**Day 3 Notebook 1**

**multiply function**

def multiply\_numbers(a, b):

return a \* b

# Call the function and print the result

result = multiply\_numbers(3, 4)

print(result) # Expected output: 12

**rectangle\_properties function**

def rectangle\_properties(length, width):

area = length \* width

perimeter = 2 \* (length + width)

print(f"Area: {area}, Perimeter: {perimeter}")

# Call the function

rectangle\_properties(length=5, width=3)

**bmi function**

def calculate\_bmi(weight=70, height=1.75):

"""Calculate BMI, determine Health Category"""

BMI = weight / height \*\* 2

if BMI < 18.5:

    healthCategory = "Underweight"

  elif BMI < 24.9:

    healthCategory = "Normal"

  elif BMI < 29.9:

    healthCategory = "Overweight"

  else:

    healthCategory = "Obese"

  print(f"BMI: {BMI:.2f}, Category: {healthCategory}")

# call the function

calculate\_bmi() #default values

calculate\_bmi(85, 1.8)

**return values**

def calculate\_square(number):

square = number \* number

return square

# Call the function and print the result

result = calculate\_square(4)

print(f"The square of 4 is: {result}")

**optional arguments**

def format\_book\_title(title, author, subtitle=''):

if subtitle:

formatted\_title = f"{title}: {subtitle} by {author}"

else:

formatted\_title = f"{title} by {author}"

return formatted\_title

# Call the function without the optional argument

print(format\_book\_title("The Great Gatsby", "F. Scott Fitzgerald"))

# Call the function with the optional argument

print(format\_book\_title("The Great Gatsby", "F. Scott Fitzgerald", "A Novel"))

**main function**

def print\_greeting(name):

print(f"Hello, {name}! Welcome to the program.")

def main():

# Display a welcome message

print("Welcome to the Greeting Program")

# Prompt the user for their name

user\_name = input("Please enter your name: ")

# Call the custom function with the input name

print\_greeting(user\_name)

# Call the main function to start the program

main()

**lambda function**

# Define a lambda function to capitalize the first letter of each word in a string

capitalize\_words = lambda s: ' '.join(word.capitalize() for word in s.split())

# Call the lambda function with a sample input

sample\_input = "hello world from the lambda function"

result = capitalize\_words(sample\_input)

# Print the result

print(f"Original string: {sample\_input}")

print(f"Capitalized string: {result}")

**Day 3 Notebook 2**

**imports**

# import\_demo.py

# Import the entire math module

import math

# Use a function from the math module

print("Using math module:")

print(f"Square root of 16 is {math.sqrt(16)}")

print()

# Import specific functions from math and datetime modules

from math import pi, sin

# Use the imported functions

print("Using specific functions:")

print(f"Value of pi is {pi}")

print(f"Sin of pi/2 is {sin(pi/2)}")

from datetime import datetime, timedelta

# Use datetime functions

now = datetime.now()

print(f"Current date and time: {now}")

one\_week\_later = now + timedelta(weeks=1)

print(f"Date and time one week later: {one\_week\_later}")

print()

# Import datetime module with alias

import datetime as dt

# Use the alias to call the function

print("Using alias for datetime module:")

now = dt.datetime.now()

print(f"Current date and time using alias: {now}")

print()

# Import all functions from the math module

from math import \*

# Use the imported functions

print("Using wildcard import from math module:")

print(f"Cosine of 0 is {cos(0)}")

print(f"Exponential of 1 is {exp(1)}")

print()

**Day 3 Notebook 3**

**Cat Class**  
  
class Cat:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def meow(self):

print(f"{self.name} is meowing.")

def scratch(self):

print(f"{self.name} is scratching.")

# Create an instance of the Cat class

my\_cat = Cat("Whiskers", 3)

# Call the methods

my\_cat.meow() # Output: Whiskers is meowing.

my\_cat.scratch() # Output: Whiskers is scratching.

**Cat class with mutators and accessors**

class Cat:

def \_\_init\_\_(self, name, age):

self.\_\_name = name

self.\_\_age = age

# Accessor (getter) for name

def getName(self):

return self.\_\_name

# Mutator (setter) for name

def setName(self, name):

self.\_\_name = name

# Accessor (getter) for age

def getAge(self):

return self.\_\_age

# Mutator (setter) for age

def setAge(self, age):

self.\_\_age = age

def meow(self):

print(f"{self.\_\_name} is meowing.")

def scratch(self):

print(f"{self.\_\_name} is scratching.")

# Main function to demonstrate the use of accessors and mutators

def main():

# Create an instance of Cat

my\_cat = Cat("Whiskers", 2)

# Use accessors to get attribute values

print(f"Name: {my\_cat.getName()}")

print(f"Age: {my\_cat.getAge()}")

# Use mutators to set attribute values

my\_cat.setName("Shadow")

my\_cat.setAge(3)

# Use accessors to get updated attribute values

print(f"Updated Name: {my\_cat.getName()}")

print(f"Updated Age: {my\_cat.getAge()}")

# Demonstrate other methods

my\_cat.meow()

my\_cat.scratch()

# Call the main function to start the program

main()

**Cat Subclass**

class Siamese(Cat):

def \_\_init\_\_(self, name, age, color):

super().\_\_init\_\_(name, age)

self.\_\_color = color

# Accessor (getter) for color

def getColor(self):

return self.\_\_color

# Mutator (setter) for color

def setColor(self, color):

self.\_\_color = color

def purr(self):

print(f"{self.getName()} is purring happily.")

# Main function to demonstrate the use of the Siamese class

def main():

# Create an instance of Siamese

my\_siamese = Siamese("Luna", 2, "Cream")

# Use accessors to get attribute values

print(f"Name: {my\_siamese.getName()}")

print(f"Age: {my\_siamese.getAge()}")

print(f"Color: {my\_siamese.getColor()}")

# Use mutators to set attribute values

my\_siamese.setName("Milo")

my\_siamese.setAge(3)

my\_siamese.setColor("Chocolate")

# Use accessors to get updated attribute values

print(f"Updated Name: {my\_siamese.getName()}")

print(f"Updated Age: {my\_siamese.getAge()}")

print(f"Updated Color: {my\_siamese.getColor()}")

# Demonstrate inherited and new methods

my\_siamese.meow()

my\_siamese.scratch()

my\_siamese.purr()

# Call the main function to start the program

main()

**Day 3 Notebook 4**

**file append**

from pathlib import Path

FILEPATH = Path('.')

FILENAME = FILEPATH / 'simplefileops.txt'

with open(FILENAME, 'a') as f:

    f.write('I enjoy solving complex problems.\n')

# Verify the contents of the file

with open(FILENAME) as f:

    for line in f:

        print(line.replace("\n", "")) # replace also removes newlines

**read/write JSON**

#!/usr/bin/env python3

# json\_data\_writer\_reader.py

# JSON data writer and reader

from pathlib import Path

import json

FILEPATH = Path('/content')

# Define a dictionary with several key-value pairs

data = {

"name": "Alice",

"age": 30,

"city": "Wonderland",

"is\_student": False,

"skills": ["Python", "Data Analysis", "Machine Learning"]

}

# Specify the filename

filename = FILEPATH / 'data.json'

# Write the dictionary to a JSON file

with open(filename, 'w') as f:

json.dump(data, f)

# Open the JSON file and read the contents

with open(filename, 'r') as f:

content = json.load(f)

# Display the contents

print(json.dumps(content, indent=4))

**try/except on a file**

#!/usr/bin/env python3

# exceptions.py

def main():

    while True:

        try:

            # Prompt the user for a file name

            filename = input("Enter the file name: ")

            # Simulate an IOError if the specific file name is entered

            if filename == "simulate\_error.txt":

                raise IOError("Simulated IOError for demonstration purposes.")

            # Attempt to open and read the file

            with open(filename, 'r') as file:

                contents = file.read()

        except FileNotFoundError:

            # Handle the case where the file does not exist

            print(f"Error: The file '{filename}' does not exist. Please try again.")

            continue

        except IOError as e:

            # Handle other I/O errors, including the simulated IOError

            print(f"Error: {e}")

            continue

        else:

            # If no exceptions occur, display the file contents

            print("File Contents:")

            print(contents)

            break

# Call the main function to start the program

if \_\_name\_\_ == "\_\_main\_\_":

  main()