```python
#-- Comments start with a hash sign     ## TODAY= 3/24/2025
  #-- they can be inserted in same line as code but after the code
        #-- if inserted before code, it blocks the code from execution
  #Go to PyCharm Themes for HIGH CONTRAST MODE !!!!!
""""
Triple quotes is another way to create a comment paragraph
See https://www.w3schools.com/python/python_comments.asp
This main.py file is based on NANA's beginner course
"""
from xml.sax import make_parser

#  DATA TYPES
text_1 = "This is a simple string"
tex_2: str = 'This is an annotated string in single quotes'

numb_1 = 15         #this a variable being assigned as an integer number
numb_2: int = 25    #this is an annotated (typed) integer object
numb_3 = 15.3       # this assigns a float value to numb_3
numb_4: float = 15.4
numb_0 = None       # variable is created but its value is empty, its type is 'None'

list_1 = [ 'Alice', 'Bob', 'Carrol', numb_1, numb_3] #Square brackets
```

```python
set_1 = (1, 3, 5, 7, 9, 'Bob', 'more__')  # Round brackets (parens)
tuple_2: tuple = ('immutable', 'set elements')
key_value_pairs = {'USER_name': 'Bob', 'age': 20, 'single': True, 'sex': 'male'}
# Note: keys should be 'strings' !!!!
print (key_value_pairs, '\n')            # key is the input inot the dictionary and value is the definitions
                                         # the distionary returns for the inputted key

String_1 = '\n this is a bunch of lower case "letters" in single quotes'
String_2 = "\n" + "*=" * 20  #<-- combo of return plus repeat pattern
print(String_2)
print(String_1)   #<-- Printing a string object
print(String_2)

print (String_1.upper())  #each string is an object having methods
String_4 = 'fix for input problem?'
String_3 = input('Type in string number three below \n')
print((String_3 +" ") * 3)
print(type(String_3))

x = 'hello'
y = 'hello'
print (x == y, '\n')  #<-- double equal is equality test, opp is !=

# Tricks
number_seq: list[int] = [1, 2, 3, 4, 5]  #<-- a sequential list object
greet: str = 'Hello Planet Earth'  #<-- a sample string object
print(number_seq[::-1], 'now reversed')
print(greet[::-1], 'now reversed \n')          ##<- Trick #1 reverses order, note sq brackets

#Trick #2 test for odd versus even using modulus operator: %
sample: int = 12
Result_mod: str = 'Even' if sample % 2 == 0 else 'Odd'
print (Result_mod)
Result_mod: str = 'Divisible by three' if sample % 3 == 0 else 'Not divisible'
print (Result_mod)

# Trick #3
List_o_emails: list[str] = ['Abe@gmail.com', 'Bob@gmail.com', 'Carl@gmail.com']
print(f'Emails: {List_o_emails}')       # Curly braces {} mean: substitute in the value of this variable
print(f'Emails: {", ".join(List_o_emails)}')
# ^^^ above didn't work until join was made a method OF the string !!!

#Trick #4
Letters = 'ABCDEF'
for i, letter in enumerate(Letters, start=1):  #forces A to be item #1 not 0
```

```
    print(f'{i}: {letter}')
    print(Letters[1], 'index was of value one' )                              # the first element A is in index position 0 !!!
# https://www.youtube.com/watch?v=YzUBJfGAFyA

condition_input = 'hello'
if condition_input == 'hello' : #<-- double equal tests for sameness
    print('Yes, hello== hello') #NOTE colon (:) at end of if, elif and else
elif condition_input == 'bye':
    print('Goodbye then')
else:
    print('Are you coming or going?')
print('done \n \n')

from datetime import datetime
print('Date and time now is', datetime.now(), '\n')

def What_time_izit()  -> None:      # Result is of type NONE, note colon after
    print('Date and time now is', datetime.now(), '\n')     #note indentation

What_time_izit()
print('wasting some time here' *10, '\n')
What_time_izit()

def mult(a: float, b: float) -> float:   # the return type is Float as opposed to None
    return a*b                           # this func returns the value of expression a
```

```python
 times b
print(mult(1.5, 2.104))

# OBJECT ORIENTED CLASS ATTRIBUTES AND METHODS()

class Automobile:                       # creating a class of objects that are each an Automobile
    def __init__(self, Make: str, Model: str, Color, HorsePower: float) -> None: # initializing the attributes
          self.Make = Make                # of each Automobile (each 'self', each instance)
          self.Model = Model; self.Color = Color; self.HorsePower = HorsePower  # HP problem semicolons
    def drive(self) -> None:       #This is a METHOD() that is executable by an Automobile
        print(f'The {self.Make} {self.Model} is now in drive mode')

    def __str__(self) -> str:     #This is a DUNDER() method that returns a string
        return f'{self.Make} {self.Model} has {self.HorsePower} Power'


Adam_sVehicle = Automobile("Volvo", "Hybrid", "Red", 350.1)
Bill_sVehicle = Automobile("BMW", "Diesel", "Metalic Blue", 380.2)

```

```python
print(f'Adam"s car color is ', Adam_sVehicle.Color)

print(f'Bill"s car power is {Bill_sVehicle.HorsePower}')
Adam_sVehicle.drive()
Bill_sVehicle.drive()
print(Bill_sVehicle)     #Dunder method returns converted str to print output

print('DONE')


```