Task2 Summary

Functions:

```
# funtion = a block of reusable code
#           place () after the function name to invoke it

#to define a function\

def function_name(parameter): #prameter are variables used
    #
    # some code
    #
    return some _data #statement used to end a function
                      #and send a result back to the caller

 #to invoke a function :
    function_name(arguments)
  #arguments can be 1. positional, 2. default, 3. keyword, 4.Arbitrary

```

Data structure:

general methods:

```
dir() #to find all available methods and attributes
help() #get description of all methods and attributes
len() #to find length of a collection
“element” in collection  #in operator used to find value within a collection returns Boolean
#All collections are iterable

```

- **Lists:** Uses [ ] they are ordered and changeable, can have duplicates

```
collection = ["1", "2", "3"]

# Set:
    collection[index] = “element”
# Add:
    # at the end:
        collection.append(“element”)
    # At given index:
        collection.insert(index, “element”)
# Delete:
    collection.remove(“element”)
# Sort:
    collection.sort() in alphabetical order
# Reverse:
        collection.reverse()
# Clear:
    collection.clear()
# Get index:
    collection.index(“element”)  cant be used for sets
# Count:
    collection.count(“element”)

```

To create 2d list: add one dimensional lists to a 2d list as elements

```
twod_collection = [oned_collection_1, oned_collection_2, oned_collection_3]

#To access 2d collections:
    twod_collection[row][column]

```

- **Dictionary:** a collection of {key:value} pairs, they are ordered and changeable, cant have duplicates

```
dictionary.get("key")  #To get value of key
dictionary.update({"key":"value"})  #To add/update
dictionary.pop("key")  #To remove a specific key-value pair
dictionary.popitem()  #To remove latest key value pair
dictionary.keys()  #To get keys
dictionary.values()  #To get values
dictionary.items()  #To get dictionary object

```

- **Tuple:** ( ) they are ordered and unchangeable, can have duplicates, Faster than lists

- **Sets:** { } they are unordered and immutable, but can add/remove elements.can't have duplicates

```
collection.pop() #removes a random element

```

Error handling:

```python
#events detected during executino that interrupt the flow of a program

#a widely used method is try-catch block

try:
  #
  # some code
  #
except Exception_name as e:  #catches error
  print(e)  #display error discription
  print("Something went wrong")
else:
  # some code will execute if there is no errors
finally:
  # whether or not an exception is caught always execute this code

```

Files input/output:

- ### File detection:

```python
import os  # provides functions for interacting with the operating system

#  file detection
path = "C:\\Users\\John\\Desktop\\example.txt" #string with file path

if os.path.exists(path):
  # checks if location exists
  print("This location exists")
  if os.path.isfile(path):
    #checks if path is file
    print("this is a file")
  elif os.path.isdir(path):
    #checks if path is directory
    print("This is a directory")
else:
  print("This location doesn't exists")

```

- **Input:**

```
#  File as Input

# with open('example.txt') as file:
# this is used when file is in the same project holder

with open('C:\\Users\\John\\Desktop\\example.txt') as file:
  print(file.read())

```

- **Output:**

```
#  File as Output

text = "Some text \n next line "

with open('example.txt', 'w') as file:  # this overwrites the file
  file.wirte(text)

#-------------------------------------------------------------

with open('example.txt', 'a') as file:  # this appends to file
  file.wirte(text)

```

Random numbers:

using random module to generate random instances

```
import random

random.randint(1,6) #return a random number in a range

random.random() #return a random floating point number between 0 and 1

random.choice(collection) #returns a random element from a collection

random.shuffle(collection) #returns a random sequence of the collection

```