```
# Functions Assignment


#Theory Questions Answers


#Ans.1
#If the line of code is called on an object, it is a method. If it is not called on an ob
#Example>> the add_numbers() function is not associated with any object, while the upper(


#Ans.2
#A parameter is the variable listed inside the parentheses in the function definition.
#An argument is the value that are sent to the function when it is called.

def greet_person(name):  # 'name' is a parameter
    """This function greets a person by their name."""
    print(f"Hello, {name}!")

greet_person("Alice")  # "Alice" is an argument
greet_person("Bob")    # "Bob" is an argument
```

```
Hello, Alice!
Hello, Bob!
```

```
#Ans.3
#Define a function with the def keyword, then write the function identifier (name) follow

def greet(name):
  """This function greets to the person passed in as a parameter"""
  print("Hello, " + name + "!")

greet("Alice")  #Calling the function with the argument "Alice"
```

```
Hello, Alice!
```

```
#Ans.4
#The Python return statement marks the end of a function and specifies the value or value
#Return statements can return data of any type, including integers, floats, strings, list

def function_name(parameter1, parameter2):
    # Function body
    return return_value


#Ans.5
#iterable>>
#An iterable is any python object/sequential structure/data structure that is capable of
#permitting it to be iterated over in a for loop
#example >> list, tuples, etc.

#iteration>>
```

```
#iteration is a process of looping through the elements for an iterable (list, string,) u
#The process of returning element one bby one iteration


#Ans.6
#Generators are a simple way to create iterators using functions and the yield keyword in

def square_number_generators(n):
    for i in range(n):
        yield i**2

square_number_generators(10)
```

```
<generator object square_number_generators at 0x0000019DD5998520>
```

```
#Ans.7
#Advantages of Generators:
#Memory Efficiency
#Lazy Evaluation
#Flexibility
#Reusability


#Ans.8
#Lambda functions are small, anonymous functions defined using the lambda keyword. They a

square = lambda x: x*x
print(square(5))
```

```
25
```

```
#Ans.9
#map >> executes a specified function for each of item of an iterable
#syntax >> map(func,*iterables)



#Ans.10
#Map >> The map function applies a given function to all items in an input iterable (like
#Example
l = ["1", "2", "3"]
list(map(lambda x: int(x), l))

#Reduce >> The reduce function from the functools module applies a given function cumulat
#Example
words =  ["Data", "Science", "courses"]
reduce(lambda x, y: x+ " "+y, words)

#Filter >> The filter function constructs an iterator from elements of an iterable for wh
#Example
l1 = [-1, -2, 3, 4, 5]
list(filter(lambda x: x < 0, l1))
```

```
#Ans.11
```

Start coding or generate with AI.

```
#Practical Questions


#Ans.1
def sum_of_even_numbers(numbers):
    return sum(num for num in numbers if num % 2 == 0)


#Ans.2
def reverse_string(text):
    return text[::-1]


#Ans.3
def square_number(n):
    result = []
    for i in n:
        result.append(i ** 2)
    return result
square_number([5, 10, 15])
```

```
[25, 100, 225]
```

```
#Ans.4
import math

def is_prime(n: int) -> bool:
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False
    limit = int(math.isqrt(n))  # or math.sqrt(n)
    for i in range(3, limit + 1, 2):  # skip even numbers
        if n % i == 0:
            return False
    return True

# Example: list primes from 1 to 200
primes = [n for n in range(1, 201) if is_prime(n)]
print(primes)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
```

```
#Ans.5
def fib(n):
    a = 0
```

```
    b = 1
    for i in range (n):  #if n=5, 0, 1, 2, 3, 4
        yield a
        a, b = b, a+b

f = fib(10000000000000)
f
```

```
<generator object fib at 0x7a1f592b8130>
```

```
#Ans.6
def powers_of_two(max_exponent):
    for exp in range(max_exponent + 1):
        yield 2 ** exp

powers_of_two(2)
```

```
<generator object powers_of_two at 0x7a1f59bdd9a0>
```

```
#Ans.7
def read_file_line_by_line(file_path):
    with open(file_path, 'r') as file:
        for line in file:
            yield line.rstrip('\n')


#Ans.8
data = [(1, 5), (3, 2), (2, 8), (4, 1)]

sorted_data = sorted(data, key=lambda x: x[1])
print(sorted_data)
```

```
[(4, 1), (3, 2), (1, 5), (2, 8)]
```

```
#Ans.9
def celsius_to_fahrenheit(celsius):
  return (celsius * 9/5) + 32

def convert_temperatures(celsius_list):
    fahrenheit_list = list(map(celsius_to_fahrenheit, celsius_list))
    return fahrenheit_list

celsius_temps = [0, 10, 20, 30, 40]
fahrenheit_temps = convert_temperatures(celsius_temps)
print(f"Celsius temperatures: {celsius_temps}")
print(f"Fahrenheit temperatures: {fahrenheit_temps}")
```

```
Celsius temperatures: [0, 10, 20, 30, 40]
Fahrenheit temperatures: [32.0, 50.0, 68.0, 86.0, 104.0]
```

```
#Ans.10
string = "Data Analytics"
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
```

```
result = ""
for i in range(len(string)):
    if string[i] not in vowels:
        result = result + string[i]

print("\nAfter removing Vowels: ", result)
```

```
 After removing Vowels:  Dt nlytcs
```

```
#Ans.11
orders = [
    [34587, "Learning Python, Mark Lutz", 4, 40.95],
    [98762, "Programming Python, Mark Lutz", 5, 56.80],
    [77226, "Head First Python, Paul Barry", 3, 32.95],
    [88112, "Einführung in Python3, Bernd Klein", 3, 24.99]
]

calculate_order_total = lambda order: (
    order[0],
    (order[2] * order[3] + 10) if (order[2] * order[3]) < 100 else (order[2] * order[3
)

result = list(map(calculate_order_total, orders))
print(result)
```

```
[(34587, 163.8), (98762, 284.0), (77226, 108.85000000000001), (88112, 84.97)]
```

Start coding or generate with AI.