

Assignment 21 Solutions

1. Write a function that takes a list and a number as arguments. Add the number to the end of the list, then remove the first element of the list. The function should then return the updated list.

Examples:

`next_in_line([5, 6, 7, 8, 9], 1) → [6, 7, 8, 9, 1]`

`next_in_line([7, 6, 3, 23, 17], 10) → [6, 3, 23, 17, 10]`

`next_in_line([1, 10, 20, 42], 6) → [10, 20, 42, 6]`

`next_in_line([], 6) → "No list has been selected"`

```
In [4]: def next_in_line(in_list,in_num):
        if len(in_list) > 1:
            in_list.append(in_num)
            in_list.remove(in_list[0])
            print(f'Output → {in_list}')
        else:
            print('No list has been selected')

        next_in_line([5, 6, 7, 8, 9], 1)
        next_in_line([7, 6, 3, 23, 17], 10)
        next_in_line([1, 10, 20, 42 ], 6)
        next_in_line([], 6)
```

Output → [6, 7, 8, 9, 1]

Output → [6, 3, 23, 17, 10]

Output → [10, 20, 42, 6]

No list has been selected

2. Create the function that takes a list of dictionaries and returns the sum of people's budgets.

Examples:

`get_budgets([{ "name": "John", "age": 21, "budget": 23000 },`

`{ "name": "Steve", "age": 32, "budget": 40000 },`

`{ "name": "Martin", "age": 16, "budget": 2700 }])` → 65700

`get_budgets([{ "name": "John", "age": 21, "budget": 29000 },`

`{ "name": "Steve", "age": 32, "budget": 32000 },`

`{ "name": "Martin", "age": 16, "budget": 1600 }])` → 62600

```
In [7]: def get_budgets(in_dict):
        sum = 0
        for ele in in_dict:
            sum += ele["budget"]
            print(f'Output → {sum}')

        get_budgets([
            { "name": "John", "age": 21, "budget": 23000 },
            { "name": "Steve", "age": 32, "budget": 40000 },
            { "name": "Martin", "age": 16, "budget": 2700 }
        ])

        get_budgets([
            { "name": "John", "age": 21, "budget": 23000 },
            { "name": "Steve", "age": 32, "budget": 40000 },
            { "name": "Martin", "age": 16, "budget": 2700 }
        ])
```

Output → 65700

Output → 65700

3. Create a function that takes a string and returns a string with its letters in alphabetical order.

Examples:

```
alphabet_soup("hello") → "ehllo"

alphabet_soup("edabit") → "abdeit"

alphabet_soup("hacker") → "acehkr"

alphabet_soup("geek") → "eegk"

alphabet_soup("javascript") → "aacijprstv"
```

```
In [8]: def alphabet_soup(in_string):
        out_string = ''.join(sorted(in_string))
        print(f'{in_string} → {out_string}')
```

```
alphabet_soup("hello")
alphabet_soup("edabit")
alphabet_soup("hacker")
alphabet_soup("geek")
alphabet_soup("javascript")
```

```
hello → ehllo
edabit → abdeit
hacker → acehkr
geek → eegk
javascript → aacijprstv
```

4. What will be the value of your investment at the end of the 10 year period?

Create a function that accepts the principal p , the term in years t , the interest rate r , and the number of compounding periods per year n . The function returns the value at the end of term rounded to the nearest cent.

For the example above:

compound_interest(10000, 10, 0.06, 12) → 18193.97

Note that the interest rate is given as a decimal and $n=12$ because with monthly compounding there are 12 periods per year. Compounding can also be done annually, quarterly, weekly, or daily.

Examples:

```
compound_interest(100, 1, 0.05, 1) → 105.0

compound_interest(3500, 15, 0.1, 4) → 15399.26

compound_interest(100000, 20, 0.15, 365) → 2007316.26
```

```
In [10]: def compound_interest(principal, years, roi, cp):
        ci = principal * (1 + (roi / cp)) ** (cp * years)
        print(f'Output → {ci:.2f}')
```

```
compound_interest(100, 1, 0.05, 1)
compound_interest(3500, 15, 0.1, 4)
compound_interest(100000, 20, 0.15, 365)
```

```
Output → 105.00
Output → 15399.26
Output → 2007316.26
```

5. Write a function that takes a list of elements and returns only the integers.

Examples:

```
return_only_integer([9, 2, "space", "car", "lion", 16]) → [9, 2, 16]

return_only_integer(["hello", 81, "basketball", 123, "fox"]) → [81, 123]

return_only_integer([10, "121", 56, 20, "car", 3, "lion"]) → [10, 56, 20, 3]

return_only_integer(["String", True, 3.3, 1]) → [1]
```

```
In [11]: def return_only_integer(in_list):
```

```

def return_only_integer(in_list):
    out_list = []
    for ele in in_list:
        if type(ele) == int:
            out_list.append(ele)
    print(f'{in_list} → {out_list}')

return_only_integer([9, 2, "space", "car", "lion", 16])
return_only_integer(["hello", 81, "basketball", 123, "fox"])
return_only_integer([10, "121", 56, 20, "car", 3, "lion"])
return_only_integer(["String", True, 3.3, 1])

```

```

[9, 2, 'space', 'car', 'lion', 16] → [9, 2, 16]
['hello', 81, 'basketball', 123, 'fox'] → [81, 123]
[10, '121', 56, 20, 'car', 3, 'lion'] → [10, 56, 20, 3]
['String', True, 3.3, 1] → [1]

```

In []:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js