Workshop Resources

Login: user@ce
password:3nG5tuDt

1. Facilitator: https://damayanthiherath.net
2. Introduction to programming: Sections 1-3
3. Biological Data Analysis with Python
4. Data Visualization More resources
5. The files for text processing exercise
6. Powerpoint slides
7. Feedback Form

1. Algorithmic Thinking (Time: 10 Mins)

ILO: The participants will get experience in working out an algorithm for a given problem and speaking it out (while working as a group)

Divide the class into 4 by asking them to count to 1 to 4. Ask them to solve below problems and explain the steps

1. How to make a cup of tea?
2. How to organize a set of books in a library by alphabetical order? Imagine you have 10 books.
3. You are given a deck of cards with letters in an order. Reverse them'; Tell us how you did it?
4. You and your friends are planning a picnic. You need to pack a picnic basket that ensures the majority's preferences and needs are met while ensuring the basket doesn't exceed its weight limit. How will you do that?

Communicating Algorithms: paired Activity (20 mins):

ILO: The participants will get hands on writing a pseudocode/drawing a flowchart to present an algorithm for a given problem.

First 10 mins:

Write the pseudocode/flowchart to calculate the rainfall given the inputs, temperature and humidity

Next 10 mins

Complete the rest of it output whether a landslide would occur or not; A landslide will occur if rainfall is more than 200 mm

2, Python Basics

ILO: The participants will get hands on experience on Python environment

```
# This is a simple Python program to print Hello, World!
print("Hello, World!")
```

Excercise: Display Hello with your name. What if instead of you, it is your friend or someone else and the name changes.?

Print the word 'Nenathambara' . Make the output seem underlined using the asterisk sign '*' .You can use double

quotes or triple quotes.

Type a short para about yourself using double quotes.

Type the same in point form(line after line) using triple quotes and using '\n'

You can only use one print command.Give a heading and underline your essay.

Lets get into our weather analyser. Let us write a program to calculate the rainfall given the temperature and humidity.

```
# Define temperature and humidity
temperature = 25.0  # Example temperature
humidity = 80       # Example humidity
```

Activity 4

ILO : The participants will get hands -on experience on implementing solutions involving Arithmetic, variables and types, Boolean variables and if else conditions, and loops.

Extend the code to calculate the rainfall; Print the value.

Extend the code to output whether there is a landslide risk or not given the rainfall value.

Can you print the numbers less than 500, divisible by 9?

```
>>> for i in range(500):
        if i%9 == 0:
            print(i)
```

Here's something a bit more tricky. Can you print the numbers less than 500, divisible by 5 and 7 and 11?

```
>>> for i in range (500):
```

```
    if i%5==0:
        if i%7==0:
            if i%11 == 0:
                print(i)
```

Thinking expected- If you want to filter something that meets several requirements, use nested if statements.
*Introduce the word nested

Can you print the numbers less than 500, divisible by 4 and 6?

Can you keep printing the characters in a string until you either print all the characters or print the character “7” if it appears in the string?

```
string = input("Enter a string: ")
index = 0
while index < len(string):
  print(string[index])
  if string[index] == '7':
    break
  index += 1
```

4. You are building a basic landslide monitoring system that asks the user to input rainfall values in each loop. The system will:

- Stop and print an alert if **rainfall exceeds 200 mm**.
- Allow the user to stop the system manually by typing "stop."

Checking Multiple Conditions in a Sequence

```
seq = "MGSNKSKPKDASQRRRSLEPAENVHGAGGGAFPASQTPSKPASADGHRGPSAAFAPAAAE"
```

```
if 'GGG' in seq and 'RRR' in seq:
    print('GGG is at position:', seq.find('GGG'))
    print('RRR is at position:', seq.find('RRR'))

if 'WWW' in seq or 'AAA' in seq:
    print('Either WWW or AAA occur in the sequence')

if 'AAA' in seq and not 'PPP' in seq:
    print('AAA occurs in the sequence but not PPP')
```

Exercise

Define a sequence variable called protein_seq with a sample protein sequence that contains multiple amino acid motifs (e.g., 'MGGGKKKRRRAAAWWT').

Write code to:

- Check if both 'GGG' and 'KKK' are present in the sequence, and print their positions.
- Print a message if either 'WWW' or 'SSS' is found.
- Check if 'AAA' exists in the sequence but not 'YYY', and print a message if true.

3. Manipulating Text

Read from a text file

readlines() reads the file line-by-line into a list.

read() reads the entire file as a single string.

```
# Using readlines() to read all lines into a list
text_file = open('text_file.txt', 'r')
lines = text_file.readlines()
text_file.close()
print(lines) # use a for loop to print each line one by one


# Using read() to read the entire file as one string
text_file = open('text_file.txt')
print(text_file.read())
text_file.close()
```

Exercise 01

Write a program to open and read the contents of the given file called neuron_data.txt. Print each line of the file individually, and then close the file. You can use readlines() and then use a for loop to print each line.

Writing to a Text File

Use 'w' mode to create a new file or overwrite an existing one.

```
output_file = open('new_file.txt', 'w')
output_file.write('sample text to be written\n')
output_file.close()
```

Exercise 02

Write a program to create and open a new file called “microbe_counts.txt” in write mode. Write a line that says "Total microbe count: 15", then close the file. Reopen the file to read and print the line you wrote.

Cleaning Text Data

The strip() method removes extra spaces. You can also use rstrip() to remove spaces on the right side only or lstrip() on the left.

```
output_file = open('file.txt')
print(output_file.read().strip())
output_file.close()
```

Exercise 03

First create a text file named “friends.txt” and then add some data with white spaces in the beginning of the sentence. Then write a program to open and read the contents of “data.txt”. Use strip() to remove any leading or trailing whitespace, then print the cleaned line.

Basics of Formatting

Specifier	Explanation	Example Output
:d	Formats an integer.	f"{42:d}" → 42
:f	Formats a float (default 6 decimal places)	f"{3.14159:f}" → 3.141590
:10.2f	Right-align, width=10, 2 decimal places	f"{3.1:10.2f}" → 3.10
:<10.2f	Left-align, width=10, 2 decimal places	f"{3.1:<10.2f}" → 3.10
:+10.2f	Include a sign (+ or -)	f"{3.1:+10.2f}" → +3.10

:010.2f	Pad with zeros (width=10, 2 decimals)	f"{3.1:010.2f}" → 0000003.10
:.0f	Display as a whole number (rounded)	f"{3.6:.0f}" → 4

```
# Integer formatting
num = 42
print(f"{num:d}")        # Output: 42 (integer)
print(f"{num:5d}")       # Output: '   42' (right-aligned, width=5)
print(f"{num:<5d}")      # Output: '42   ' (left-aligned, width=5)

# Float formatting
pi = 3.14159
print(f"{pi:f}")         # Output: 3.141590 (default float)
print(f"{pi:.2f}")       # Output: 3.14 (2 decimal places)
print(f"{pi:10.2f}")     # Output: '      3.14' (width=10, 2 decimals)
print(f"{pi:<10.2f}")    # Output: '3.14      ' (left-aligned)

# Including signs
neg = -3.14
print(f"{pi:+.2f}")      # Output: +3.14 (positive sign)
print(f"{neg:+.2f}")     # Output: -3.14 (negative sign)

# Zero padding
print(f"{pi:010.2f}")    # Output: 0000003.14 (width=10, padded with zeros)

# Rounded floats
large_num = 12345.6789
print(f"{large_num:.0f}") # Output: 12346 (rounded to nearest whole number)
```

Exercise 04

Format the following values as described:

Print an integer 56 with:

- Right-alignment in a field of width 8.
- Left-alignment in a field of width 8.

Print a float 45.678 with:

- Exactly 2 decimal places.
- Right-aligned in a width of 10, with 2 decimal places.
- Zero-padded, width of 8, and rounded to 1 decimal place.

Calculate the average from a list of numbers

```
# calculate average from float numbers
data = [3.53, 3.47, 3.51, 3.72, 3.43]
average = sum(data) / len(data)
print average

# calculate average from integer numbers
data = [1, 2, 3, 4]
average = float(sum(data)) / len(data)
print average
```

Exercise 05

Print the average of [4.56,8.55,9.25,10.32,23.91]

*****Find items common to two lists*****

```
# Reading lines from both files
lines1 = open("file1.txt").readlines()
lines2 = open("file2.txt").readlines()

# Strip newlines and spaces
lines1 = [line.strip() for line in lines1]
lines2 = [line.strip() for line in lines2]

# Comparing the two lists
for line in lines1:
    if line in lines2:
        print(line, 'same one detected')
    else:
        print(line, 'not detected')
```

Exercise 06

Use the given files “cell_cycle_proteins.txt” and “cancer_cell_proteins.txt” that contain gene IDs.

Write code to print each gene ID from cell_cycle_proteins, indicating if it is also present in cancer_cell_proteins.

Sets

Sets are unordered collections of unique elements. They are useful for removing duplicates, finding intersections, unions, and differences between groups of objects.

```
s1 = set('LDFGJLDFGDGD')
print(s1)  # Output: {'J', 'F', 'L', 'G', 'D'}

# Check membership
print('L' in s1)  # True
print('Z' not in s1)  # True
```

With sets, finding the intersection becomes straightforward:

```
data_a = {1, 2, 3, 4, 5, 6}
data_b = {1, 5, 7, 8, 9}

a_and_b = data_a.intersection(data_b)
print(a_and_b)
```

Exercise 07

Define two sets {21, 11, 2, 64, 35, 60 } and {81, 11, 52, 94, 23, 59 } and print their intersection.

Find Common Elements in Multiple Sets

Using reduce() to find common elements in three sets:

```
from functools import reduce

a = {1, 2, 3, 4, 5}
b = {2, 4, 6, 7, 1}
c = {1, 4, 5, 9}

common = reduce(set.intersection, [a, b, c])
print(common)
```

Exercise 08

Define four sets and then find the common elements among them.

Removing Elements from a List

Use pop(), del, and remove() to remove elements:

```
data_a = [1, 2, 3, 4, 5, 6, 7]

data_a.pop()  # Removes and returns last element
del data_a[1]  # Deletes element at index 1
data_a.remove(3)  # Removes first occurrence of the value 3
```

Slicing a List

```
data_a = [1, 2, 3, 4, 5, 6]
print(data_a[:2])  # Output: [1, 2]
```

Exercise 09

Consider the list [23, 55, 67, 94, 35, 77, 12, 74]. Slice it from '67' and create a new list and then pop the last element.

Remove Duplicates from a File

Exercise 10

Open the given file "UniprotID.txt" Then read the content and use a set to remove duplicates from the file.

Reading Tabular Data from a File

Tabular data can be read from a text file and processed into a Python list of lists. This structure makes it easier to manipulate rows and columns.

```
# Read tabular data from a tab-separated text file
table = []

for line in open('lowry_data.txt'):
    table.append(line.strip().split('\t'))

print(table)
```

Exercise 11

Create a text file named data.txt with the following content:

Use tab spaces…

```
name  age  country
Alice  30  USA
Bob  25  UK
Charlie 35  Canada
```

Write a Python script to read the file and print its contents as a list of lists.

****Basic Sorting with sorted() and sort()****

Python provides two main ways to sort data:

- sort(): A method that modifies lists in-place
- sorted(): A built-in function that works with any iterable and returns a new sorted object

```
# Using sort() method
numbers = [3, 1, 4, 1, 5, 9, 2, 6]
numbers.sort()
print("Sorted list:", numbers)

# Using sorted() function
original = [3, 1, 4, 1, 5, 9, 2, 6]
new_sorted = sorted(original)
print("Original list:", original)
print("New sorted list:", new_sorted)
```

Reverse Sorting

Both sort() and sorted() accept a reverse parameter to sort in descending order.

```
numbers = [1, 5, 2, 8, 3, 9, 7]

# Descending order with sorted()
desc_numbers = sorted(numbers, reverse=True)
print("Descending order:", desc_numbers)
```

Exercise 12

Create two lists:

1. Use sort() to sort a list of your favorite numbers in ascending order
2. Use sort() to sort a list of your favorite numbers in descending order
3. Use sorted() to create a new sorted list from a list of fruit names

- Hint: Try ['banana', 'apple', 'orange', 'kiwi']

Basic Pattern Matching with Regular Expressions

Regular expressions (regex) are powerful tools for pattern matching in text. The re module in Python provides methods to work with regex patterns

```
import re

# String to search in
sequence = 'VSVLTMFRYAGWLDRLYMLVGTQLAAIIHGVALPLMMLI'

# Compile and search for pattern [ST]Q (S or T followed by Q)
pattern = re.compile('[ST]Q')
match = pattern.search(sequence)

if match:
    # Print 4 characters before and after the match
    print(sequence[match.start() - 4:match.end() + 4])
    print(match.group())
```

Exercise 13

Create a script that searches for the pattern [AG]T (A or G followed by T) in the sequence: ATGATCGTAGTCGATGCTAGCTAGCTAGT

Print the match and 2 characters before and after each match.

Finding Multiple Matches

The re module provides several methods for finding multiple matches:

- findall(): Returns a list of all matches
- group(): Returns the matched string
- span(): Returns start and end positions
- start(): Returns starting position
- end(): Returns ending position

```
import re

sequence = 'RQSAMGSNKSKPKDASQRRRSLEPAENVHGAGGGAFPASQRPSKP'
pattern = re.compile('R.[ST][^P]')  # R followed by any char, then S or T, then not P

# Using findall
matches = pattern.findall(sequence)
print("All matches:", matches)
```

Exercise 14

Write a script to find all occurrences of the pattern [AG]{2}[CT] (two A's or G's followed by C or T) in the sequence: AAGCTAAGTCGAGGCTTAGCTAGGC

Print both the matches.

Debugging:

Here's a code that someone learning coding typed but doesn't run. Can you figure out what they wanted to do with this code? Can you fix the code?

Sample code (erroneous):

```
price=20
money=30
if price<=money:
print("can buy")
else:
print("can not buy")
```

Exercise:

You left your computer for a moment and a grade 1 kid came along and scrambled everything. Now you've forgotten what you were trying to do as well. Can you make sense of the codes here and get a working algorithm?

Sample code (erroneous):

```
length=20
height=40
```

print("length>=height")
if length>=height:print("length<height")
else:

Data Visualization with Python

Matplotlib is a Python package, which provides a wide variety of plot types such as lines, bars, pie charts, and histograms.

Seaborn is a Python visualization library based on Matplotlib and provides a high-level interface for drawing attractive statistical graphics.

matplotlib.pyplot provides an implicit, MATLAB-like, way of plotting. It also opens figures on your screen, and acts as the figure GUI manager.

Line plot

```
from matplotlib.pyplot import figure, plot, savefig

xdata = [1, 2, 3, 4]
ydata = [1.25, 2.5, 5.0, 10.0]

figure()
plot(xdata, ydata)
```

https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html

```
import math
from matplotlib.pyplot import figure, plot, text, axis
figure()

xdata = [0.1 * i for i in range(100)]
ydata = [math.sin(j) for j in xdata]

plot(xdata, ydata, 'kd', linewidth = 1)
text(4.8, 0, "$y = sin(x)$", horizontalalignment = 'center', fontsize = 20)
axis([0, 3 * math.pi, -1.2, 1.2])
```

Histograms

```
from pickle import TRUE
from matplotlib.pyplot import plot, figure, title, xlabel, ylabel, hist, axis, grid

data = [1, 1, 9, 1, 3, 5, 8, 2, 1, 5, 11, 8, 3, 4, 2, 5]
n_bins = 5

figure()
num, bins, patches = hist(data, n_bins, density = TRUE, histtype = 'bar', facecolor = 'green', alpha = 0.75)

title('Histogram')
xlabel('value')
ylabel('frequency')
axis()
grid(True)
```

Bar plots

```
from matplotlib.pyplot import figure, title, xlabel, ylabel, xticks, bar, legend, axis

nucleotides = ["A", "G", "C", "U"]

counts = [    [606, 1024, 759, 398],
    [762, 912, 639, 591],
    ]

figure()
title('RNA nucleotides in the ribosome')
xlabel('RNA')
ylabel('base count')

x1 = [2.0, 4.0, 6.0, 8.0]
x2 = [x - 0.5 for x in x1]

xticks(x1, nucleotides)

bar(x1, counts[1], width = 0.5, color = "#cccccc", label = "E.coli 23S")
bar(x2, counts[0], width = 0.5, color = "#808080", label = "T.thermophilus 23S")

legend()
axis([1.0, 9.0, 0, 1200])
```

```
# scatterplot with error bars
x1 = [1.1, 1.2, 1.3, 1.4, 1.5]
y1 = [10, 15, 10, 15, 17]
err1 = (2, 3, 4, 1, 2)
width = 0.05
bar(x1, y1,width, color = 'r', yerr = err1, ecolor = "black")
```

boxplots

```
from matplotlib.pyplot import plot, boxplot
import matplotlib.pyplot as plt
import numpy as np


# Creating dataset
np.random.seed(10)
data = np.random.normal(100, 20, 200)
# Creating plot
boxplot(data)
```

Resources:

1. https://docs.python.org/3/tutorial/index.html
2. Lessons 3,4,6 of

https://www.youtube.com/playlist?list=PLqAsWscG_yDP6U_dJWhuSuQWMvv2XbJ9W

3. Matplotlib tutorials: https://matplotlib.org/stable/tutorials/index.html