'll add Break to your pizza.
```

[61]:
```
while True:
    age = input("Enter your age(or 'quit' to stop)")
    if age == 'quit':
        break
    age = int(age)

    if age <=3:
        print("Tickt cost is free")
    elif    3<= age <= 12:
        print("Ticket cost is $10")
    else:
        print("Ticket cost is $15")
```

```
Ticket cost is $10
Ticket cost is $10
Ticket cost is $15
Ticket cost is $15
```

[62]:
```
# Write a loop to calculate the price of movie tickets based on age
while True:
    age = input("Enter your age (or 'quit' to stop): ")
    if age == 'quit':
        break

    age = int(age)

    if age < 3:
        print("Your ticket is free.")
    elif 3 <= age <= 12:
        print("Your ticket costs $10.")
    else:
        print("Your ticket costs $15.")
```

```
Your ticket costs $10.
Your ticket costs $15.
```

[64]:
```python
# Make a list of sandwich orders

sandwich_orders = ['tuna', 'ham', 'chicken', 'veggie', 'cheese']
finished_sandwiches = []

# Loop through sandwich orders
while sandwich_orders:
    current_sandwich = sandwich_orders.pop()
    print(f"I made your {current_sandwich} sandwich.")
    finished_sandwiches.append(current_sandwich)

# Print finished sandwiches
print("\nAll sandwiches made:")
for sandwich in finished_sandwiches:
    print(sandwich)
```

```
I made your cheese sandwich.
I made your veggie sandwich.
I made your chicken sandwich.
I made your ham sandwich.
I made your tuna sandwich.

All sandwiches made:
cheese
veggie
chicken
ham
tuna
```

[65]:
```python
# List of sandwich orders with 'pastrami' appearing multiple times
sandwich_orders = ['tuna', 'pastrami', 'ham', 'pastrami', 'chicken',␣
  ↪'pastrami', 'veggie']
finished_sandwiches = []

# Print message that pastrami is out of stock
print("Sorry, the deli has run out of pastrami.\n")

# Remove all occurrences of 'pastrami'
while 'pastrami' in sandwich_orders:
    sandwich_orders.remove('pastrami')

# Loop through remaining sandwich orders
while sandwich_orders:
    current_sandwich = sandwich_orders.pop()
    print(f"I made your {current_sandwich} sandwich.")
    finished_sandwiches.append(current_sandwich)
```

```python

# Print finished sandwiches
print("\nAll sandwiches made (no pastrami):")
for sandwich in finished_sandwiches:
    print(sandwich)
```

```
Sorry, the deli has run out of pastrami.

I made your veggie sandwich.
I made your chicken sandwich.
I made your ham sandwich.
I made your tuna sandwich.

All sandwiches made (no pastrami):
veggie
chicken
ham
tuna
```

[66]:
```python
dream_vacation = {}
while True:
    name = input("Enter the your name(or 'quit)")
    if name == 'quit':
        break
    place = input("if you could visit in the world where could it be?")
    dream_vacation[name] = place
    repeat = input("would you like to let another person ?(yes/no)")
    if repeat == 'no':
        break
    for name, place in dream_vacation.items():
        print(f"{name} could like to visit {place}")
```

```
Awais could like to visit Islamabad
```

22 Try Exercise

```
[68]: # 01Add a key to a dictionary

      # Sample Dictionary
      dictionary = {0: 10, 1: 20}

      # Add a new key-value pair. Alternatively,
      dictionary.update({2: 30})

      # dictionary[2] = 30

      print(dictionary)
```

```
{0: 10, 1: 20, 2: 30}
```

```
[69]: # 02 Concatenate dictionaries*

      dic1 = {1: 10, 2: 20}
      dic2 = {3: 30, 4: 40}
      dic3 = {5: 50, 6: 60}

      dic4 = {}
      for d in (dic1, dic2, dic3):
          dic4.update(d)
      print(dic4)
```

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

```
[ ]: # 03 Check if a key exists in the dictionary*



     dictionary = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
     key_to_check = 1
     value = dictionary.get(key_to_check)
     if value is not None:
         print(f"key exists{key_to_check} with value {value}")
     else:
         print(f"key does not exist{key_to_check}")


     #if key_to_check in dictionary:
     #    print(f"The key {key_to_check} exists with value␣
      ↪{dictionary[key_to_check]}")
     #else:
     #    print(f"The key {key_to_check} does not exist")

     key_to_check = 4
     value = dictionary.get(key_to_check)
```

```
if value is not None:
    print(f"the key{key_to_check} exists with value{value}")
else:
    print(f"The key {key_to_check} does not exist")
key_to_check = 30
value = dictionary.get(key_to_check)
if value is not None:
    print(f"The key {key_to_check} exists with value {value}")
else:
    print(f"The key {key_to_check} does not exist")
```

```
key exists1 with value 10
the key4 exists with value40
The key 30 does not exist
```

[71]:
```
# 04 Iterate over using for loops



d = {'x': 10, 'y': 20, 'z': 30}

# Iterate over keys
for key in d.keys():
    print(key)

# Iterate over values
for value in d.values():
    print(value)

# Iterate over key-value pairs
for key, value in d.items():
    print(f"{key}:{value} ")
```

```
x
y
z
10
20
30
x:10
y:20
z:30
```

[72]:
```
# 05 dictionary with squares of numbers



square_dict = {x: x**2 for x in range(1, 16)}
```

```
print(square_dict)


#Alternatively, using a for loop:
#square_dict = {}
#for x in range(1, 16):
#    square_dict[x] = x**2
#print(square_dict)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121,
12: 144, 13: 169, 14: 196, 15: 225}
```

[73]:
```
# 06  Write a Python program to remove a key from a dictionary.
my_dict = {"name" : "Zohaib", "points" : 5}
del my_dict["name"]
print(my_dict)  # Output: {'points': 5}
```

```
{'points': 5}
```

[74]:
```
# 7. Write a Python program to map two lists into a dictionary.
keys = ['red', 'green', 'blue']
values = [1, 2, 3]
# Using dictionary comprehension to map keys and values into a dictionary
result = dict(zip(keys,values))
# Printing the result
print(result)
```

```
{'red': 1, 'green': 2, 'blue': 3}
```

[75]:
```
#8.        Write a Python program to sort a given dictionary by key.
color_dict = {'red' : 1,'green' : 2,'black' : 3}
for key in sorted(color_dict):
    print(f"{key} : {color_dict[key]}")
```

```
black : 3
green : 2
red : 1
```

[76]:
```
# 9.        Write a Python program to remove duplicates from the dictionary.
student_data = {'id1':
   {'name': ['Sara'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
 'id2':
  {'name': ['David'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
 'id3':
    {'name': ['Sara'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
 'id4':
   {'name': ['Surya'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
}

result = {}

for key,value in student_data.items():
    if value not in result.values():
        result[key] = value

print(result)
```

```
{'id1': {'name': ['Sara'], 'class': ['V'], 'subject_integration': ['english,
math, science']}, 'id2': {'name': ['David'], 'class': ['V'],
'subject_integration': ['english, math, science']}, 'id4': {'name': ['Surya'],
'class': ['V'], 'subject_integration': ['english, math, science']}}
```

23 Conditional satement.

[77]:
```
# if with for loops
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
```

```
Audi
BMW
Subaru
Toyota
```

[78]:
```
# check equality.
cars = 'bmw'
print(cars == 'bmw')  # True

# check inequality
cars = 'audi'
```

```
print(cars == 'bmw')  # False

# case sensetive also give you False.
cars = 'Audi'
print(cars == 'audi')  # False
```

```
True
False
False
```

[79]:
```
# The two strings when match give True answer
car = 'Audi'
car.lower() == 'audi'
#True
print(car)
```

```
Audi
```

[80]:
```
# checking nequality.
requested_topping = 'mushrooms'
if requested_topping != 'anchovies':
 print("Hold the anchovies!")
```

```
Hold the anchovies!
```

[81]:
```
# Numeric function
answer = 17
if answer != 42:
 print("You are wrong!.please try again")

# Mathametical comperesion
age = 33
age > 50

age >= 50
age < 50
```

```
You are wrong!.please try again
```

[81]:
```
True
```

24 Operator and, or, not.

[82]:
```
# and operator
age_1 = 22
age_2 = 18
# Using the and operator to check if both conditions are met
```

```
if age_1 > 18 and age_2 > 18:
    print("Both are adults")
else:
    print("Not both are adults")
```

```
Not both are adults
```

[83]:
```
# other method
age_1 =  22
age_2 =  18

if age_1 >= 21 and age_2 >= 21:
    print("Adult")
else:
    print("Not Adult")
```

```
Not Adult
```

[84]:
```
# or operator
age_1 = 22
age_2 = 18
if age_1 >21 or age_2 >21:
    print("At least one of the ages is greater than 21") # Output: At least one
else:
    print("Both are incorrecrt")
```

```
At least one of the ages is greater than 21
```

[85]:
```
# To find
age = [19,20,39,40]
19 in age
```

[85]:
```
True
```

[86]:
```
# is not in
banned_users = ['andrew', 'carolina', 'david']
user = 'marie'
if user not in banned_users:
 print(f"{user.title()}, you can post a response if you wish.")
```

```
Marie, you can post a response if you wish.
```

25 Prepration of IF elif else:

[87]:
```
# 01 Assign the alien color to 'green'

alien_color = ['green', 'yellow', 'red']
alien_color = 'green'
if alien_color == 'green':
    print("The player just earned 5 points for shooting the alien.")

# Assign the alien color to 'red'
alien_color = 'red'

#  if the alien color is green
if alien_color == 'green':
    print("The player just earned 5 points!")
```

```
The player just earned 5 points for shooting the alien.
```

[88]:
```
# 02 Assign alien color to green
alien_color = 'red'
if alien_color == 'green':
    print("The player just earned 5 points for shooting the alien")
else:
    print("The player just earned 10 points!")
```

```
The player just earned 10 points!
```

[89]:
```
#03 using if elif else
alien_color == 'red'
if alien_color == 'green':
    print("The player earned 5 points.")
elif alien_color == 'yellow':
    print("The player earned 10 points.")
else :
    print("The player earned 15 points.")
```

```
The player earned 15 points.
```

[90]:
```
# 04 Determine the person's stage of life
# use user inputs

age = int(input("Enter your age: "))

if age < 2:
    print("The person is a baby.")
elif age >= 2 and age < 4:
    print("The person is a toddler.")
```

```
elif age >= 4 and age < 13:
    print("The person is a kid.")
elif age >= 13 and age < 20:
    print("The person is a teenager.")
elif age >= 20 and age < 65:
    print("The person is an adult.")
else:
    print("The person is an elder.")
```

```
The person is a toddler.
```

26 Function

[91]:
```
# Positional arguments
def favorite_book(management, python):
    management = "The Alchemist"
    python = "Python Crash Course"
    """Display a favorite book"""
    print("\nFavorite Book".title())
    print(f"My favorite books are '{management}' and '{python}'")
favorite_book("The Alchemist", "Python Crash Course")
```

```

Favorite Book
My favorite books are 'The Alchemist' and 'Python Crash Course'
```

[92]:
```
# making a function
def is_even(num):
    """ print output if number is even or odd """
    if type(num) == int:
        if num % 2 == 0:
            return "even"
        else:
            return "odd"
    else:
        return "pagal ha kayia "
for i in range(1,11):
    x = is_even(i)
    print(x)
```

```
odd
even
odd
even
odd
even
```

```
odd
even
odd
even
```

[93]:
```python
# making a function
# use args for allow any value of keyword arguments
def multiply(* args):
    product = 1
    for i in args:
        product =product * i
        print(args)
        return product
    # calling a function
multiply(2,4)
```

```
(2, 4)
```

[93]: 2

[94]:
```python
# making a function
def is_even(num):
    """
    chechk if function is even or odd
     input- allow any valid integars
     output- even/odd
     """
    if type(num) == int:
        if num % 2==0:
            return "even"
        else:
            return "odd"
    else:
        return "pagal ha kia"
    # calling function
for i in range(1,11):
    x = is_even(i)
    print(x)
is_even("hello")
```

```
odd
even
odd
even
odd
even
odd
even
```

```
odd
even
```

[94]:
```
'pagal ha kia'
```

[97]:
```python
# converter function
def converter(feet_value):
    cm_value = feet_value * 83.99
    print(feet_value, "Feet=", cm_value, "cm")
converter(14)
```

```
14 Feet= 1175.86 cm
```

[98]:
```python
def converter(Usd_value):
    Inr_value = Usd_value * 400
    print(Usd_value, "USD =", Inr_value, "INR")

# Prompt the user for input
Usd_value = float(input("Enter the value in USD: "))
converter(Usd_value)
```

```
5.0 USD = 2000.0 INR
```

[99]:
```python
def converter(Usd_value):
    Inr_value = Usd_value * 400
    print(Usd_value, "USD =", Inr_value, "INR")
converter(13333)
```

```
13333 USD = 5333200 INR
```

[100]:
```python
def convrt(feet):
    cm = feet * 30.48
    print(feet, "feet=", cm, "cm")
convrt(1000)
```

```
1000 feet= 30480.0 cm
```

[102]:
```python
# function defination it is block of satetement that perform a specific task
# reduce redendency
def cal_sum(a, b): #parameters
    """Display a sum of two numbers """
    return a + b
sum = cal_sum(2200,3300)
print(sum)
```

```
5500
```

[103]:
```python
# average of 3 numbers
def calc_avg(a,b,c):
    return (a+b+c)/3
# calculate average of 3 numbers
print(calc_avg(10,20,30))  # Output: 20.0

# other method
def calc_avg(a,b,c):
    sum = a+b+c
    avg = sum/3
    return avg
calc_avg(98, 97, 95)
```

```
20.0
```

[103]:
```
96.66666666666667
```

[104]:
```python
def multiply(x, y):
    """Return the product of x and y"""
    return x * y

x = int(input("Input a number: "))
y = int(input("Enter another number: "))

print("Result: ", multiply(x, y))
```

```
Result:  12
```

[105]:
```python
# Function types
# Built in and user defind

print("Hello", end="")
print("World")
```

```
HelloWorld
```

[106]:
```python
# Defalt parameters
def cal_prod(a=1,b=1):
    print(a * b)
    return a * b
# Call the function with default parameters
cal_prod(6,7)
```

```
42
```

[106]:
```
42
```

[108]:
```
def cal_prod(a=1,b=1):

    return a*b

cal_prod(2,4)
```

[108]:
```
8
```

[109]:
```
cities = ['Taunsa', 'Karachi', 'Islamabd']

def print_len(list):
    print(len(list))

print_len(cities)
```

```
3
```

[110]:
```
# in a single line printing
cities = ['Taunsa', 'Karachi', 'Islamabd']
def print_list(list):
    for item in list:
        print(item, end=" ")
print_list(cities)
```

```
Taunsa Karachi Islamabd
```

[111]:
```
# factorial calling
def calc_fact(n):
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    print(fact)

calc_fact(3)
```

```
6
```

[112]:
```
# USD TO INR
def converter(usd_val):
    inr_val = usd_val * 74.5
    print(usd_val, "USD=", inr_val, "INR")

converter(1)
```

```
1 USD= 74.5 INR
```

[113]:
```
def converter(usd_value):
    inr_value = usd_value *80
    print(usd_value, "USD=", inr_value, "INR")
converter(100)
```

```
100 USD= 8000 INR
```

27 Recursion

[114]:
```
def show(n):
    if n == 0:
        return 0
    print(n)
    show(n-1)
show(10)
```

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

[115]:
```
# Recursion is same as it is loops
def show(n):
    if n == 0:
        return
    print(n)
    show(n-1)

show(5)
```

```
5
4
3
2
1
```

[116]:
```
def fact(n):
    if(n == 0 or n == 1):
        return 1
    else:
        return (n-1) * n
```

```
print(fact(6))
```

```
30
```

[117]:
```
# lets create  a function ( with docstring)
def is_even(num):
    """
    This function return if the logic is even or oddd number
    inut ny valid int
    output odd or even
    created on 16th nov 2024
    """
    if num % 2 == 0:
        return "even"
    else:
        return "odd"
    # lets test the function
for i in range(1,11):
    x = is_even(i)
    print(x)

print(x.__doc__)
print(type.__doc__)
```

```
odd
even
odd
even
odd
even
odd
even
odd
even
str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str

Create a new string object from the given object. If encoding or
errors is specified, then the object must expose a data buffer
that will be decoded using the given encoding and error handler.
Otherwise, returns the result of object.__str__() (if defined)
or repr(object).
encoding defaults to sys.getdefaultencoding().
errors defaults to 'strict'.
type(object) -> the object's type
type(name, bases, dict, **kwds) -> a new type
```

28 Types of Arguments

.Default .positional .Keyword

[118]:
```
# Default Arguments.
def power (a=1,b=1):
    return a ** b
power()
```

[118]:
```
1
```

[119]:
```
# positional Arguments
power(2,3)
```

[119]:
```
8
```

[120]:
```
# Keyword Arguments
power(b=3,a=2)
```

[120]:
```
8
```

29 Args and Kwargs

[121]:
```
# Args
# allows us to pass a variable no of non keywords arguments to a function
def multiply(*args):
    product = 1
    for i in args:
        product = product * i
    return product
# calling the function with variable number of arguments
multiply(33,44,55,66,77,88)
```

[121]:
```
35714669760
```

[122]:
```
# Kwargs
# It is used to pass any  of keyword arguments to a function.
def display(** kwargs):
    for key, value in kwargs.items():
        print(key,'->', value)

display(india='delhi', pakistan='islamabad')
```

```
india -> delhi
pakistan -> islamabad
```

30 Benefits of using a Function

Code Modularity Code Readibility Code Reusability

[123]:
```
# x,y -> x+y
a = lambda x,y:x/y
a(5,2)
```

[123]:
```
2.5
```

[124]:
```
# check if a string has 'a'
a = lambda s:'a' in s
a('hello')

# odd or even
a = lambda x:'even' if x%2 == 0 else 'odd'
a(6)
```

[124]:
```
'even'
```

[125]:
```
# Higher Order Functions
# Example

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)
```

```
[1, 8, 27, 64, 125]
```

31 Map

[126]:
```
#1. square the items of a list
list(map(lambda x:x**2,[1,2,3,4,5]))

# 2odd/even labelling of list items
L = [1,2,3,4,5]
list(map(lambda x:'even' if x%2 == 0 else 'odd',L))

# 3fetch 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'
    }
]

list(map(lambda users:users['gender'],users))
```

[126]:
```
['male', 'male', 'female']
```

32 Lambda

[127]:
```
"""No name
lambda has no return value(infact,returns a function)
lambda is written in 1 line
not reusabl"""
# odd or even
a = lambda x:'even' if x%2 == 0 else 'odd'
a(6)
```

[127]:
```
'even'
```

33 Filter

```
[128]: # numbers greater than 5
       L = [3,4,5,6,7]

       list(filter(lambda x:x>5,L))
```

```
[128]: [6, 7]
```

```
[129]: # fetch fruits starting with 'a'
       fruits = ['apple','guava','cherry']

       list(filter(lambda x:x.startswith('a'),fruits))
```

```
[129]: ['apple']
```

34 Reduce

```
[130]: # sum of all item
       import functools

       functools.reduce(lambda x,y:x+y,[1,2,3,4,5])
```

```
[130]: 15
```

```
[131]: # find min
       functools.reduce(lambda x,y:x if x>y else y,[23,11,45,10,1])
```

```
[131]: 45
```

35 Prepration of function

```
[132]: # 1 function of fav_books
       def favorite_book(title):
           return (f"One of my favorite books is {title}.")
       favorite_book('Alice in Woderland')
```

```
[132]: 'One of my favorite books is Alice in Woderland.'
```

```
[133]: # 2 using positional arguments

       def make_shirt(size, message):
           print(f"The shirt size is {size} and the message printed on it is:␣
        ↪{message}.")
```

```
# Calling the function using positional arguments
make_shirt('Medium', 'Python is fun!')

# Calling the function using keyword arguments
make_shirt(message='Hello, World!', size='Large')
```

```
The shirt size is Medium and the message printed on it is: Python is fun!.
The shirt size is Large and the message printed on it is: Hello, World!.
```

[134]:
```
# Write a function called describe_city().
def describe_city(city, country="Iceland"):
    print(f"{city} is in {country}")
    # Test the function with a city and a country
describe_city("Reykjavik") # use default country
describe_city("Oslo", "Norway") # use specified country
describe_city("New York", "USA")
```

```
Reykjavik is in Iceland
Oslo is in Norway
New York is in USA
```

[135]:
```
# 04.Write a function called city_country().
def city_country(city, country):
    return f"{city},is in  {country}"
# Test the function
print(city_country("Tokyo", "Japan"))
print(city_country("Berlin", "Germany"))
print(city_country("Santiago", "Chile"))
```

```
Tokyo,is in  Japan
Berlin,is in  Germany
Santiago,is in  Chile
```

[136]:
```
def make_car(manufacturer, model, **car_info):
    """Build a dictionary with information about a car."""
    car = {
        'manufacturer': manufacturer,
        'model': model,
    }
    car.update(car_info)
    return car

# Calling the function with additional information
car = make_car('subaru', 'outback', color='blue', tow_package=True)

# Printing the result
print(car)
```

```
{'manufacturer': 'subaru', 'model': 'outback', 'color': 'blue', 'tow_package':
True}
```

36 OOPS

[139]:
```
# making a oops
class Student:
    name = "Awais"
s1 = Student()
print(s1.name)
```

```
Awais
```

[140]:
```
# makin a oops
s2 = Student()
print(s2.name)
```

```
Awais
```

[141]:
```
# making car color
class Car:
    color = "blue"
    brand = "mercedes"
Car1 = Car()
print(Car1.color)
print(Car1.brand)
```

```
blue
mercedes
```

[142]:
```
class Student:
    name = "Awais"
    def __init__(self):
        print("adding a new student in Database")
        print(self)

s1 = Student()
print(s1)
```

```
adding a new student in Database
<__main__.Student object at 0x000001F89F8AE5A0>
<__main__.Student object at 0x000001F89F8AE5A0>
```

[143]:
```
# create a object
# object name = class name()
```

```
l = list()
l
```

[143]: []

[144]:
```
class Remote:
    def __init__(self):
        self.power = "Off"
        self.volume = 10
        self.channel = 1
        self.mute = "Unmuted"

    def power_on(self):
        if self.power == "Off":
            self.power = "On"
        else:
            self.power = "Off"
        print(f"The TV is now {self.power}.")

    def volume_change(self, new_volume):
        if self.power == "On":
            self.volume = new_volume
            print(f"The volume is changed to {self.volume}.")
        else:
            print("The TV is off. Turn it on to change the volume.")

    def change_channel(self, new_channel):
        if self.power == "On":
            self.channel = new_channel
            print(f"The channel is changed to {self.channel}.")
        else:
            print("The TV is off. Turn it on to change the channel.")

    def mute_toggle(self):
        if self.power == "On":
            if self.mute == "Unmuted":
                self.mute = "Muted"
            else:
                self.mute = "Unmuted"
            print(f"The TV is now {self.mute}.")
        else:
            print("The TV is off. Turn it on to toggle mute.")

def main():
    remote = Remote()
    while True:
        print("\n--- Remote Control Menu ---")
```

```python
        print("1. Power On/Off")
        print("2. Change Volume")
        print("3. Change Channel")
        print("4. Mute/Unmute")
        print("5. Exit")

        choice = input("Enter your choice (1-5): ")

        if choice == '1':
            remote.power_on()
        elif choice == '2':
            if remote.power == "On":
                try:
                    new_volume = int(input("Enter new volume (0-100): "))
                    if 0 <= new_volume <= 100:
                        remote.volume_change(new_volume)
                    else:
                        print("Please enter a volume between 0 and 100.")
                except ValueError:
                    print("Invalid input. Please enter a number.")
            else:
                print("The TV is off. Turn it on first.")
        elif choice == '3':
            if remote.power == "On":
                try:
                    new_channel = int(input("Enter new channel number: "))
                    remote.change_channel(new_channel)
                except ValueError:
                    print("Invalid input. Please enter a number.")
            else:
                print("The TV is off. Turn it on first.")
        elif choice == '4':
            remote.mute_toggle()
        elif choice == '5':
            print("Exiting the remote control program. Goodbye!")
            break
        else:
            print("Invalid choice. Please select a valid option.")

if __name__ == "__main__":
    main()
```

```
--- Remote Control Menu ---
1. Power On/Off
2. Change Volume
3. Change Channel
```

```
4. Mute/Unmute
5. Exit
The TV is now On.

--- Remote Control Menu ---
1. Power On/Off
2. Change Volume
3. Change Channel
4. Mute/Unmute
5. Exit
The TV is now Off.

--- Remote Control Menu ---
1. Power On/Off
2. Change Volume
3. Change Channel
4. Mute/Unmute
5. Exit
The TV is off. Turn it on first.

--- Remote Control Menu ---
1. Power On/Off
2. Change Volume
3. Change Channel
4. Mute/Unmute
5. Exit
The TV is off. Turn it on first.

--- Remote Control Menu ---
1. Power On/Off
2. Change Volume
3. Change Channel
4. Mute/Unmute
5. Exit
Exiting the remote control program. Goodbye!
```

[145]:
```python
import math as m

r = int(input("Enter the radius of the circle: "))

area = m.pi * r ** 2
perimeter = 2 * m.pi * r

class Circle:
    def __init__(self, radius):
        self.radius = radius
        self.area = m.pi * radius ** 2
        self.perimeter = 2 * m.pi * radius

p = Circle(r)
print("Area:", p.area)
print("Perimeter:", p.perimeter)
```

```
Area: 3421.194399759285
Perimeter: 207.34511513692635
```

[146]:
```
class instructor:
    pass
instructor_1=instructor()
print(type(instructor_1))
```

```
<class '__main__.instructor'>
```

[148]:
```
class instructor:
    pass
instructor_1=instructor()
instructor_1.name="Awais"
instructor_1.address="Islamabad"
print(instructor_1.name)
print(instructor_1.address)
instructor_2=instructor()
instructor_2.name="Natsha"
instructor_2.address="Karachi"
print(instructor_2.name)
print(instructor_2.address)
```

```
Awais
Islamabad
Natsha
Karachi
```

[150]:
```
class Student:
    def __init__(self,name,marks):
        self.name=name
        self.marks=marks
    def get_avg(self):
        sum = 0
        for val in self.marks:
            sum += val
        print("hi", self.name, "your avg score is", sum/3)

s = Student("Awais Manzoor", [90,96,90])
s.get_avg()
```

```
hi Awais Manzoor your avg score is 92.0
```

```
[152]: class Atm:
           def __init__(self):
               self.pin = ''
               self.balance = 0
               self.menue()

           def menue(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. pres 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 = input("Enter our balance")
               self.balance = user_balance
               self.menue()
               print("Pin created successfully ")
           def change_pin(self):
               old_pin = input("Enter your old pin")
               self.pin = old_pin
               if self.pin == old_pin:
                   new_pin = input("Enter your new pin")
                   self.pin = new_pin
               else:
                   print("Invalid pin")
               self.menue()
               print("Pin changed succeswsfully ")
           def check_balance(self):
               user_pin = input("Enter your pin")
               if user_pin == self.pin:
```

```
            print("Your balance is ", self.balance)
        else:
            print("Invalid pin")
        self.menue()
    def withdraw(self):
        user_pin = input("Enter you pin")
        if user_pin == self.pin:
            amount = input("Enter the amount ")
            if amount <= self.balance:
                self.balance -=amount
                print("Amount withdraw successfully, ", self.balance)
            else:
                print("Insufficient balance")
        else:
            print("Invalid pin")
        self.menue()
```

[153]:
```
atm1 = Atm()
```

```
Pin changed succeswsfully
Pin created successfully
```

[151]:
```
class Fraction:

  # parameterized constructor
  def __init__(self,x,y):
    self.num = x
    self.den = y

  def __str__(self):
    return '{}/{}'.format(self.num,self.den)

  def __add__(self,other):
    new_num = self.num*other.den + other.num*self.den
    new_den = self.den*other.den

    return '{}/{}'.format(new_num,new_den)

  def __sub__(self,other):
    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):
    new_num = self.num*other.num
    new_den = self.den*other.den

    return '{}/{}'.format(new_num,new_den)

  def __truediv__(self,other):
    new_num = self.num*other.den
    new_den = self.den*other.num

    return '{}/{}'.format(new_num,new_den)

  def convert_to_decimal(self):
    return self.num/self.den
```

[154]:
```
fr1 = Fraction(3,4)
fr2 = Fraction(1,2)
print(fr1 + fr2)
print(fr1 - fr2)
print(fr1 * fr2)
print(fr1 / fr2)
```

```
10/8
2/8
3/8
6/4
```

37 Encapsulation

[156]:
```
# Encapsulation is a concept in object-oriented programming (OOP) that binds␣
 ↪together the data and
# the methods that manipulate that data, and keeps both safe from outside␣
 ↪interference and misuse.
# instance var -> python tutor
class Person:

  def __init__(self,name_input,country_input):
    self.name = name_input # instance variableis a special variable  ka value␣
 ↪her obj kia lia different hota ha
    self.country = country_input

p1 = Person('Awais','Pakistan')
p2 = Person('Head','australia')
print(p1.name,p1.country)
print(p2.name,p2.country)
```

```
Awais Pakistan
Head australia
```

[157]:
```python
class Atm:

  # constructor(special function)->superpower ->
  def __init__(self):
    print(id(self))
    self.pin = ''
    self.__balance = 0
    #self.menu()

  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')
```

[158]:
```
obj = Atm()
```

```
2167340197568
```

[159]:
```
obj.get_balance()
```

[159]:
```
0
```

[160]:
```
obj.set_balance(1000)
```

[161]:
```
# list of objects
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

p1 = Person('Awais','male')
p2 = Person('Ali','male')
p3 = Person('ankita','female')

L = [p1,p2,p3]

for i in L:
  print(i.name,i.gender)
```

```
Awais male
Ali male
ankita female
```

[162]:
```
# dict of objects
# list of objects
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

p1 = Person('Awais','male')
p2 = Person('Zohaib','male')
p3 = Person('ankita','female')

d = {'p1':p1,'p2':p2,'p3':p3}

for i in d:
  print(d[i].gender)
```

```
male
male
female
```

[163]:
```
# Static Variables(Vs Instance variables)
""""Points to remember about static
Static attributes are created at class level.
Static attributes are accessed using ClassName.
Static attributes are object independent. We can access them without creating␣
 ↪instance (object) of the class in which they are defined.
The value stored in static attribute is shared between all instances(objects)␣
 ↪of the class in which the static attribute is defined.


[ ]
"""
```

```python
class Atm:

  __counter = 1

  # constructor(special function)->superpower ->
  def __init__(self):
    print(id(self))
    self.pin = ''
    self.__balance = 0
    self.cid = Atm.__counter
    Atm.__counter = Atm.__counter + 1
    #self.menu()

  # utility functions
  @staticmethod
  def get_counter():
    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')
```

[164]:
```
class Lion:
  __water_source="well in the circus"

  def __init__(self,name, gender):
      self.__name=name
```

```
        self.__gender=gender

    def drinks_water(self):
        print(self.__name,
        "drinks water from the",Lion.__water_source)

    @staticmethod
    def get_water_source():
        return Lion.__water_source

simba=Lion("Simba","Male")
simba.drinks_water()
print( "Water source of lions:",Lion.get_water_source())
```

```
Simba drinks water from the well in the circus
Water source of lions: well in the circus
```

Aggregation(Has-A relationship)

[165]:
```
"""Class Relationships
Aggregation
Inheritance"""
```

[165]:
```
'Class Relationships\nAggregation\nInheritance'
```

[166]:
```
# example
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()
# method example
# what about private attribute
```

```
gurgaon 122011 haryana
mumbai 111111 maharastra
```

38 Inheritance

[168]:
```
"""nheritance in summary
A class can inherit from another class.

Inheritance improves code reuse

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"""



#Example

# parent
class User:

  def __init__(self):
    self.name = 'Awais'
    self.gender = 'male'

  def login(self):
    print('login')

# child
class Student(User):

  def __init__(self):
    self.rollno = 100

  def enroll(self):
    print('enroll into the course')
    # creating an object of the child class


u = User()
print(u.name)
s = Student()
s.login()
s.enroll()
```

```
Awais
login
enroll into the course
```

[169]:
```
# 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()
s.price
s.brand
```

```
Inside phone constructor
```

```
Buying a phone
```

[169]:
```
'Apple'
```

[170]:
```python
s.camera
```

[170]:
```
13
```

[171]:
```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)
s.show()
```

```
Inside phone constructor
20000
```

[172]:
```python
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
```

39 Method Overriding

[173]:
```python
# 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()
```

```
Inside phone constructor
Buying a smartphone
```

40 Super Keyword

[174]:
```python
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.buy()
```

```
Inside phone constructor
Buying a smartphone
Buying a phone
```

[175]:
```
# using super outside the class give error
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.buy
```

```
Inside phone constructor
```

[175]:
```
<bound method SmartPhone.buy of <__main__.SmartPhone object at
0x000001F89FA17E00>>
```

[176]:
```
# 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)
        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
```

[177]:
```
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
```

41 Types of Inheritance

[178]:
```
"""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
```

[179]:
```
# 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
```

[180]:
```
# 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
```

[181]:
```
# 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()
s.review()
```

```
Inside phone constructor
Buying a phone
Customer review
```

[182]:
```
# 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
class SmartPhone(Phone,Product):
    pass

s=SmartPhone(20000, "Apple", 12)

s.buy()
```

```
Inside phone constructor
Buying a phone
```

[183]:
```
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
```

42 Polymorphism

i. Method Overriding
ii. Method Overloading
iii. Operator Overloading

[184]:
```
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
```

[185]:
```
'hello' + 'world'
```

[185]:
```
'helloworld'
```

[186]:
```
4 + 5
```

[186]:
```
9
```

[187]:
```
[1,2,3] + [4,5]
```

[187]:
```
[1, 2, 3, 4, 5]
```

43 Abstraction

[188]:
```
from abc import ABC,abstractmethod
class BankApp(ABC):

  def database(self):
    print('connected to database')
  @abstractmethod
  def security(self):
    pass

  @abstractmethod
  def display(self):
    pass
```

[189]:
```
class MobileApp(BankApp):

  def mobile_login(self):
    print('login into mobile')

  def security(self):
    print('mobile security')

  def display(self):
    print('display')
```

[190]:
```
mob = MobileApp()
```

[191]:
```
mob.security()
```

```
mobile security
```

44 Mini projects using function and oops

45 Full Menue of Restaurant

[43]:
```
# Define the menue of Restaurant
menu = {
    'pizza': 60,
    'Pasta': 40,
    'Burger': 60,
    'salad': 70,
    'coffee': 80,
}

print("Welcome to Python Restaurant")
print("pizza: 60 Rs\nPasta: 40 Rs\nBurger: 60 Rs\nsalad: 70 Rs\ncoffee: 80 Rs")

order_total = 0

while True:
    item = input("Enter the item you want to order: ")
    if item in menu:
        order_total += menu[item]
        print(f"Your item {item} has been added to your order")
    else:
        print("Sorry, we don't have that item on the menu")

    another_order = input("Do you want to add another item? (yes/no): ")
    if another_order.lower() != 'yes':
        break

print(f"The total amount to pay is {order_total}Rs")
```

```
Welcome to Python Restaurant
pizza: 60 Rs
Pasta: 40 Rs
Burger: 60 Rs
salad: 70 Rs
coffee: 80 Rs
Your item pizza has been added to your order
The total amount to pay is 60Rs
```

[192]:
```
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        """Calculate and return the area of the circle."""
        return math.pi * (self.radius ** 2)

    def calculate_perimeter(self):
        """Calculate and return the perimeter (circumference) of the circle."""
        return 2 * math.pi * self.radius

# Create an instance of the Circle class with a specific radius
# for user input
radius = float(input("Enter the radius of the circle: "))
circle = Circle(radius)
print(f"The area of the circle is: {circle.calculate_area()}")
print(f"The perimeter of the circle is: {circle.calculate_perimeter()}")
```

```
The area of the circle is: 13684.77759903714
The perimeter of the circle is: 414.6902302738527
```

[194]:
```
class Instructor:
    pass
instructor_1=Instructor()
```

```
instructor_1.name="Fawad"
instructor_1.address= "Sharjah"
print(instructor_1.name)
print(instructor_1.address)

instructor_2 = Instructor()
instructor_2.name ='Awais'
instructor_2.address = 'Multlan'
print(instructor_2.name)
print(instructor_2.address)
```

```
Fawad
Sharjah
Awais
Multlan
```

[195]:
```
class Tree:
    def __init__(self,height):
        self.__height= height

    def get_height(self):
        return self.__height

    def set_height(self,new_height):
        if not isinstance(new_height,int):
            raise TypeError("Tree must be an integer")
        if 0 < new_height <= 40:
            self.__height = new_height
        else:
            raise ValueError("Invalid height for a pine tree")


pine = Tree(40.9)
pine.get_height()
```

[195]:
```
40.9
```

[196]:
```
class Human: # parent class
    def __init__(self): # constructor  -> super poweer
        self.num_eyes = 2  # instance variables
        self.num_nose = 1

    def eat(self):      # methods and self is obj
        print("I can eat")

    def work(self):
        print("I can teach")
```

```
class Male(Human): # base class

    def __init__(self,name):
        super().__init__() # super keyword to access the parent clas
        self.name = name

    def cricket(self):
        print("Plays cricket")

    def work(self):
        super().work() # to access the parent class
        print("can do research also with coding")

male_1=Male("Abraham")  # Male_1 is a ovject
male_1.cricket()
male_1.work()
print(male_1.num_eyes)
print(male_1.num_nose)
```

```
Plays cricket
I can teach
can do research also with coding
2
1
```

[198]:
```
def check_holiday(month, day):
    holidays = {
        (2, 5): "Pakistan Day",
        (3, 23): "Kashmir Day",
        (5, 1): "Labour Day",
        (8, 14): "Independence Day",
        (11, 9): "Allama Iqbal Day",
        (12, 25): "Quaid-e-Azam Day/Christmas"
    }
    return holidays.get((month, day), "No holiday on this date.")

def is_valid_password(password):
    if len(password) < 8:
        return "Password must be at least 8 characters long."
    if not any(c.islower() for c in password):
        return "Password must contain at least one lowercase letter."
    return "Password is valid."

# Main program
if __name__ == "__main__":
    # Read month and day
    month = int(input("Enter month (1-12): "))
    day = int(input("Enter day (1-31): "))

    # Check for holiday
    holiday_message = check_holiday(month, day)
    print(holiday_message)

    # Read password
    password = input("Enter a password: ")
    password_message = is_valid_password(password)
    print(password_message)
```

```
No holiday on this date.
Password must be at least 8 characters long.
```

[200]:
```
def calculate_total_cost(meal_cost, tip_percentage):
    return meal_cost + (meal_cost * tip_percentage / 100)

if __name__ == "__main__":
    meal_cost = float(input("Enter the cost of the meal: "))
    tip_percentage = float(input("Enter the tip percentage (e.g., 15 for 15%):␣
 ↪"))

    total_cost = calculate_total_cost(meal_cost, tip_percentage)
    print(f"The total cost of the meal including tip is: ${total_cost}")
```

```
The total cost of the meal including tip is: $133.0
```

[201]:
```
def determine_shape(sides):
    shapes = {
        3: "Triangle",
        4: "Square",
        5: "Pentagon",
        6: "Hexagon",
        7: "Heptagon",
        8: "Octagon",
        9: "Nonagon",
        10: "Decagon"
    }
    return shapes.get(sides, "Invalid number of sides")

if __name__ == "__main__":
    sides = int(input("Enter the number of sides: "))
    shape = determine_shape(sides)
    print(f"The shape with {sides} sides is: {shape}")
```

```
The shape with 4 sides is: Square
```

[202]:
```
def assign_class(score):
    if score >= 90:
        return "A Class"
    elif score >= 80:
        return "B Class"
    elif score >= 70:
        return "C Class"
    elif score >= 60:
        return "D Class"
    else:
        return "F Class"

if __name__ == "__main__":
    score = float(input("Enter the student's score: "))
    class_assigned = assign_class(score)
    print(f"The student has been assigned to: {class_assigned}")
```

```
The student has been assigned to: F Class
```

[203]:
```
class Employee:
    def __init__(self, name, position, salary):
        self.name = name
        self.position = position
        self.salary = salary

    def display_info(self):
        print(f"Employee Name: {self.name}")
        print(f"Position: {self.position}")
        print(f"Salary: ${self.salary}")

if __name__ == "__main__":
    employee_name = input("Enter the employee's name: ")
    employee_position = input("Enter the employee's position: ")
    employee_salary = float(input("Enter the employee's salary: "))

    employee = Employee(employee_name, employee_position, employee_salary)
    employee.display_info()
```

```
Employee Name: Awais
Position: HR
Salary: $2000.0
```

46 Advanced Projects

47 Hospital Management System

[205]:
```
class Patient:
    def __init__(self, patient_id, name, age, contact):
        self.patient_id = patient_id
        self.name = name
        self.age = age
        self.contact = contact

    def __str__(self):
        return f"Patient[ID: {self.patient_id}, Name: {self.name}, Age: {self.
 ↪age}, Contact: {self.contact}]"


class Doctor:
    def __init__(self, doctor_id, name, specialty, contact):
        self.doctor_id = doctor_id
        self.name = name
        self.specialty = specialty
        self.contact = contact

    def __str__(self):
        return f"Doctor[ID: {self.doctor_id}, Name: {self.name}, Specialty:␣
 ↪{self.specialty}, Contact: {self.contact}]"


class Appointment:
    def __init__(self, appointment_id, patient, doctor, date, time):
        self.appointment_id = appointment_id
        self.patient = patient
        self.doctor = doctor
        self.date = date
        self.time = time

    def __str__(self):
        return f"Appointment[ID: {self.appointment_id}, Patient: {self.patient.
 ↪name}, Doctor: {self.doctor.name}, Date: {self.date}, Time: {self.time}]"


class InventoryItem:
    def __init__(self, item_id, name, quantity):
        self.item_id = item_id
        self.name = name
        self.quantity = quantity
    def __str__(self):
        return f"InventoryItem[ID: {self.item_id}, Name: {self.name}, Quantity:␣
↪{self.quantity}]"


class HospitalManagementSystem:
    def __init__(self):
        self.patients = {}
        self.doctors = {}
        self.appointments = {}
        self.inventory = {}

    def add_patient(self, patient_id, name, age, contact):
        if patient_id in self.patients:
            print("Patient already exists.")
        else:
            self.patients[patient_id] = Patient(patient_id, name, age, contact)
            print("Patient added successfully.")

    def add_doctor(self, doctor_id, name, specialty, contact):
        if doctor_id in self.doctors:
            print("Doctor already exists.")
        else:
            self.doctors[doctor_id] = Doctor(doctor_id, name, specialty,␣
↪contact)
            print("Doctor added successfully.")

    def schedule_appointment(self, appointment_id, patient_id, doctor_id, date,␣
↪time):
        if appointment_id in self.appointments:
            print("Appointment already exists.")
        elif patient_id not in self.patients:
            print("Patient not found.")
        elif doctor_id not in self.doctors:
            print("Doctor not found.")
        else:
            patient = self.patients[patient_id]
            doctor = self.doctors[doctor_id]
            self.appointments[appointment_id] = Appointment(appointment_id,␣
↪patient, doctor, date, time)
            print("Appointment scheduled successfully.")

    def add_inventory_item(self, item_id, name, quantity):
        if item_id in self.inventory:
            self.inventory[item_id].quantity += quantity
            print("Inventory updated successfully.")
        else:
            self.inventory[item_id] = InventoryItem(item_id, name, quantity)
            print("Inventory item added successfully.")

    def display_patients(self):
        for patient in self.patients.values():
            print(patient)

    def display_doctors(self):
        for doctor in self.doctors.values():
            print(doctor)

    def display_appointments(self):
        for appointment in self.appointments.values():
            print(appointment)

    def display_inventory(self):
        for item in self.inventory.values():
            print(item)


# Example Usage
if __name__ == "__main__":
    hms = HospitalManagementSystem()
    while True:
        print("\n--- Hospital Management System ---")
        print("1. Add Patient")
        print("2. Add Doctor")
        print("3. Schedule Appointment")
        print("4. Add Inventory Item")
        print("5. Display Patients")
        print("6. Display Doctors")
        print("7. Display Appointments")
        print("8. Display Inventory")
        print("9. Exit")
        choice = input("Enter your choice: ").strip()

        if choice == '1':
            patient_id = int(input("Enter Patient ID: "))
            name = input("Enter Patient Name: ").strip()
            age = int(input("Enter Patient Age: "))
            contact = input("Enter Patient Contact: ").strip()
            hms.add_patient(patient_id, name, age, contact)
        elif choice == '2':
            doctor_id = int(input("Enter Doctor ID: "))
            name = input("Enter Doctor Name: ").strip()
            specialty = input("Enter Doctor Specialty: ").strip()
            contact = input("Enter Doctor Contact: ").strip()
```

```python
            hms.add_doctor(doctor_id, name, specialty, contact)
        elif choice == '3':
            appointment_id = int(input("Enter Appointment ID: "))
            patient_id = int(input("Enter Patient ID: "))
            doctor_id = int(input("Enter Doctor ID: "))
            date = input("Enter Appointment Date (YYYY-MM-DD): ").strip()
            time = input("Enter Appointment Time (HH:MM AM/PM): ").strip()
            hms.schedule_appointment(appointment_id, patient_id, doctor_id,␣
↪date, time)
        elif choice == '4':
            item_id = int(input("Enter Inventory Item ID: "))
            name = input("Enter Item Name: ").strip()
            quantity = int(input("Enter Quantity: "))
            hms.add_inventory_item(item_id, name, quantity)
        elif choice == '5':
            print("\nPatients:")
            hms.display_patients()
        elif choice == '6':
            print("\nDoctors:")
            hms.display_doctors()
        elif choice == '7':
            print("\nAppointments:")
            hms.display_appointments()
        elif choice == '8':
            print("\nInventory:")
            hms.display_inventory()
        elif choice == '9':
            print("Exiting the system. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
```

```
--- Hospital Management System ---
1. Add Patient
2. Add Doctor
3. Schedule Appointment
4. Add Inventory Item
5. Display Patients
6. Display Doctors
7. Display Appointments
8. Display Inventory
9. Exit
Patient added successfully.

--- Hospital Management System ---
1. Add Patient
```

```
2. Add Doctor
3. Schedule Appointment
4. Add Inventory Item
5. Display Patients
6. Display Doctors
7. Display Appointments
8. Display Inventory
9. Exit
Doctor added successfully.

--- Hospital Management System ---
1. Add Patient
2. Add Doctor
3. Schedule Appointment
4. Add Inventory Item
5. Display Patients
6. Display Doctors
7. Display Appointments
8. Display Inventory
9. Exit
Appointment scheduled successfully.

--- Hospital Management System ---
1. Add Patient
2. Add Doctor
3. Schedule Appointment
4. Add Inventory Item
5. Display Patients
6. Display Doctors
7. Display Appointments
8. Display Inventory
9. Exit

Inventory:

--- Hospital Management System ---
1. Add Patient
2. Add Doctor
3. Schedule Appointment
4. Add Inventory Item
5. Display Patients
6. Display Doctors
7. Display Appointments
8. Display Inventory
9. Exit
Exiting the system. Goodbye!
```

48 Student Management System project

[204]:
```python
# Student Management System
"""A system that manages student information, including name, roll number,␣
↪grades, and class assignments. The system should allow adding, updating, and␣
↪viewing student details."""
class Student:
    def __init__(self, roll_number, name, grades):
        self.roll_number = roll_number
        self.name = name
        self.grades = grades

    def display_info(self):
        return f"Roll Number: {self.roll_number}, Name: {self.name}, Grades:␣
↪{self.grades}"

    def update_grades(self, new_grades):
        self.grades = new_grades
        print(f"Updated grades for {self.name}: {self.grades}")

def manage_students():
    students = []
    while True:
        choice = input("Choose an option: (1) Add Student (2) View Students (3)␣
↪Update Grades (4) Exit: ")
        if choice == '1':
            roll_number = input("Enter roll number: ")
            name = input("Enter name: ")
            grades = input("Enter grades: ")
            student = Student(roll_number, name, grades)
            students.append(student)
        elif choice == '2':
            for student in students:
                print(student.display_info())
        elif choice == '3':
            roll_number = input("Enter roll number to update grades: ")
            for student in students:
                if student.roll_number == roll_number:
                    new_grades = input(f"Enter new grades for {student.name}: ")
                    student.update_grades(new_grades)
        elif choice == '4':
            break
        else:
            print("Invalid option. Please try again.")

# Test the Student Management System
if __name__ == "__main__":
    manage_students()
```

```
Roll Number: 111, Name: Awais, Grades: 4
Updated grades for Awais: 5
```

49 Library Management System

[206]:
```
""" Library Management System
This system helps in managing a library by keeping track of books, issuing␣
 ↪them, and allowing users to return books."""
class Book:
    def __init__(self, title, author, book_id):
        self.title = title
        self.author = author
        self.book_id = book_id
        self.is_issued = False # using Abstraction method

    def issue_book(self):
        if not self.is_issued:
            self.is_issued = True
            print(f"Book {self.title} issued.")
        else:
            print(f"Book {self.title} is already issued.")

    def return_book(self):
        if self.is_issued:
            self.is_issued = False
            print(f"Book {self.title} returned.")
        else:
            print(f"Book {self.title} was not issued.")

def manage_library():
    books = []
    books.append(Book("Harry Potter", "J.K. Rowling", 1))
    books.append(Book("To Kill a Mockingbird", "Harper Lee", 2))
    books.append(Book("1984", "George Orwell", 3))

    while True:
        choice = input("Choose an option: (1) View Books (2) Issue Book (3)␣
 ↪Return Book (4) Exit: ")
        if choice == '1':
            for book in books:
                status = "Issued" if book.is_issued else "Available"
                print(f"{book.title} by {book.author} ({status})")
        elif choice == '2':
            book_id = int(input("Enter book ID to issue: "))
            for book in books:
                if book.book_id == book_id:
                    book.issue_book()
                    break
            else:
                print("Book not found.")
        elif choice == '3':
            book_id = int(input("Enter book ID to return: "))
            for book in books:
                if book.book_id == book_id:
                    book.return_book()
                    break
            else:
                print("Book not found.")
        elif choice == '4':
            break
        else:
            print("Invalid option. Please try again.")

# Test the Library Management System
if __name__ == "__main__":
    manage_library()
```

```
Harry Potter by J.K. Rowling (Available)
To Kill a Mockingbird by Harper Lee (Available)
1984 by George Orwell (Available)
Harry Potter by J.K. Rowling (Available)
To Kill a Mockingbird by Harper Lee (Available)
1984 by George Orwell (Available)
Book 1984 issued.
Book 1984 returned.
```

50 Simple Calculator

[208]:
```
# Define the Calculator class
class Calculator:
    def __init__(self):
        pass

    # Method for addition
    def add(self, num1, num2):
        return num1 + num2

    # Method for subtraction
    def subtract(self, num1, num2):
        return num1 - num2
```

```python
    # Method for multiplication
    def multiply(self, num1, num2):
        return num1 * num2

    # Method for division
    def divide(self, num1, num2):
        if num2 == 0:
            return "Error: Division by zero"
        else:
            return num1 / num2


# Main code to use the Calculator class
def main():
    calc = Calculator()

    # Get user input
    print("Simple Calculator:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")



    # Choose operation
    choice = int(input("Enter the operation number (1/2/3/4): "))

    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    # Perform the chosen operation
    if choice == 1:
        result = calc.add(num1, num2)
        print(f"The result of addition is: {result}")
    elif choice == 2:
        result = calc.subtract(num1, num2)
        print(f"The result of subtraction is: {result}")
    elif choice == 3:
        result = calc.multiply(num1, num2)
        print(f"The result of multiplication is: {result}")
    elif choice == 4:
        result = calc.divide(num1, num2)
        print(f"The result of division is: {result}")
    else:
        print("Invalid choice")
```

```
# Run the program
if __name__ == "__main__":
    main()
```

```
Simple Calculator:
1. Add
2. Subtract
3. Multiply
4. Divide
The result of addition is: 55.0
```

51 Simple Bank Account System

[210]:
```
class BankAccount:
    def __init__(self, holder_name, balance=0):
        self.holder_name = holder_name
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited {amount}. New balance: {self.balance}")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")
        else:
            print("Insufficient funds.")

    def check_balance(self):
        print(f"Current balance: {self.balance}")

def bank_system():
    name = input("Enter your name: ")
    account = BankAccount(name)
    while True:
        choice = input("Choose an option: (1) Deposit (2) Withdraw (3) Check␣
 ↪Balance (4) Exit: ")
        if choice == '1':
            amount = float(input("Enter amount to deposit: "))
            account.deposit(amount)
        elif choice == '2':
            amount = float(input("Enter amount to withdraw: "))
            account.withdraw(amount)
        elif choice == '3':
            account.check_balance()
        elif choice == '4':
            break
        else:
            print("Invalid option. Please try again.")

# Test the Simple Bank Account System
if __name__ == "__main__":
    bank_system()
```

```
Deposited 10000.0. New balance: 10000.0
Withdrew 2000.0. New balance: 8000.0
Current balance: 8000.0
```

52 Currency Converter

[211]:
```
def convert_currency(amount, from_currency, to_currency):
    rates = {
        'USD': {'EUR': 0.85, 'INR': 74.93, 'GBP': 0.74},
        'EUR': {'USD': 1.18, 'INR': 88.04, 'GBP': 0.87},
        'INR': {'USD': 0.013, 'EUR': 0.011, 'GBP': 0.010},
        'GBP': {'USD': 1.35, 'EUR': 1.15, 'INR': 100.55}
    }
    if from_currency == to_currency:
        return amount
    try:
        return amount * rates[from_currency][to_currency]
    except KeyError:
        return "Conversion rate not available."

def currency_converter():
    amount = float(input("Enter amount: "))
    from_currency = input("Enter the currency to convert from (USD, EUR, INR,␣
 ↪GBP): ").upper()
    to_currency = input("Enter the currency to convert to (USD, EUR, INR, GBP):␣
 ↪").upper()
    converted_amount = convert_currency(amount, from_currency, to_currency)
    print(f"{amount} {from_currency} = {converted_amount} {to_currency}")

# Test the Currency Converter
if __name__ == "__main__":
    currency_converter()
```

```
2000.0 USD = 149860.0 INR
```

53 10.Email Slicer

[213]:
```
class Email:
    def __init__(self, sender, recipient, subject, body):
        self.sender = sender
        self.recipient = recipient
        self.subject = subject
        self.body = body

    def __str__(self):
        return f"From: {self.sender}\nTo: {self.recipient}\nSubject: {self.
 ↪subject}\nBody: {self.body}\n"

class EmailClient:
    def __init__(self):
        self.emails = []

    def send_email(self, email):
        self.emails.append(email)
        print("Email sent successfully!")

    def get_emails(self):
        if not self.emails:
            print("No emails found.")
        else:
            for idx, email in enumerate(self.emails, 1):
                print(f"Email {idx}:\n{email}")


def email_client_app():
    client = EmailClient()

    while True:
        print("\nEmail Client Menu:")
        print("1. Send an email")
        print("2. View sent emails")
        print("3. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            sender = input("Enter sender's email: ")
            recipient = input("Enter recipient's email: ")
            subject = input("Enter email subject: ")
            body = input("Enter email body: ")

            email = Email(sender, recipient, subject, body)
            client.send_email(email)

        elif choice == "2":
            client.get_emails()

        elif choice == "3":
            print("Exiting the application.")
            break

        else:
            print("Invalid choice. Please try again.")

# Run the application
if __name__ == "__main__":
    email_client_app()
```

```
Email Client Menu:
1. Send an email
2. View sent emails
3. Exit
Email sent successfully!

Email Client Menu:
1. Send an email
2. View sent emails
3. Exit
Exiting the application.
```

[214]:
```
pip install emoji
```

```
Requirement already satisfied: emoji in
c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (2.14.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[215]:
```
import emoji
print(emoji.emojize('Python is :grinning_face:'))
```

```
Python is
```

54 File Handeling

[216]:
```
"""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')
f.write('Hello world')
f.close()
```

[217]:
```
# write multiline strings
f = open('sample1.txt','w')
f.write('hello world')
f.write('\nhow are you?')
f.close()
```

[13]:
```
# case 2 - if the file is already present
f = open('sample.txt','w')
f.write('Awais Manzoor')
f.close()
```

[14]:
```
# reading from files
# -> using read()
f = open("sample.txt", "r")
s = f.read()
print(s)
f.close()
```

```
Awais Manzoor
```

[15]:
```
# reading upto n chars
f = open("sample.txt", "r")
s = f.read(10)
print(s)
f.close()
```

```
Awais Manz
```

[16]:
```python
# readline() -> to read line by line
f = open("sample.txt", "r")
print(f.readline(),end='')
print(f.readline(),end='')
f.close()
```

```
Awais Manzoor
```

[17]:
```python
# reading entire using readline
f = open("sample.txt", "r")

while True:

  data = f.readline()

  if data == '':
    break
  else:
    print(data,end='')

f.close()
```

```
Awais Manzoor
```

[20]:
```python
"""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
with open('sample.txt','w') as f:
  f.write('Awais')
```

[22]:
```python
# try f.read() now
with open('sample.txt','r') as f:
  print(f.readline())
```

```
Awais
```

[23]:
```python
# moving within a file -> 10 char then 10 char
with open('sample.txt','r') as f:
  print(f.read(10))
  print(f.read(10))
  print(f.read(10))
  print(f.read(10))
```

```
Awais
```

```
[24]: # benefit? -> to load a big file in memory
      big_L = ['hello world ' for i in range(1000)]

      with open('big.txt','w') as f:
        f.writelines(big_L)
```

```
[25]: with open('big.txt','r') as f:

        chunk_size = 10

        while len(f.read(chunk_size)) > 0:
          print(f.read(chunk_size),end='***')
          f.read(chunk_size)
```

```
d hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***d hello wo***o world he***d
hello wo***o world he***d hello wo***o world he***
```

[26]:
```python
# seek and tell function
with open('sample.txt','r') as f:
  f.seek(15)
  print(f.read(10))
  print(f.tell())

  print(f.read(10))
  print(f.tell())
```

```
15
```

```
15
```

[27]:
```
# seek during write
with open('sample.txt','w') as f:
  f.write('Hello')
  f.seek(0)
  f.write('Xa')
```

[29]:
```
with open('sample.txt','w') as f:
  f.write('5')
```

55 json Serialization and Deserialization

[30]:
```
#Serialization - process of converting python data types to JSON format
#Deserialization - process of converting JSON to python data types
# serialization using json module
# list
import json

L = [1,2,3,4]

with open('demo.json','w') as f:
  json.dump(L,f)
```

[31]:
```
# dict
d = {
    'name':'nitish',
     'age':33,
     'gender':'male'
}

with open('demo.json','w') as f:
  json.dump(d,f,indent=4)
```

[32]:
```
# deserialization
import json

with open('demo.json','r') as f:
  d = json.load(f)
  print(d)
  print(type(d))
```

```
{'name': 'nitish', 'age': 33, 'gender': 'male'}
<class 'dict'>
```

[33]:
```
class Person:

  def __init__(self,fname,lname,age,gender):
    self.fname = fname
    self.lname = lname
    self.age = age
    self.gender = gender

# format to printed in
# -> Nitish Singh age -> 33 gender -> male
person = Person('Nitish','Singh',33,'male')
```

[34]:
```
# 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)
```

[35]:
```
# As a dict
import json

def show_object(person):
  if isinstance(person,Person):
    return {'name':person.fname + ' ' + person.lname,'age':person.age,'gender':
↪person.gender}

with open('demo.json','w') as f:
  json.dump(person,f,default=show_object,indent=4)
```

[36]:
```
# deserializing
import json

with open('demo.json','r') as f:
  d = json.load(f)
  print(d)
  print(type(d))
```

```
{'name': 'Nitish Singh', 'age': 33, 'gender': 'male'}
<class 'dict'>
```

56 Pickling

[37]:
```
#Pickling` is the process whereby a Python object hierarchy is converted into a␣
↪byte stream, and `unpickling` is the inverse operation, whereby a byte␣
↪stream (from a binary file or bytes-like object) is converted back into an␣
↪object hierarchy.
class Person:

  def __init__(self,name,age):
    self.name = name
    self.age = age

  def display_info(self):
    print('Hi my name is',self.name,'and I am ',self.age,'years old')
```

[39]:
```
p = Person('Awais',33)
```

[40]:
```
# pickle dump
import pickle
with open('person.pkl','wb') as f:
  pickle.dump(p,f)
```

[41]:
```
# pickle load
import pickle
with open('person.pkl','rb') as f:
  p = pickle.load(f)

p.display_info()
```

```
Hi my name is Awais and I am  33 years old
```

[42]:
```
#Pickle Vs Json
#Pickle lets the user to store data in binary format.
# JSON lets the user store data in a human-readable text format.
```

57 Awais Manzoor

57.1 Data Analyst