practical-answers

August 1, 2024

[26]:
```
'''1. Write a Python function that takes a list of numbers as input and returns␣
 ↪the sum of all even numbers in
the list.'''

#ANSWER
def sum_of_even_numbers(numbers):
    return sum(num for num in numbers if num % 2 == 0)
```

[27]:
```
'''2. Create a Python function that accepts a string and returns the reverse of␣
 ↪that string.'''

def reverse_string(s):
    return s[::-1]
```

[28]:
```
'''3. Implement a Python function that takes a list of integers and returns a␣
 ↪new list containing the squares of
each number.'''


def squares_of_numbers(numbers):
    return [num ** 2 for num in numbers]
```

[29]:
```
'''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
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def primes_up_to_200():
    return [n for n in range(1, 201) if is_prime(n)]
```

[30]:
```
'''5. Create an iterator class in Python that generates the Fibonacci sequence␣
 ↪up to a specified number of
terms.'''

class FibonacciIterator:
    def __init__(self, terms):
        self.terms = terms
        self.a, self.b = 0, 1
        self.count = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.count < self.terms:
            self.count += 1
            self.a, self.b = self.b, self.a + self.b
            return self.a
        else:
            raise StopIteration
```

[31]:
```
'''6. Write a generator function in Python that yields the powers of 2 up to a␣
 ↪given exponent.'''


def powers_of_two(exponent):
    for i in range(exponent + 1):
        yield 2 ** i
```

[32]:
```
'''7. Implement a generator function that reads a file line by line and yields␣
 ↪each line as a string.'''

def read_file_lines(filename):
    with open(filename, 'r') as file:
        for line in file:
            yield line.strip()
```

[33]:
```
'''8. Use a lambda function in Python to sort a list of tuples based on the␣
 ↪second element of each tuple.'''

tuples_list = [(1, 3), (2, 1), (4, 2)]
sorted_list = sorted(tuples_list, key=lambda x: x[1])
```

[34]:
```
'''9. Write a Python program that uses to convert a list of temperatures from␣
 ↪Celsius to Fahrenheit.'''
```

```
def celsius_to_fahrenheit(celsius_list):
    return [((9/5) * temp + 32) for temp in celsius_list]
```

[35]:
```
'''10. Create a Python program that uses to remove all the vowels from a given␣
 ↪string.'''

def remove_vowels(s):
    vowels = "aeiouAEIOU"
    return ''.join(char for char in s if char not in vowels)
```

[36]:
```
'''11) Imagine an accounting routine used in a book shop. It works on a list␣
 ↪with sublists, which look like this:
Order Number
34587
98762
77226
88112
Book Title and Author
Learning Python, Mark Lutz
Programing Python, Mark Lutz
Head First Python, Paul Barry
Einführung in Python3, Bernd Klein
Quantity
4
5
3
3
Price per Item
40.95
56.80
32.95
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,- C if the value of the
order is smaller than 100,00 C.
Write a Python program using lambda and map.'''



orders = [
    (34587, "Learning Python, Mark Lutz", 4, 40.95),
    (98762, "Programing Python, Mark Lutz", 5, 56.80),
    (77226, "Head First Python, Paul Barry", 3, 32.95),
    (88112, "Einführung in Python3, Bernd Klein", 3, 24.99)
```

```
]

def calculate_order_value(order):
    order_number, title, quantity, price_per_item = order
    total_price = quantity * price_per_item
    if total_price < 100:
        total_price += 10
    return (order_number, total_price)

# Using map and lambda
result = list(map(lambda order: calculate_order_value(order), orders))

print(result)
```

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

[ ]:

[ ]: