/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"

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def next\_in\_line(my\_list,my\_numb):

if my\_list == []:

print("No list has been selected")

return "No list has been selected"

else:

my\_list.pop(0)

my\_list.append(my\_numb)

print(my\_list)

return my\_list

try:

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)

lg.info("""Class next\_in\_line() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def get\_budgets(my\_dictionaries):

total\_budget = 0

for i in my\_dictionaries:

total\_budget += i["budget"]

print(total\_budget)

try:

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": 29000 },

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

{ "name": "Martin", "age": 16, "budget": 1600 }

])

lg.info("""Class get\_budgets() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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"

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def alphabet\_soup(strings):

final\_arrange = ""

string\_list = list(strings)

string\_list.sort()

for i in string\_list:

final\_arrange += i

print(final\_arrange)

try:

alphabet\_soup("hello")

alphabet\_soup("edabit")

alphabet\_soup("hacker")

alphabet\_soup("geek")

alphabet\_soup("javascript")

lg.info("""Class alphabet\_soup() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def compound\_interest(principal, term\_years, interest\_rate, compounding):

ci = (principal\*(1+(interest\_rate/compounding))\*\*(term\_years\*compounding))

print(ci)

try:

compound\_interest(100, 1, 0.05, 1)

compound\_interest(3500, 15, 0.1, 4)

compound\_interest(100000, 20, 0.15, 365)

lg.info("""Class compoubd\_interest() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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]

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def return\_only\_integer(my\_list):

my\_integers = []

for i in my\_list:

if type(i) == int:

my\_integers.append(i)

print(my\_integers)

try:

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])

lg.info("""Class return\_only\_integer() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass