

Activity 1

Numeric numbers:

```
[1]: #Numeric Numbers
x=10
y=-30
z=0

print (x)
print (y)
print (z)

10
-30
0
```

Floating point:

```
-30
0

[3]: #Floating point

a = 3.14
b = -2.719
c = 0.1

print (a)
print (b)
print (c)

3.14
-2.719
0.1
```

Complex numbers:

```
#Complex number examples
x = 3 + 4j
y = 1.5 - 2.5j

print(x) # Output: (3+4j)
print(y) # Output: (1.5 -2.5j)

(3+4j)
(1.5-2.5j)
```

Arithmetic operations:

```
#Arithmetic operations

a=10
b=3

#Addition
print(a+b) # Output: 13
#Subtraction
print(a-b) # Output: 7
#Multiplication
print(a*b) # Output: 30
#Division
print(a/b) # Output: 3.33333
#Floor Division (returns the integer part of the division)
print(a//b) # Output: 3
#Modulus
print(a%b) # Output:1
#Exponentiation
print(a**b) # Output:1000

13
7
30
3.3333333333333335
3
1
1000
```

List:

```
# List (mutable)
my_list = [1, 2, 3, 'apple', 'banana']
print("List:", my_list, type(my_list))

List: [1, 2, 3, 'apple', 'banana'] <class 'list'>
```

Tuple:

```
# Tuple (immutable)
my_tuple = (1, 2, 'apple', 'banana')
print("Tuple:", my_tuple, type(my_tuple))

Tuple: (1, 2, 'apple', 'banana') <class 'tuple'>
```

String:

```
# String (immutable)
my_string = "Hello, Python!"
print("String:", my_string, type(my_string))

String: Hello, Python! <class 'str'>
```

Dictionary:

```
# Dictionary
my_dict = {
    'name': 'John',
    'age': 30,
    'city': 'New York'
}

print("Dictionary:", my_dict, type(my_dict))

# Dictionary Operations
print("\nDictionary Operations:")
print("Access by key (my_dict['name']):", my_dict['name'])
my_dict['email'] = 'john@example.com'
print("After adding a key-value pair:", my_dict)
del my_dict['age']
print("After deleting a key-value pair:", my_dict)

Dictionary: {'name': 'John', 'age': 30, 'city': 'New York'} <class 'dict'>

Dictionary Operations:
Access by key (my_dict['name']): John
After adding a key-value pair: {'name': 'John', 'age': 30, 'city': 'New York', 'email': 'john@example.com'}
After deleting a key-value pair: {'name': 'John', 'city': 'New York', 'email': 'john@example.com'}
```

Set:

```
# Set (mutable, unordered collection of unique elements)
my_set = {1, 2, 3, 3, 4, 5} # Duplicates are removed
print("Set:", my_set, type(my_set))

# Set Operations
print("\nSet Operations:")
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
print("Set 1:", set1)
print("Set 2:", set2)
print("Union (set1 | set2):", set1 | set2)
print("Intersection (set1 & set2):", set1 & set2)
print("Difference (set1 - set2):", set1 - set2)
print("Symmetric Difference (set1 ^ set2):", set1 ^ set2)

Set: {1, 2, 3, 4, 5} <class 'set'>

Set Operations:
Set 1: {1, 2, 3, 4, 5}
Set 2: {4, 5, 6, 7, 8}
Union (set1 | set2): {1, 2, 3, 4, 5, 6, 7, 8}
Intersection (set1 & set2): {4, 5}
Difference (set1 - set2): {1, 2, 3}
Symmetric Difference (set1 ^ set2): {1, 2, 3, 6, 7, 8}
```

Boolean:

```
# 5. Boolean Type
print("\n### Boolean Type ###")

# Boolean values
x = True
y = False
print("Boolean values:", x, y)

# Boolean operations
print("Boolean Operations:")
print("AND (True and False):", True and False)
print("OR (True or False):", True or False)
print("NOT (not True):", not True)

# Comparison operators
print("Comparison Operators:")
print("Equal (5 == 5):", 5 == 5)
print("Not Equal (5 != 3):", 5 != 3)
print("Greater Than (5 > 3):", 5 > 3)
print("Less Than (5 < 3):", 5 < 3)
print("Greater Than or Equal (5 >= 5):", 5 >= 5)
print("Less Than or Equal (3 <= 5):", 3 <= 5)

### Boolean Type ###
Boolean values: True False
Boolean Operations:
AND (True and False): False
OR (True or False): True
NOT (not True): False
Comparison Operators:
Equal (5 == 5): True
Not Equal (5 != 3): True
Greater Than (5 > 3): True
Less Than (5 < 3): False
Greater Than or Equal (5 >= 5): True
Less Than or Equal (3 <= 5): True
```

Writing and reading a file:

```
# Writing to a file (example lang ha)
print("Writing to a file...")
with open("example.txt", "w") as file:
    file.write("Hello, Python!\nThis is a test file.\nLearning file handling.")#Note: Dapoi exactly defined yung file location name otherwise di malolocate yung file

# Reading from a file
print("Reading from a file...")
with open("example.txt", "r") as file:
    content = file.read()
    print("File content:\n", content)

# Reading lines from a file
print("Reading lines from a file...")
with open("example.txt", "r") as file:
    lines = file.readlines()
    print("Lines:", lines)

Writing to a file...
Reading from a file...
File content:
Hello, Python!
This is a test file.
Learning file handling.
Reading lines from a file...
Lines: ['Hello, Python!\n', 'This is a test file.\n', 'Learning file handling.']
```

If statement:

```
# If statement
x = 10
if x > 5:
    print("x is greater than 5")

x is greater than 5
```

If-else statement:

```
# If-Else statement
y = 3
if y > 5:
    print("y is greater than 5")
else:
    print("y is not greater than 5")

y is not greater than 5
```

If-elif statement:

```
# If-Elif-Else statement
z = 5
if z > 5:
    print("z is greater than 5")
elif z < 5:
    print("z is less than 5")
else:
    print("z is equal to 5")

z is equal to 5
```

For loop:

```
# For Loop
print("For loop example:")
for i in range(5):
    print(i, end=" ")
print()

For loop example:
0 1 2 3 4
```

Iterating through a list and summing numbers:

```
# Iterating through a list and summing numbers
numbers = [1, 2, 3, 4, 5]
total = 0
for num in numbers:
    total += num
print("Sum of numbers:", total)

Sum of numbers: 15
```

Printing patterns using nested loops:

```
# Printing patterns using nested for Loops
print("Pattern example:")
for i in range(1, 6):
    for j in range(i):
        print("*", end=" ")
    print()

Pattern example:
*
**
***
****
*****
```

Counting vowels:

```
# Counting vowels in a string
text = "Hello, Python!"
vowels = "aeiouAEIOU"
count = 0
for char in text:
    if char in vowels:
        count += 1
print("Number of vowels in '{}': {}".format(text, count))

Number of vowels in 'Hello, Python!': 3
```

Extracting even numbers from a list:

```
# Extracting even numbers from a list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = []
for num in numbers:
    if num % 2 == 0:
        even_numbers.append(num)
print("Even numbers:", even_numbers)

Even numbers: [2, 4, 6, 8, 10]
```

While loop:

```
# While Loop
print("\nWhile loop example:")
i = 0
while i < 5:
    print(i, end=" ")
    i += 1
print()
```

While loop example:
0 1 2 3 4

Len:

```
# Len()
my_list = [1, 2, 3, 4, 5]
print("Length of list:", len(my_list))
```

Length of list: 5

Range:

```
# range()
print("Range example:", list(range(1, 6)))
```

Range example: [1, 2, 3, 4, 5]

Enumeration:

```
# enumerate()
fruits = ['apple', 'banana', 'cherry']
print("Enumerate example:")
for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}")
```

Enumerate example:
0: apple
1: banana
2: cherry

Zip:

```
# zip()
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 35]
print("Zip example:")
for name, age in zip(names, ages):
    print(f"{name} is {age} years old")
```

Zip example:
Alice is 25 years old
Bob is 30 years old
Charlie is 35 years old

Map:

```
# map()
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, numbers))
print("Map example:", squared)
```

Map example: [1, 4, 9, 16, 25]

Calculator:

```
def calculator():
    print("Simple Calculator")
    print("Operations: +, -, *, /")

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

        if operation == '+':
            result = num1 + num2
        elif operation == '-':
            result = num1 - num2
        elif operation == '*':
            result = num1 * num2
        elif operation == '/':
            if num2 == 0:
                return "Error: Division by zero"
            result = num1 / num2
        else:
            return "Invalid operation"

        return f"{num1} {operation} {num2} = {result}"

    except ValueError:
        return "Invalid input. Please enter numbers only."

if __name__ == "__main__":
    while True:
        output = calculator()
        print(output)
        another = input("Do you want to perform another calculation? (yes/no: ).lower()")
        if another != 'yes':
            break
    print("Thank you for using the calculator!")
```

```
Simple Calculator
Operations: +, -, *, /
Enter first number: 1
Enter second number: 2
Enter operation: +
1.0 + 2.0 = 3.0
Do you want to perform another calculation? (yes/no): no
Thank you for using the calculator!
```

Quiz program:

```
def quiz_program():
    questions = [
        ("question": "Capital of France?", "answer": "Paris"),
        ("question": "2 + 2 = ?", "answer": "4"),
        ("question": "Largest planet?", "answer": "Jupiter")
    ]

    score = 0
    total = len(questions)

    print("Quiz Time!")

    for q in questions:
        print(f"\n{q['question']}")
        user_answer = input("Answer: ").strip().lower()
        correct_answer = q['answer'].lower()

        if user_answer == correct_answer:
            print("Correct!")
            score += 1
        else:
            print(f"Wrong! It's {q['answer']}")

    grade = (score / total) * 100
    print(f"\nYour score: {score}/{total}")
    print(f"Grade: {grade:.2f}%")

    if grade >= 80:
        print("Great!")
    elif grade >= 60:
        print("Okay.")
    else:
        print("Try again!")

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

Quiz Time!

Capital of France?
Answer: sd
Wrong! It's Paris.

2 + 2 = ?
Answer: da
Wrong! It's 4.

Largest planet?
Answer: s
Wrong! It's Jupiter.

Your score: 0/3
Grade: 0.00%
Try again!

Temperature converter:

```
def temperature_converter():
    print("Temperature Converter")
    print("1. Celsius to Fahrenheit")
    print("2. Fahrenheit to Celsius")

    try:
        choice = int(input("Enter 1 or 2: "))
        if choice == 1:
            celsius = float(input("Enter °C: "))
            fahrenheit = (celsius * 9/5) + 32
            print(f"{celsius}°C = {fahrenheit}°F")
        elif choice == 2:
            fahrenheit = float(input("Enter °F: "))
            celsius = (fahrenheit - 32) * 5/9
            print(f"{fahrenheit}°F = {celsius}°C")
        else:
            print("Invalid choice.")
    except ValueError:
        print("Invalid input. Please enter a number.")

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

```
Temperature Converter
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
Enter 1 or 2: 1
Enter °C: 32
32.0°C = 89.6°F
```

Basic class and object:

```
# Basic class example(using class item)
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def show(self):
        print(f"Name: {self.name}, Price: ${self.price:.2f}")

# Example usage
if __name__ == "__main__":
    item1 = Item("Book", 19.99)
    item2 = Item("Pen", 2.50)
    item1.show()
    item2.show()
```

```
Name: Book, Price: $19.99
Name: Pen, Price: $2.50
```

```
# Create an object
obj = MyClass(10, 20)
obj.display()

Values: 10, 20
```

Sim card example:

SimCard and CellPhone example

class SimCard:

"""Represents a SIM card with a number, provider, and activation status."""

def __init__(self, number, provider):

"""Initializes a new SimCard object."""

self.number = number

self.provider = provider

self.is_active = False

print(f"SIM card {self.number} created.")

def activate(self):

"""Activates the SIM card."""

if not self.is_active:

self.is_active = True

print(f"SIM card {self.number} activated.")

else:

print(f"SIM card {self.number} is already active.")

def deactivate(self):

```

        """Deactivates the SIM card."""
        if self.is_active:
            self.is_active = False
            print(f"SIM card {self.number} deactivated.")
        else:
            print(f"SIM card {self.number} is already inactive.")

    def get_info(self):
        """Returns a string with the SIM card's information."""
        status = "active" if self.is_active else "inactive"
        return f"SIM: {self.number} ({self.provider}) - {status}"

class CellPhone:
    """Represents a cell phone with a model, manufacturer, and an optional SIM card."""
    def __init__(self, model, manufacturer):
        """Initializes a new CellPhone object."""
        self.model = model
        self.manufacturer = manufacturer
        self.sim_card = None
        print(f"Cell phone {self.manufacturer} {self.model} created.")

    def insert_sim_card(self, sim_card):
        """Inserts a SimCard object into the cell phone."""
        if isinstance(sim_card, SimCard):
            if self.sim_card is None:
                self.sim_card = sim_card
                print(f"SIM card {sim_card.number} inserted into {self.manufacturer} {self.model}.")
            else:
                print(f"A SIM card is already present in {self.manufacturer} {self.model}.")
        else:
            print("Invalid SIM card object provided.")

    def remove_sim_card(self):
        """Removes the SIM card from the cell phone."""
        if self.sim_card:
            removed_sim = self.sim_card
            self.sim_card = None
            print(f"SIM card {removed_sim.number} removed from {self.manufacturer} {self.model}.")
            return removed_sim
        else:
            print(f"No SIM card to remove from {self.manufacturer} {self.model}.")
            return None

    def make_call(self, number):

```

```

        """Attempts to make a call using the inserted SIM card."""
        if self.sim_card:
            if self.sim_card.is_active:
                print(f"{self.manufacturer} {self.model} calling {number} using SIM
{self.sim_card.number}...")
            else:
                print(f"Cannot make call from {self.manufacturer} {self.model}. SIM card
{self.sim_card.number} is not active.")
        else:
            print(f"Cannot make call from {self.manufacturer} {self.model}. No SIM card inserted.")

    def get_phone_info(self):
        """Returns a string with the cell phone's information, including SIM details if present."""
        sim_info = self.sim_card.get_info() if self.sim_card else "No SIM inserted"
        return f"Phone: {self.manufacturer} {self.model} - SIM: {sim_info}"

# Example Usage
if __name__ == "__main__":
    sim1 = SimCard("09171234567", "Globe")
    sim2 = SimCard("09187654321", "Smart")

    phone1 = CellPhone("iPhone 13", "Apple")
    phone2 = CellPhone("Galaxy S21", "Samsung")

    phone1.insert_sim_card(sim1)
    sim1.activate()
    phone1.make_call("09191112222")
    print(phone1.get_phone_info())

    phone1.remove_sim_card()
    phone1.make_call("09191112222")

    phone2.insert_sim_card(sim2)
    print(phone2.get_phone_info())
    sim2.deactivate()
    phone2.make_call("09193334444")

```

Procurement System example:

```

# Procurement System example
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

```

```

def get_description(self):
    return f"Item: {self.name}, Price: ${self.price}"

class RawMaterial(Item):
    def __init__(self, name, price, quantity, unit):
        super().__init__(name, price)
        self.quantity = quantity
        self.unit = unit

    def get_description(self):
        return f"Raw Material: {self.name}, Price: ${self.price}, Quantity: {self.quantity} {self.unit}"

class OfficeSupply(Item):
    def __init__(self, name, price, category):
        super().__init__(name, price)
        self.category = category

    def get_description(self):
        return f"Office Supply: {self.name}, Price: ${self.price}, Category: {self.category}"

class ProcurementSystem:
    def __init__(self):
        self.items = []

    def add_item(self, item):
        self.items.append(item)
        print(f"Added {item.name} to the procurement system.")

    def display_items(self):
        print("\nItems in Procurement System:")
        for item in self.items:
            print(item.get_description())

# Using the Procurement System
print("\nProcurement System example:")
proc_system = ProcurementSystem()

steel = RawMaterial("Steel", 500, 100, "kg")
paper = OfficeSupply("A4 Paper", 5, "Stationery")

proc_system.add_item(steel)
proc_system.add_item(paper)
proc_system.display_items()

```

Reservation system:

Reservation System example

```
class Reservation:
    def __init__(self, guest_name, date, num_guests):
        self.guest_name = guest_name
        self.date = date
        self.num_guests = num_guests

    def get_details(self):
        return f"Reservation for {self.guest_name} on {self.date} for {self.num_guests} guests"

class HotelReservation(Reservation):
    def __init__(self, guest_name, date, num_guests, room_type):
        super().__init__(guest_name, date, num_guests)
        self.room_type = room_type

    def get_details(self):
        return f"Hotel {super().get_details()}, Room Type: {self.room_type}"

class RestaurantReservation(Reservation):
    def __init__(self, guest_name, date, num_guests, table_number):
        super().__init__(guest_name, date, num_guests)
        self.table_number = table_number

    def get_details(self):
        return f"Restaurant {super().get_details()}, Table Number: {self.table_number}"

class ReservationSystem:
    def __init__(self):
        self.reservations = []

    def add_reservation(self, reservation):
        self.reservations.append(reservation)
        print("Reservation added successfully!")

    def display_reservations(self):
        print("\nReservations:")
        for reservation in self.reservations:
            print(reservation.get_details())

# Using the Reservation System
print("\nReservation System example:")
res_system = ReservationSystem()
```

```
hotel_res = HotelReservation("John Smith", "2023-05-15", 2, "Deluxe")
restaurant_res = RestaurantReservation("Jane Doe", "2023-05-20", 4, "12")
```

```
res_system.add_reservation(hotel_res)
res_system.add_reservation(restaurant_res)
res_system.display_reservations()
```

Login System:

```
# Login System example
```

```
class User:
```

```
    def __init__(self, username, password):
        self.username = username
        self.password = password
```

```
    def verify_credentials(self, username, password):
        return self.username == username and self.password == password
```

```
class Admin(User):
```

```
    def __init__(self, username, password, admin_level):
        super().__init__(username, password)
        self.admin_level = admin_level
```

```
    def display_admin_info(self):
        return f"Admin: {self.username}, Level: {self.admin_level}"
```

```
class LoginSystem:
```

```
    def __init__(self):
        self.users = []
```

```
    def add_user(self, user):
        self.users.append(user)
        print(f"User '{user.username}' added to the system.")
```

```
    def login(self, username, password):
        for user in self.users:
            if user.verify_credentials(username, password):
                print(f"Login successful: Welcome, {username}!")
                if isinstance(user, Admin):
                    print(user.display_admin_info())
                return True
        print("Login failed: Invalid credentials.")
        return False
```

```
# Using the Login System
```

```
print("\nLogin System example:")
login_system = LoginSystem()

regular_user = User("john", "password123")
admin_user = Admin("admin", "admin123", 2)

login_system.add_user(regular_user)
login_system.add_user(admin_user)

login_system.login("john", "password123")
login_system.login("admin", "admin123")
login_system.login("unknown", "wrong")
```