Programming_Assingment21

Question1

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 [1]:

```
lst = [5, 6, 7, 8, 9]
def next_in_line(lst,num):
    if len(lst) > 0 :
        lst.append(num)
        return lst[1:]
    else:
        print("'No list has been selected'")
```

In [2]:

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

Out[2]:

```
[6, 7, 8, 9, 1]
```

In [3]:

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

Out[3]:

```
[6, 3, 23, 17, 10]
```

In [4]:

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

Out[4]:

```
[10, 20, 42, 6]
```

In [5]:

```
next_in_line([], 6)
```

```
'No list has been selected'
```

Question2

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 [6]:

```
def get_budgets(listDict):
    sum = 0
    for dc in listDict:
        for k,v in dc.items():
            if k == 'budget':
                sum = sum + v
    return sum
```

In [7]:

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

Out[7]:

```
65700
```

In [8]:

```
get_budgets([
{ 'name': 'John', 'age': 21, 'budget': 29000 },
{ 'name': 'Steve', 'age': 32, 'budget': 32000 },
```

```
{ 'name': 'Martin', 'age': 16, 'budget': 1600 }
])
```

Out[8]:

```
62600
```

In [ ]:

Question3

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 [9]:

```
def alphabet_soup(str):
    return ''.join(sorted(str))
```

In [10]:

```
alphabet_soup('hello')
```

Out[10]:

```
'ehllo'
```

In [11]:

```
alphabet_soup('edabit')
```

Out[11]:

```
'abdeit'
```

In [12]:

```
alphabet_soup('hacker')
```

Out[12]:

```
'acehkr'
```

In [13]:

```
alphabet_soup('geek')
```

Out[13]:

```
'eegk'
```

In [14]:

```
alphabet_soup('javascript')
```

Out[14]:

```
'aacijprstv'
```

Question4

Suppose that you invest $10,000 for 10 years at an interest rate of 6% compounded monthly.

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

FV = PV(1 + r/m)mt

In [15]:

```
def compound_interest(amt, years, intrest, compPeriod):
    future_value = amt *(1 + (intrest/compPeriod)) ** (years * compPeriod)
    return round(future_value,2)
```

In [16]:

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

Out[16]:

```
105.0
```

In [17]:

```
compound_interest(3500, 15, 0.1, 4)
```

Out[17]:

```
15399.26
```

In [18]:

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

Out[18]:

```
2007316.26
```

Question5

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 [19]:

```
def return_only_integer(lst):
    intLst = []
    for i in lst:
        if type(i) == int:
            intLst.append(i)
    return intLst
```

In [20]:

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

Out[20]:

```
[9, 2, 16]
```

In [21]:

```
return_only_integer(['hello', 81, 'basketball', 123, 'fox'])
```

Out[21]:

```
[81, 123]
```

In [22]:

```
return_only_integer([10, '121', 56, 20, 'car', 3, 'lion'])
```

Out[22]:

```
[10, 56, 20, 3]
```

In [23]:

```
return_only_integer(['String', True, 3.3, 1])
```

Out[23]:

```
[1]
```