BB_DS_prework - Python exercises

Exercise 1

- Add 5 and 225 and print the result. Create 3 different variables
- Divide 7 with 3
- Divide 7 with 3 and get only the integer part
- Find the remainder when dividing 7 with 3

```
# Add 5 and 225
a = 5
b = 225
result =0
print(result)

# Divide 7 with 3
print(7/3)

# Divide 7 with 3 get only integer part
print(int(7/3))
print(7 // 3)

# Find remainder when dividing 7 with 3
print(7%3)

```

```
0
2.3333333333333335
2
2
1
```

Exercise 2

- Create a string with 40 dashes using the “-” symbol only once

```
# Create string with 40 dashes
print('-'* 40)
```

```
----------------------------------------
```

Exercise 3

- Clean the following string: "John=is-a-great web

```
# "John=is-a-great web developer\t    "
```

- (it should look like “John is a great web developer“)

```
string = "John=is-a-great web developer\t    "
clean_string= string.strip("=\t ")
clean_string=string.replace('-',' ').replace('=',' ')
print(clean_string)
```

```
John is a great web developer
```

Exercise 4

Use input function and string formatting to create the following program. Ask the user to give: first name, last name, age and occupation. Then use this information to create the following sentence: “New user {first name} {last name}, {age} years old, {occupation}, has been registered in the system”

```
#Use input function and string formatting
FirstName = input ("Please enter your first name: ")
LastName =  input ("Please enter your lsat name: ")
age =  input ("Please enter your age: ")
occupation =  input ("Please enter your occupation: ")
print(f"New user {FirstName} {LastName}, {age} years old, {occupation}, has been registered in the sy
```

```
Please enter your first name: mike
Please enter your lsat name: Ford
Please enter your age: 22
Please enter your occupation: teacher
New user mike Ford, 22 years old, teacher, has been registered in the sysmste
```

Exercise 5

- Make a list of the first ten multiples of ten (10, 20, 0... 90, 100). Print out your list.

```
#First ten multiples of ten

multiple =[]
for n in range(1, 11):
  multiple.append(n *10)
print(multiple)

```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Exercise 6

- Multiply all elements of the following list: x = [3,5,23,6,7]

```
#Multiply list elements
import numpy
x = [3,5,23,6,7]
x = numpy.prod(x)
print(x)
```

```
14490
```

Exercise 7

- Write a program that takes as input an integer n and calculates the sum of all integers between 1 and n.

```
def calculate_sum(n):
  """
  This function calculates the sum of all integers between 1 and n.

  Args:
      n: The upper limit of the range (inclusive).

  Returns:
      The sum of all integers between 1 and n.
  """

  # Use the formula for the sum of an arithmetic series
  return n * (n + 1) // 2

# Example usage
n = 10
sum_of_numbers = calculate_sum(n)
print(f"The sum of all integers between 1 and {n} is: {sum_of_numbers}")

```

```
The sum of all integers between 1 and 10 is: 55
```

Exercise 8

- Find the second largest number in the following list: x = [44,32,65,77,12,86]

```
# Second largest number in list
x = [44,32,65,77,12,86]
x.sort()
print(x)
second_largest_num = x[-2]
print(second_largest_num)
```

```
[12, 32, 44, 65, 77, 86]
77
```

Exricese 9

- Join the elements of the following list and add a - between them: fruits = ['Apple', 'Grapes', 'Berry', 'Orange']

```
#Join elements of list with '-'
fruits = ['Apple', 'Grapes', 'Berry', 'Orange']
joined_fruits = "-".join(fruits)
print(joined_fruits)
```

```
Apple-Grapes-Berry-Orange
```

Exricese 10

- Create a new list without duplicates from the given list: x = [32,3,5,4,4,3,2,6,4,5,6,7,8]

```
# Remove duplicated
x = [32,3,5,4,4,3,2,6,4,5,6,7,8]
lst = list(set(x))
print(lst)
```

```
[32, 2, 3, 4, 5, 6, 7, 8]
```

Exercise 11

- Given the following list, use indexing to grab the word Data Science lst = [1,2,[3,4],[5,[100,200,['Data Science']],23,11],1,7]

```
# List indexing
lst = [1,2,[3,4],[5,[100,200,['Data Science']],23,11],1,7]
x = lst[3][1][2][0]
print(x)
```

```
Data Science
```

Exercise 12

- From the two following lists: x = [2,4,5,6,7,1,6,8], y = [3,4,5,6,0,3,9] create a new list with the non common elements.

```
# non common elememts = nce
x = [2,4,5,6,7,1,6,8]
y = [3,4,5,6,0,3,9]

set_x = set(x)
set_y = set(y)

# Find non-common elements
nce = list(set_x ^ set_y)

print(nce)
```

```
[0, 1, 2, 3, 7, 8, 9]
```

Exercise 13

- Write a program that replaces a string in a list with 0. Example: x = [4,67,8, 'None', 32, 'Missing', 21]

```
# Replace string with 0
x = [4,67,8, 'None', 32, 'Missing', 21]

for n in range(len(x)):
  if x[n] == 'None' or  x[n] == 'Missing':
    x[n] = 0
print(x)
```

```
[4, 67, 8, 0, 32, 0, 21]
```

Exercise 14

- A list cleaner: create a program that detects if an element in the list is an inner list and then places these elements in the primary list (right where the element was).

```
def flatten_list(list1):
  """
  This function flattens a list by removing any nested lists and adding their elements to the primary

  Args:
      list1: The list to be flattened.

  Returns:
      A new list containing the elements of the original list and its nested lists.
  """

  flat_list = []
  for item in list1:
    if isinstance(item, list):
      flat_list.extend(flatten_list(item))  # Recursively flatten nested lists
    else:
      flat_list.append(item)
  return flat_list

# Example usage
original_list = [1, [2, 3], 4, [5, 6, [7]]]
flattened_list = flatten_list(original_list)

print(f"Original list: {original_list}")
print(f"Flattened list: {flattened_list}")

```

```
Original list: [1, [2, 3], 4, [5, 6, [7]]]
Flattened list: [1, 2, 3, 4, 5, 6, 7]
```

Exercise 15

- Create a new list without duplicates from the given list (use sets): x = [32,3,5,4,4,3,2,6,4,5,6,7,8]

```
# it is the same excerise with ex.10
# Remove duplicated
x = [32,3,5,4,4,3,2,6,4,5,6,7,8]
lst = list(set(x))
print(lst)
```

```
[32, 2, 3, 4, 5, 6, 7, 8]
```

Exercise 16

- From the two following lists: x = [2,4,5,6,7,1,6,8], y = [3,4,5,6,0,3,9] create a new list with the non common elements (use sets).

```
# it is same exercise with 12

# non common elememts = nce
x = [2,4,5,6,7,1,6,8]
y = [3,4,5,6,0,3,9]

set_x = set(x)
set_y = set(y)

# Find non-common elements with XOR operator
nce = list(set_x ^ set_y)

print(nce)
```

```
[0, 1, 2, 3, 7, 8, 9]
```

Exercise 17

- Given the following dictionary use indexing to grab the word cylinder d = {'section':{'production':['wheel','tyre', {'engine':['pistons','valves','cylinder']}, 'door']}, 'development':'gearbox'}

```
#disctionary indexing
d = {
    'section': {
        'production': ['wheel', 'tyre', {'engine': ['pistons', 'valves', 'cylinder']}, 'door']
    },
    'development': 'gearbox'
}

dictioanry = d['section']['production'][2]['engine'][2]
print(dictioanry)
```

```
cylinder
```

Exercise 18

```
careers = ["programmer", "doctor", "teacher", "truck driver"]

# Find the index of "programmer"
programmer_index = careers.index("programmer")
print(f"'programmer' is at index {programmer_index} in the list.")

# Check if "doctor" is in the list using if and in
if "doctor" in careers:
  print("'doctor' is in the list.")

# Add "engineer" to the list using append()
careers.append("engineer")
print(f"List after adding 'engineer': {careers}")

# Add "musician" to the beginning using insert()
careers.insert(0, "musician")
print(f"List after adding 'musician' at the beginning: {careers}")

# Print the list in different orders

# Alphabetical order
print("\nList in alphabetical order:")
print(sorted(careers))

# Original order
print("\nList in original order:")
print(careers)

# Reverse alphabetical order
print("\nList in reverse alphabetical order:")
print(sorted(careers, reverse=True))

```

```
'programmer' is at index 0 in the list.
'doctor' is in the list.
List after adding 'engineer': ['programmer', 'doctor', 'teacher', 'truck driver', 'engineer']
List after adding 'musician' at the beginning: ['musician', 'programmer', 'doctor', 'teacher', 'truc

List in alphabetical order:
['doctor', 'engineer', 'musician', 'programmer', 'teacher', 'truck driver']

List in original order:
['musician', 'programmer', 'doctor', 'teacher', 'truck driver', 'engineer']

List in reverse alphabetical order:
['truck driver', 'teacher', 'programmer', 'musician', 'engineer', 'doctor']
```

Exercise 19

Alphabet Slices.

- Store the first ten letters of the alphabet in a list.
- Use a slice to print out the first three letters of the alphabet.
- Use a slice to print out any three letters from the middle of your list.
- Use a slice to print out the letters from any point in the middle of your list, to the end.

```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

#store  the first 10 letters
lst = alphabet[:10]
print(lst)

# slice tje fisrt 3 leets
lst1 = lst[:3]
print(lst1)

# slice any 3 letters from the middle of the list
lst2 = lst [4:7]
[print(lst2)]

# slice any letter from the middle to the end
lst3 = lst[5:]
print(lst3)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
['a', 'b', 'c']
['e', 'f', 'g']
['f', 'g', 'h', 'i', 'j']
```

Exercise 20

- Create a function that returns true when the parameter passed is a string and false otherwise

```
def is_string(input):
    return isinstance(input, str)

print(is_string("Hello, World!"))  # Output: True
print(is_string(12345))  # Output: False

```

```
True
False
```

Exercise 21

- Check if the word holiday is in a given string. Spring example: 'St. Moritz is a nice location to visit over Christmas holidays!'

```
#Check for word in string
string = 'St. Moritz is a nice location to visit over Christmas holidays!'
word = 'holiday'
if word in string:
  print('it is present in the string')
else:
  print('it is not found in the string')
```

```
it is present in the string
```

Exercise 22

- Take a random number and convert it to a reverse-sorted list of digits

```
import random

# Generate a random number
number = random.randint(1, 999)  # Adjust the range as needed

# Convert the number to a string and then to a list of digits
digits = list(str(number))

# Sort the digits in reverse order
digits.sort(reverse=True)

# Print the result
print(f"Original number: {number}")
print(f"Reverse-sorted digits: {digits}")

```

```
Original number: 979
Reverse-sorted digits: ['9', '9', '7']
```

Exercise 23

```
from itertools import product


data = {'part 1': ['a', 'b'], 'part 2': ['c', 'd']}

# Get all combinations of choices from each part
all_combinations = product(*[data[part] for part in data])

# Print each combination by joining its elements
for combo in all_combinations:
  print("".join(combo))
```

```
ac
ad
bc
bd
```

Exercise 24

- Find how much is the average bill and how much is the total cost over the whole year.

```
total_cost = 0
average_cost = 0
electricity_bill = {
        'January': 232,
        'February': 245,
        'March': 267,
        'April': 223,
        'May': 257,
        'June': 284,
        'July': 243,
        'August': 120,
        'September': 245,
        'October': 278,
        'November': 345,
        'December': 326,
}
total_cost = sum(electricity_bill.values())
average_cost = total_cost /len(electricity_bill)
print("{:.3f}".format(average_cost))
```

```
255.417
```

Exercise 25

```
#convert grades from numbers to letters
grades = [91, 64, 47, 82, 67, 96]
letter_grades = []

for grade in grades:
  if grade >= 90:
    letter_grades.append('A')
  elif grade >= 80:
    letter_grades.append('B')
  elif grade >= 70:
    letter_grades.append('C')
  elif grade >= 60:
    letter_grades.append('D')
  else:
    letter_grades.append('F')

print(letter_grades)

```

```
['A', 'D', 'F', 'B', 'D', 'A']
```

Exercise 26

```
temperature = {
    'June': [25, 25, 26, 27, 25, 25, 24, 27, 28, 28, 31, 32, 33],
    'July': [34, 34, 36, 39, 39, 38, 39, 37, 39, 41, 41, 39, 37],
    'August': [37, 37, 36, 37, 35, 35, 34, 37, 38, 34, 32, 33, 31],
}

# Find the month with the highest average temperature
highest_avg_temp = None
highest_avg_temp_value = 0

for month, days in temperature.items():
    avg_temp = sum(days) / len(days)
    if avg_temp > highest_avg_temp_value:
        highest_avg_temp = month
        highest_avg_temp_value = avg_temp

# Find the month with the lowest average temperature
lowest_avg_temp = None
lowest_avg_temp_value = float('inf')

for month, days in temperature.items():
    avg_temp = sum(days) / len(days)
    if avg_temp < lowest_avg_temp_value:
        lowest_avg_temp = month
        lowest_avg_temp_value = avg_temp

print(f"The month with the hottest day is {highest_avg_temp} with an average temperature of {highest_
print(f"The month with the coldest day is {lowest_avg_temp} with an average temperature of {lowest_av

```

```
The month with the hottest day is July with an average temperature of 37.92 degrees.
The month with the coldest day is June with an average temperature of 27.38 degrees.
```

Exercise 27

```
# Make a list of the squares of the even numbers between 1 to 10
# empty list
even_squares = []

# Iterate through numbers from 1 to 10
for num in range(1, 11):
  # Check if the number is even
  if num % 2 == 0:
    # If even, square the number and append it to the list
    even_squares.append(num * num)

# Print the list of squares
print(even_squares)
```

```
[4, 16, 36, 64, 100]
```

Exercise 28

```
def find_even_numbers(number_list):
  """
  This function takes a list of numbers and returns a list containing only the even numbers.

  Args:
  number_list: A list of numbers.

  Returns:
  A list containing only the even numbers from the input list.
  """

  even_numbers = []
  for num in number_list:
    if num % 2 == 0:
      even_numbers.append(num)
  return even_numbers

# Example usage
number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_list = find_even_numbers(number_list)
print(even_list)

```

```
[2, 4, 6, 8, 10]
```

Exercise 29

```
def product_of_list(list_of_numbers):
  """
  This function takes a list of integers as input and returns the product of all of the elements in t

  Args:
      list_of_numbers: A list of integers.

  Returns:
      The product of all the elements in the list.
  """

  product = 1
  for num in list_of_numbers:
    product = product * num
  return product

# Example usage
number_list = [1, 2, 3, 4, 5]
list_product = product_of_list(number_list)
print(list_product)

```

```
120
```

Exercsie 30

```
# Write a function that takes in a list and gives only the unique elements.
def find_unique_elements(list_of_items):
  """
  This function takes a list and returns a list containing only the unique elements.

  Args:
      list_of_items: A list of any data type.

  Returns:
      A list containing only the unique elements from the input list.
  """

  unique_elements = []
  for item in list_of_items:
    if item not in unique_elements:
      unique_elements.append(item)
  return unique_elements

# Example usage
mixed_list = [1, 9, 5, 3, 4, 5, 11, 1]
unique_list = find_unique_elements(mixed_list)
print(unique_list)

```

```
[1, 9, 5, 3, 4, 11]
```

Exercise 31

```
#Write a function to calculate the average of a list of numbers
def calculate_average(number_list):
  """
  This function takes a list of numbers and returns the average of the elements in the list.

  Args:
      number_list: A list of numbers.

  Returns:
      The average of the elements in the list.
  """

  if not number_list:
    return None  # Handle empty list case

  total_sum = sum(number_list)
  average = total_sum / len(number_list)
  return average

# Example usage
number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list_average = calculate_average(number_list)
print(list_average)

```

```
5.5
```

Exercise 32

```
order_list = [
    'fries', 'burger', 'eggs', 'pasta', 'pizza', 'schnitzel', 'salad', 'water',
    'soda', 'wine', 'eggs', 'pasta', 'pizza', 'schnitzel', 'salad', 'water',
    'soda', 'wine', 'pizza', 'schnitzel', 'soda', 'wine', 'lemonade', 'steak',
    'pasta', 'salad', 'fries', 'burger', 'water', 'burger'
]

# dictionary to store the order counts
order_counts = {}

# Iterate through the order list
for order in order_list:
  # Check if the order exists in the dictionary
  if order in order_counts:
    # Increment the count for the existing order
    order_counts[order] += 1
  else:
    # Add the order to the dictionary with a count of 1
    order_counts[order] = 1

# Print the order counts
print(order_counts)
```

```
{'fries': 2, 'burger': 3, 'eggs': 2, 'pasta': 3, 'pizza': 3, 'schnitzel': 3, 'salad': 3, 'water': 3,
```

Exercise 33

```
customers = {
    'customer 1': {'name': 'John Smith', 'city': 'New York', 'nr_or_purchases': 3, 'items': ['coffee'
    'customer 2': {'name': 'Rebeca Collins', 'city': 'New York', 'nr_or_purchases': 1, 'items': ['sug
    'customer 3': {'name': 'Edward Matthews', 'city': 'Boston', 'nr_or_purchases': 4, 'items': ['bana
    'customer 4': {'name': 'Maria Simmons', 'city': 'Boston', 'nr_or_purchases': 3, 'items': ['ham',
}
```

Exercise 34

```
customers['customer 3']['nr_or_purchases'] += 1
```

```
def count_letters(sentence):
    """
```