Using BRef-01: Chapter 09 answer and experiment the following questions:

1. **What is a function in Python?**
   A function is a piece of code, separate from all others. It can take any number and type of input parameters and return any number and type of output results.
2. **What are the main elements of a Python function? Define a simple function that does nothing.**
   Def do_norhing():
   pass
3. **How can a function be used (called)?**
   do_nothing()
4. **How can one return the result of a function?**
   Def function():
   return
5. **What are the arguments and/or parameters?**

Exercises:

1. **Explain in your own words the difference between arguments and parameters.**
   Parameters are placeholders for arguments.
2. **Create a function that just prints the word hello. Call the function and run your program. Where the function is *defined*? Where is it *called*?**

```
def greet():
print("Hello")
greet()
```

3. **Create a function that takes a text as an argument. The function prints the text it receives. Call the function and run your program.**

```
def text(word):
        print(f"{word}")
text("Telmo")
```

4. **Create a function that takes two numbers as argument. The function adds the numbers together and returns the results. Call the function and run your program.**

```
def meth(a, b):
        print(a + b)
meth(32, 21)
```

5. **Create two functions, each takes a number as argument. The first one returns the number multiplied by 2 and returns it. The second multiplies it by 10 and returns it. Calling both functions add the two returned numbers together and print it. Run your program and check the results.**

```
def int1(a):
        return a // 2

def int2(b):
        return b * 10

print(int1(2) + int2(7))
```

6. **Create two functions. One that prints hello, the other prints bye. Ask the user to input a number, if the number is higher than 10, call the first function. If the number if lower or equal to 10, call the second function. Test your program.**

```
def hello():
        print("Hello")

def bye():
        print("Bye")

number = int(input("Give number: "))

if number >= 10:
        hello()
else:
        bye()
```

7. **Create two functions. One that prints hello, the other prints bye. The first functions calls the second one after printing. Call the first function.**

```
def hello():
        print("Hello")
        bye()

def bye():
        print("Bye")

hello()
```

Step-02: Tuples.

Goals:

After taking this step, you will be able to:

1. interpret and implement Python programs using tuples: creating, unpacking, modifying, combining two tuples, iterating over a tuple.

What to Learn?

1. Using **BRef-01: Chapter 07** answer and experiment the following questions:
    1. **What is a tuple in Python and how is it defined?**
       A tuple is a sequence structure like lists.
       Defined like *tuple = ()*
    2. **How can one combine and compare two (or more) tuples?**
       With "+" for example: ("Groucho",) + ("Chico", "Harpo")
       Compare can be done with the operators
    3. **How can one iterate over the elements of a tuple?**
       With a for loop
    4. **How is a tuple modified?**

       It isn't

Exercises:

1. **You can create a tuple with mixed types in it, for example texts and numbers. Can you think of an advantage and a disadvantage of doing this? Implement your example.**

2. **Create a tuple with three numbers in it. Unpack the tuple into three different variables. Print the last one.**

```
numbers = (1, 2, 3)
a, b, c = numbers
print(c)
```

3. **Create a tuple with two numbers in it, create a second tuple with two texts in it. Add them together into a new tuple. Print the new tuple.**

```
numbers = (1, 2)
letters = ("gaming", "space")
print(numbers + letters
```

4. **Create a tuple with three numbers in it. Use a for loop to iterate over each value. Multiply each value by 2 and print each result.**

```
numbers = (1, 2, 3)
for i in numbers:
          i *= 2
          print(i)
```

5. **Create a function that returns a tuple containing three texts. Call the function. Unpack the tuple into variables. Print the variables.**

```
def tuples():
    x = ("Big", "Booty", "Latina")
    a, b, c = x
    print(a, b, c)
tuples()
```

Step-03: Lists.

Goals:

```
After taking this step, you will be able to:
        1. interpret and implement Python programs using lists: defining,
offset, slicing, adding new element, modifying an element.
```

What to Learn?

1. Using **BRef-01: Chapter 07** answer and experiment the following questions:
    1. **What is a list in Python and how is it defined?**
       A list is also a sequence structure, like tuples.
       It's defined with list = []
    2. **What is the result of *split()* on a string?**
       It will make a list after each "space".
    3. **There are two ways to get items from a list: offset and slice. What are the pros / cons of each? Experiment with some examples.**
       idk
    4. **How can you add new elements to a list?**
       With append.("element")
    5. **How can you modify elements of a list?**
       With the offset[1]
    6. **How can one iterate over the elements of a list?**
       Iterate with for and in
    7. ***Mutability* is one of the main differences between a tuple and a list. Elaborate this with some example.**
       It means that I can add or take way things from the list

Exercises:

1. Define a list of integers representing scores of a game. Write a program that prints out maximum and minimum of the `scores`.

```
scores = [1,2,3,4,5,6,7,8,9,10]
print(scores[0], scores[-1])
```

2. Extend your program from the previous exercise such that it prints two largest and two smallest elements of the `scores`.

```
scores = [1,2,3,4,5,6,7,8,9,10]
print(scores[0:2], scores[8:10])
```

3. Write a program that asks the user to enter a list of integers. Do the following:
    - Print the total number of items in the list.
    - Print the last item in the list.
    - Print the list in reverse order.
    - Print Yes if the list contains a 5 and No otherwise.
    - Print the number of fives in the list.
    - Remove the first and last items from the list, sort the remaining items, and print the result.
    - Print how many integers in the list are less than 5.

```
numbers = input("List of integers: ").split()

for i in range(len(numbers)):
    numbers[i] = int(numbers[i])

print(numbers)
print(numbers[-1])

numbers.reverse()
print(numbers)
numbers.reverse()

for x in numbers:
    if not x % 5:
        print("Yes")
        break
    else:
        print("No")

count = 0
for y in numbers:
    if not y % 5:
        count += 1
    else:
        continue
print(count)

nocount = 0
for p in numbers:
    if p % 5:
        nocount += 1
    else:
        continue
print(nocount)
```

4. Write a program that generates a list of 20 random numbers between 1 and 100.
    - Print the list.
    - Print the average of the elements in the list.
    - Print the largest and smallest values in the list.
    - Print the second largest and second smallest entries in the list.
    - Print how many even numbers are in the list.

```
list = []
for x in range(1, 20):
    random_number = random.randrange(1,101)
    list.append(random_number)
print(list)

def average(list):
    return sum(list) / len(list)

print(round(average(list)))

list.sort()
def largest(list):
    return max(list)
def smallest(list):
    return min(list)

print(largest(list))
print(smallest(list))

list.sort()
print(list)
second_smallest = list[1]
second_largest = list[-2]
print(second_largest)
print(second_smallest)

even_count = 0
for q in list:
    if q % 2 == 0:
        even_count += 1
    else:
        continue
print(even_count)
```

5. Start with the list `[8,9,10]`. Do the following:
    - Set the second entry (index 1) to 17
    - Add 4,5, and 6 to the end of the list
    - Remove the first entry from the list
    - Sort the list
    - Double the list
    - Insert 25 at index 3
    - The final list should equal `[4,5,6,25,10,17,4,5,6,10,17]`

```
list=[8,9,10]
print(list)
list.insert(1, 17)
print(list)
list.append(4)
list.append(5)
list.append(6)
print(list)
list.pop(0)
print(list)
list.sort()
print(list)
copy = list.copy()
list = list + copy
print(list)
list.insert(3, 25)
print(list)
```

6. Create the following lists using a loop.
    - A list consisting of the integers 0 through 49

```
count = 0
list = []
for x in range(0, 50):
    list.append(count)
    count += 1
print(list)
```

    - A list containing the squares of the integers 1 through 50.

```
count = 0
list = []
for x in range(1, 51):
    square = count * count
    list.append(square)
    count += 1
print(list)
```

- The list `['a','bb','ccc','dddd', . . . ]` that ends with 26 copies of the letter z.

```
list = [chr(i) for i in range(65, 90)]
empty = []
print(list)
count = 1

for i in list:
    replace = i * count
    count += 1
    empty.append(replace)
print(empty)
```

7. Write a program that takes any two lists L and M of the same size and adds their elements together to form a new list N whose elements are sums of the corresponding elements in L and M. For instance, if `L=[3,1,4]` and `M=[1,5,9]`, then N should equal `[4,6,13]`.

```
L = [3,1,4]
M = [1,5,9]
N = []

for L, M in zip(L, M):
    new = L + M
    N.append(new)
print(N)
```

Learning Activities:

Code Analysis

1. Analyse the two given codes below without executing them. What will be the result of the programs?

```
a_tuple = ('Never', 'gonna', 'give', 'you', 'up')
counter = 0
for x in a_tuple:
    if x[0] ==  'g':
        counter = counter + 1
    else:
        counter = counter + 2
print(counter)
```

It’s gonna print (8)

```
def do_something(x):
    rtuple = x,
    for i in range(2,11):
        rtuple = rtuple + ((x*i),)
    return rtuple
print(do_something(6))
```

It’s gonna print 6, 12, etc… until 60

```
def process_strings(strings):
    processed_strings = []
    for string in strings:
        processed_string = ""
        for char in reversed(string):
            processed_string += char
        processed_strings.append(processed_string)
    return processed_strings

def main():
    names = ["Alice", "Bob", "Charlie", "Dave"]
    processed_names = process_strings(names)
    for name in processed_names:
        print(name)

main()
```

It’s gonna show the names reversed.

Supporting Topics

Testing: Input/Output

Implementing a *correct* program is not easy. Programs always tend to have bugs. **Input/Output Test** is one of the common techniques to test the correctness of a program. The main idea is to *design* certain inputs, run the program with the inputs and check if the *produced* output satisfies *expected* output.

1. **Make a brief research about input/output testing with some examples. Make a summary of the techniques with the examples.**
   Different techniques include *Black Box Testing*, *White Box Testing*, *Functional Testing* and *Non Functional Testing.*
2. **Choose one of the problems you have solved for this week.**
   - **Certainly there are some normal inputs for which your program is supposed to provide a correct output. Think about three different *expected* inputs and see if the output is correct.**
   - **There are some inputs that are not expected. For example, if your program is supposed to receive an integer between 0 and 10, then a negative integer is not expected. Choose three different unexpected inputs and test your program. If your program crashes, then fix your program such that provides a proper message instead of crashing.**
   - **Usually, users unexpectedly make mistakes when trying to give an input. For example, if they try to enter a digit between 0 and 10, mistakenly they press `6;`. Check if your program is resilient for such errors.**

Sources:

- https://docs.replit.com/teams-edu/input-output-testing
- https://www.browserstack.com/guide/software-testing-techniques