**Day 2 Notebook 1**

**while loop**

# Define the secret word

secret\_word = "python"

# Initialize an empty string for the user's guess

guess = ""

# Use a while loop to keep asking until the user guesses correctly

while guess != secret\_word:

# Ask the user to guess the secret word

guess = input("Guess the secret word: ")

# Provide feedback to the user

if guess == secret\_word:

print("Congratulations! You guessed the secret word.")

else:

print("Incorrect. Try again!")

# Print a final message

print("Game over. You guessed it right!")

**continue**

# Demonstrate a continue statement

more = "y"

while more.lower() == "y":

    miles\_driven = float(input("Enter miles driven: "))

    gallons\_used = float(input("Enter gallons of gas used: "))

    # validate input

    if miles\_driven <= 0 or gallons\_used <= 0:

          print("Both entries must be greater than zero. Try again.")

          continue   #  while more.lower() == "y":

    mpg = round(miles\_driven / gallons\_used, 2)

    print("Miles Per Gallon:", mpg)

    more = input("Continue? (y/n): ")

    print()

print("Okay, bye!")

**future value**

# Display welcome message

print("Welcome to the Future Value Calculator!")

while True:

# Input monthly investment, yearly interest rate, and years

monthly\_investment = float(input("Enter monthly investment amount: "))

yearly\_interest\_rate = float(input("Enter yearly interest rate (as a percentage): "))

years = int(input("Enter number of years: "))

# Validate input values

if monthly\_investment <= 0 or yearly\_interest\_rate <= 0 or years <= 0:

print("All entries must be greater than zero. Try again.")

continue # Skip the rest of the loop and prompt the user again

# Convert yearly interest rate to monthly interest rate

monthly\_interest\_rate = yearly\_interest\_rate / 100 / 12

# Convert years to months

months = years \* 12

# Set future value to 0

future\_value = 0

# Calculate the future value using a while loop

month = 0

while month < months:

future\_value += monthly\_investment

interest = future\_value \* monthly\_interest\_rate

future\_value += interest

month += 1

# Display future value

print(f"Future value: ${future\_value:.2f}")

# Ask if user wants to continue

continue\_choice = input("Do you want to continue? (yes/no): ").lower()

if continue\_choice != "yes":

break

print("Thank you for using the Future Value Calculator!")

**for loops**

sum\_nums = 0 # accumulator

for number in range(1, 6): # range requires 6 to include 5

    sum\_nums = sum\_nums + number

# this print is not indented so it is not in the loop body

print("The sum of 1 to 5 is " + str(sum\_nums))

**for loop w/else**  
  
# Show primes, or factors of non-primes

for n in range(2, 10):

for x in range(2, n):

if n % x == 0:

print(n, 'equals', x, '\*', n // x)

break # inner loop break

else:

# This else clause executes if no break occurred in the inner loop

print(n, 'is a prime number')

**Day 2 Notebook 2**

**displaying list values**

# Create a list of elements

elements\_list = ["apple", "banana", "cherry", "date", "elderberry"]

# Display elements using positive index values

print("List elements using positive index values:")

print("Element at index 0:", elements\_list[0])

print("Element at index 2:", elements\_list[2])

print("Element at index 4:", elements\_list[4])

# Determine and display corresponding negative index values

print("\nList elements using corresponding negative index values:")

print("Element at index -5:", elements\_list[-5]) # Corresponds to index 0

print("Element at index -3:", elements\_list[-3]) # Corresponds to index 2

print("Element at index -1:", elements\_list[-1]) # Corresponds to index 4

# Create a tuple with the same elements as the list

elements\_tuple = ("apple", "banana", "cherry", "date", "elderberry")

# Display elements using positive index values

print("\nTuple elements using positive index values:")

print("Element at index 1:", elements\_tuple[1])

print("Element at index 3:", elements\_tuple[3])

print("Element at index 4:", elements\_tuple[4])

# Determine and display corresponding negative index values

print("\nTuple elements using corresponding negative index values:")

print("Element at index -4:", elements\_tuple[-4]) # Corresponds to index 1

print("Element at index -2:", elements\_tuple[-2]) # Corresponds to index 3

print("Element at index -1:", elements\_tuple[-1]) # Corresponds to index 4

**modifying list elements**  
  
# Create a list of fruits

fruits = ["apple", "banana", "cherry", "date", "elderberry"]

print("Initial fruits:", fruits)

# Save the value of the next to last element

saved\_value = fruits[-2]

# Modify the next to last element

fruits[-2] = "fig"

print("Modified fruits:", fruits)

# Restore the modified element

fruits[-2] = saved\_value

print("Restored fruits:", fruits)

**append/insert**

# Create a list with initial elements

fruits = ["apple", "banana", "cherry"]

print("Initial list:", fruits)

# Append a new element to the end of the list

fruits.append("date")

print("List after appending 'date':", fruits)

# Insert a new element at the second position (index 1)

fruits.insert(1, "blueberry")

print("List after inserting 'blueberry' at index 1:", fruits)

**remove/del**

# Create a list with initial elements

fruits = ["apple", "banana", "cherry", "date", "elderberry"]

print("Initial list:", fruits)

# Remove a specific element using the remove method

fruits.remove("banana")

print("List after removing 'banana':", fruits)

# Remove an element at a specific position using the del statement

del fruits[2] # Removing the element at index 2

print("List after deleting element at index 2:", fruits)

**pop**

# Create a list with initial elements

colors = ["red", "green", "blue", "yellow"]

print("Initial list:", colors)

# Use pop without an index to remove the last element

last\_color = colors.pop()

print("Popped element:", last\_color)

print("List after popping the last element:", colors)

# Use pop with an index to remove the second element

second\_color = colors.pop(1)

print("Popped element at index 1:", second\_color)

print("List after popping the second element:", colors)

# Use pop with an index to remove the first element

first\_color = colors.pop(0)

print("Popped element at index 0:", first\_color)

print("List after popping the first element:", colors)

# Attempt to pop from an empty list to demonstrate an IndexError exception

try:

empty\_pop = colors.pop()

print("Popped element from empty list:", empty\_pop)

except IndexError as e:

print("IndexError exception caught:", e)

**sorting**

# Create a list with initial numerical elements

numbers = [42, 23, 16, 15, 8]

print("Initial list of numbers:", numbers)

# Use the sort method to sort the list in ascending order

numbers.sort()

print("List of numbers after sort():", numbers)

# Create a list with initial string elements

fruits = ["banana", "apple", "cherry", "date", "elderberry"]

print("\nInitial list of fruits:", fruits)

# Use the sorted function to sort the list in ascending order

sorted\_fruits = sorted(fruits)

print("Original list of fruits after sorted():", fruits)

print("New sorted list of fruits:", sorted\_fruits)

**2 Dimensional Lists**

# Create a two-dimensional list with initial elements

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Initial two-dimensional list:")

print(matrix)

# Append a new sublist to the two-dimensional list

new\_row = [10, 11, 12]

matrix.append(new\_row)

print("Append new sublist:")

print(matrix)

# Insert a new element into one of the sublists

matrix[1].insert(1, 99) # Insert 99 into the second sublist at index 1

print("Insert element into the second sublist:")

print(matrix)

# Delete an element from one of the sublists

del matrix[0][2] # Delete the third element of the first sublist

print("Deleting element from the first sublist:")

print(matrix)

**lists and for loops**

# Create a list with initial numerical elements

numbers = [2, 4, 6, 8, 10]

print("Initial list of numbers:", numbers)

# Use a for loop to iterate through each element in the list

print("\nProcessed list elements (each element multiplied by 2):")

for number in numbers:

result = number \* 2

print(result)

**list comprehension**

# Create a list with initial numerical elements

number\_list = [2, 4, 6, 8, 10]

print("Initial list of numbers:", number\_list)

# Use a list comprehension to process each element in the list

processed\_numbers = [number \* 2 for number in number\_list]

print("Processed list elements (each element multiplied by 2):")

print(processed\_numbers)

**list slicing**

# Create a list with initial numerical elements

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

print("Initial list of numbers:", numbers)

# Use slicing to print the first five elements of the list

first\_five = numbers[:5]

print("\nFirst five elements:", first\_five)

# Use slicing to print the last three elements of the list

last\_three = numbers[-3:]

print("Last three elements:", last\_three)

# Use slicing to print every second element of the list

every\_second = numbers[::2]

print("Every second element:", every\_second)

# Use slicing to modify a middle segment of the list

numbers[4:7] = [50, 60, 70]

print("\nModified list after changing middle segment:")

print(numbers)

**create and unpack tuple**

# Create a tuple with initial elements

my\_tuple = (1, 2, 3, 4, 5)

print("Original tuple:", my\_tuple)

# Unpack the tuple into individual variables

a, b, c, d, e = my\_tuple

# Print each variable to show the unpacked values

print("\nUnpacked values:")

print("a =", a)

print("b =", b)

print("c =", c)

print("d =", d)

print("e =", e)

**Day 2 Notebook 3**

**create/access dictionary**

# Create a dictionary with initial key-value pairs

person = {

"name": "Alice",

"age": 30,

"city": "New York",

"email": "alice@example.com",

"phone": "123-456-7890"

}

print("Original dictionary:", person)

# Access and print elements using the [] syntax

print("\nAccessing elements using [] syntax:")

print("Name:", person["name"])

print("Age:", person["age"])

# Access and print elements using the get method

print("\nAccessing elements using get method:")

print("City:", person.get("city"))

print("Email:", person.get("email"))

# Demonstrate the use of the get method with a default value for a non-existent key

print("\nUsing get method with a default value:")

print("Address:", person.get("address", "Not Available"))

**delete from dictionary**

# Create a dictionary with initial key-value pairs

student = {

"name": "John",

"age": 21,

"major": "Computer Science",

"university": "XYZ University",

"email": "john@example.com"

}

print("Original dictionary:", student)

# Delete an element using the del statement

del student["email"]

print("\nDictionary after deleting 'email' using del:")

print(student)

# Delete an element using the pop method

major = student.pop("major")

print("\nDictionary after deleting 'major' using pop:")

print(student)

print("Popped value:", major)

# Use the pop method to delete a non-existent key with a default value

address = student.pop("address", "Not Available")

print("\nUsing pop method to delete a non-existent key 'address' with a default value:")

print("Result:", address)

print("Final dictionary:", student)

**process dictionary elements with a for loop**

# Create a dictionary with initial key-value pairs

student = {

"name": "Alice",

"age": 22,

"major": "Physics",

"university": "XYZ University",

"email": "alice@example.com"

}

# Use a for loop to iterate through the dictionary

print("Dictionary elements:")

for key, value in student.items():

print(f"{key}: {value}")

**dictionary of dictionaries**

# Create a dictionary of dictionaries

students = {

"1001": {"name": "Alice", "age": 22, "major": "Physics"},

"1002": {"name": "Bob", "age": 23, "major": "Chemistry"},

"1003": {"name": "Charlie", "age": 21, "major": "Mathematics"},

"1004": {"name": "David", "age": 24, "major": "Biology"},

"1005": {"name": "Eve", "age": 22, "major": "Computer Science"}

}

# Use a for loop to iterate through the outer dictionary

print("Dictionary of dictionaries:")

for student\_id, details in students.items():

print(f"\nStudent ID: {student\_id}")

for key, value in details.items():

print(f" {key}: {value}")