completed_list_assignment_c_ANIL

September 28, 2025

1 List Assignments – Solved

Solutions to all 12 list-based Python assignments.

1.0.1 Assignment 1: Creating and Accessing Lists

Create a list of the first 20 positive integers. Print the list.

[1]:
```
lst = list(range(1, 21))
print(lst)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

1.0.2 Assignment 2: Accessing List Elements

Print the first, middle, and last elements of the list created in Assignment 1.

[2]:
```
lst = list(range(1, 21))
first = lst[0]
middle = lst[len(lst)//2]  # or lst[9] for 20 elements (0-indexed)
last = lst[-1]
print("First:", first)
print("Middle:", middle)
print("Last:", last)
```

```
First: 1
Middle: 11
Last: 20
```

1.0.3 Assignment 3: List Slicing

Print the first five elements, the last five elements, and the elements from index 5 to 15 of the list created in Assignment 1.

[3]:
```
lst = list(range(1, 21))
print("First five:", lst[:5])
print("Last five:", lst[-5:])
print("Index 5 to 15:", lst[5:16])
```

```
First five: [1, 2, 3, 4, 5]
Last five: [16, 17, 18, 19, 20]
Index 5 to 15: [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

1.0.4 Assignment 4: List Comprehensions

Create a new list containing the squares of the first 10 positive integers using a list comprehension. Print the new list.

[4]:
```
squares = [i**2 for i in range(1, 11)]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

1.0.5 Assignment 5: Filtering Lists

Create a new list containing only the even numbers from the list created in Assignment 1 using a list comprehension. Print the new list.

[5]:
```
evens = [x for x in range(1, 21) if x % 2 == 0]
print(evens)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

1.0.6 Assignment 6: List Methods

Create a list of random numbers and sort it in ascending and descending order. Remove the duplicates from the list and print the modified list.

[6]:
```
import random

# Create list with duplicates
lst = [random.randint(1, 10) for _ in range(15)]
print("Original:", lst)

# Sort ascending
lst_sorted_asc = sorted(lst)
print("Ascending:", lst_sorted_asc)

# Sort descending
lst_sorted_desc = sorted(lst, reverse=True)
print("Descending:", lst_sorted_desc)

# Remove duplicates (preserve order)
unique = []
for x in lst:
    if x not in unique:
        unique.append(x)
print("Without duplicates:", unique)
```

```
# Or simply: list(set(lst)) - but order not preserved
```

```
Original: [8, 3, 5, 7, 5, 2, 4, 8, 7, 1, 3, 3, 5, 5, 6]
Ascending: [1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 8]
Descending: [8, 8, 7, 7, 6, 5, 5, 5, 5, 4, 3, 3, 3, 2, 1]
Without duplicates: [8, 3, 5, 7, 2, 4, 1, 6]
```

1.0.7 Assignment 7: Nested Lists

Create a nested list representing a 3x3 matrix and print the matrix. Access and print the element at the second row and third column.

[7]:
```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

print("Matrix:")
for row in matrix:
    print(row)

# Second row (index 1), third column (index 2)
element = matrix[1][2]
print("Element at row 2, col 3:", element)
```

```
Matrix:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Element at row 2, col 3: 6
```

1.0.8 Assignment 8: List of Dictionaries

Create a list of dictionaries where each dictionary represents a student with keys 'name' and 'score'. Sort the list of dictionaries by the 'score' in descending order and print the sorted list.

[8]:
```
students = [
    {'name': 'Alice', 'score': 88},
    {'name': 'Bob', 'score': 95},
    {'name': 'Charlie', 'score': 76},
    {'name': 'Diana', 'score': 92}
]

sorted_students = sorted(students, key=lambda x: x['score'], reverse=True)
print("Sorted by score (desc):")
for s in sorted_students:
    print(s)
```

```
Sorted by score (desc):
{'name': 'Bob', 'score': 95}
{'name': 'Diana', 'score': 92}
{'name': 'Alice', 'score': 88}
{'name': 'Charlie', 'score': 76}
```

1.0.9 Assignment 9: Matrix Transposition

Write a function that takes a 3x3 matrix (nested list) as input and returns its transpose. Print the original and transposed matrices.

[9]:
```
def transpose(matrix):
    return [[matrix[j][i] for j in range(len(matrix))] for i in␣
 ↪range(len(matrix[0]))]

original = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

transposed = transpose(original)

print("Original:")
for row in original:
    print(row)

print("\nTransposed:")
for row in transposed:
    print(row)
```

```
Original:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Transposed:
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
```

1.0.10 Assignment 10: Flattening a Nested List

Write a function that takes a nested list and flattens it into a single list. Print the original and flattened lists.

[11]:
```
def flatten(nested):
    flat = []
    for item in nested:
```

```
        if isinstance(item, list):
            flat.extend(flatten(item))
        else:
            flat.append(item)
    return flat

nested_list = [[1, 2], [3, [4, 5]], 6, [7, 8, [9]]]
flattened = flatten(nested_list)

print("Original:", nested_list)
print("Flattened:", flattened)
```

```
Original: [[1, 2], [3, [4, 5]], 6, [7, 8, [9]]]
Flattened: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1.0.11 Assignment 11: List Manipulation

Create a list of the first 10 positive integers. Remove the elements at indices 2, 4, and 6, and insert the element '99' at index 5. Print the modified list.

[13]:
```
lst = list(range(1, 11))  # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("Original:", lst)

# Remove from highest index to lowest to avoid shifting issues
for index in sorted([2, 4, 6], reverse=True):
    del lst[index]

# Now insert 99 at index 5
lst.insert(5, 99)

print("Modified:", lst)
```

```
Original: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Modified: [1, 2, 4, 6, 8, 99, 9, 10]
```

1.0.12 Assignment 12: List Reversal

Write a function that takes a list and returns a new list with the elements in reverse order. Print the original and reversed lists.

[14]:
```
def reverse_list(lst):
    return lst[::-1]  # or list(reversed(lst))

original = [1, 2, 3, 4, 5]
reversed_lst = reverse_list(original)

print("Original:", original)
print("Reversed:", reversed_lst)
```

```
Original: [1, 2, 3, 4, 5]
Reversed: [5, 4, 3, 2, 1]
```