Function Basic Assignment

July 23, 2024

[1]:
```
# Que 1. Write a Python function that takes a list of numbers as input and␣
 ↪returns the sum of all even numbers in the list.
def sum_of_even_numbers(numbers):
  even_sum = 0
  for num in numbers:
    if num % 2 == 0:
      even_sum += num
  return even_sum
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
result = sum_of_even_numbers(my_list)
print(result)
```

```
42
```

[2]:
```
# Que 2. Create a Python function that accepts a string and returns the reverse␣
 ↪of that string.
def reverse_string(string):
  return string[::-1]
my_string = "I am a PW Student"
reversed_string = reverse_string(my_string)
print(reversed_string)
```

```
tnedutS WP a ma I
```

[3]:
```
# Que 3.Implement a Python function that takes a list of integers and returns a␣
 ↪new list containing the squares of each number.
my_list = [1, 2, 3, 4, 5, 6]
my_list
```

[3]:
```
[1, 2, 3, 4, 5, 6]
```

[4]:
```
def square(my_list):
    squared_list = []
    for i in my_list:
        squared_list.append(i ** 2)
    return squared_list
```

[5]:
```
square(my_list)
```

[5]: [1, 4, 9, 16, 25, 36]

[7]:
```python
# Que 4. Write a Python function that checks if a given number is prime or not from 1 to 200.
def is_prime(n):
    if n <= 1:
        return False  # 1 and numbers less than 1 are not prime numbers
    elif n == 2:
        return True   # 2 is a prime number
    elif n % 2 == 0:
        return False  # Any even number greater than 2 is not prime
    else:
        # Check for odd factors from 3 up to the square root of n
        sqrt_n = int(n**0.5) + 1
        for i in range(3, sqrt_n, 2):
            if n % i == 0:
                return False  # n is divisible by i, hence not prime
        return True  # n is a prime number

# Test the function for numbers from 1 to 200
for number in range(1, 201):
    if is_prime(number):
        print(f"{number} is a prime number")
```

```
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
53 is a prime number
59 is a prime number
61 is a prime number
67 is a prime number
71 is a prime number
73 is a prime number
79 is a prime number
83 is a prime number
```

```
89 is a prime number
97 is a prime number
101 is a prime number
103 is a prime number
107 is a prime number
109 is a prime number
113 is a prime number
127 is a prime number
131 is a prime number
137 is a prime number
139 is a prime number
149 is a prime number
151 is a prime number
157 is a prime number
163 is a prime number
167 is a prime number
173 is a prime number
179 is a prime number
181 is a prime number
191 is a prime number
193 is a prime number
197 is a prime number
199 is a prime number
```

[8]:
```python
# Que 5.Create an iterator class in Python that generates the Fibonacci␣
 ↪sequence up to a specified number of terms.
class FibonacciIterator:
    def __init__(self, num_terms):
        self.num_terms = num_terms
        self.current_index = 0
        self.a, self.b = 0, 1  # Initialize first two Fibonacci numbers

    def __iter__(self):
        return self

    def __next__(self):
        if self.current_index >= self.num_terms:
            raise StopIteration

        if self.current_index == 0:
            result = self.a
        elif self.current_index == 1:
            result = self.b
        else:
            result = self.a + self.b
            self.a, self.b = self.b, result
```

```
        self.current_index += 1
        return result

num_terms = 10
fib_iter = FibonacciIterator(num_terms)

print(f"Fibonacci sequence up to {num_terms} terms:")
for fib_num in fib_iter:
    print(fib_num)
```

```
Fibonacci sequence up to 10 terms:
0
1
1
2
3
5
8
13
21
34
```

[10]:
```
# Que 6. Write a generator function in Python that yields the powers of 2 up to␣
 ↪a given exponent.
def powers_of_two_up_to(exponent):
    power = 0
    while power <= exponent:
        yield 2 ** power
        power += 1
exponent = 5  # Generate powers of 2 up to 2^5
for power_of_two in powers_of_two_up_to(exponent):
    print(power_of_two)
```

```
1
2
4
8
16
32
```

[15]:
```
# Que 7. Implement a generator function that reads a file line by line and␣
 ↪yields each line as a string.
def read_lines(file_path):
    with open(file_path, 'r')as file:
        for line in file:
            yield line.strip()
file_path = r"C:\Users\lenovo\Desktop\example.txt"
```

```
line_gen = read_lines(file_path)
for line in "line_gen":
    print(line)
```

```
l
i
n
e

_
g
e
n
```

[16]:
```
# Que 8. Use a lambda function in Python to sort a list of tuples based on the␣
 ↪second element of each tuple.
my_list = [(3, 2), (1, 5), (4, 1), (2, 4)]

sorted_list = sorted(my_list, key=lambda x: x[1])

print(sorted_list)
```

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

[17]:
```
# Que 9. Write a Python program that uses `map()` to convert a list of␣
 ↪temperatures from Celsius to Fahrenheit.
def celsius_to_fahrenheit(celsius):
  return (celsius * 9/5) + 32

temperatures_celsius = [25, 30, 15, 20]
temperatures_fahrenheit = list(map(celsius_to_fahrenheit, temperatures_celsius))

print(temperatures_fahrenheit)
```

```
[77.0, 86.0, 59.0, 68.0]
```

[18]:
```
# Que 10. Create a Python program that uses `filter()` to remove all the vowels␣
 ↪from a given string.
def is_not_vowel(char):
  vowels = "aeiouAEIOU"
  return char not in vowels

my_string = "This is a sample string"
filtered_string = "".join(filter(is_not_vowel, my_string))

print(filtered_string)
```

```
Ths s  smpl strng
```

[14]:
```
# Que 11.Imagine an accounting routine used in a book shop. It works on a list␣
 ↪with sublists, which look like this:
#  order Number  Book Title and Author               Quantity  Price per Item
#   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        Einfuhrung in Python3, Bernd Klein    3          24.99
# Write a Python program, which returns a list with 2-tuples. Each tuple␣
 ↪consists of the order number and the
# product of the price per item and the quantity. The product should be␣
 ↪increased by 10,- € if the value of the
# order is smaller than 100,00 €.

# Write a Python program using lambda and map.

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, "Einfuhrung in Python3, Bernd Klein", 3, 24.99)
]

# Using lambda function and map to calculate total order value
result = list(map(lambda order: (order[0], (order[2] * order[3]) if (order[2] *␣
 ↪order[3]) >= 100 else (order[2] * order[3]) + 10), orders))

print(result)
```

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