**자바프로그래밍 중간 대체 과제**

12162129 경제학과 신수진

**<Multiply>**

Q : Write a function that will receive 2 numbers as input and it should return the multiplication of these 2 numbers

A :

def mult\_two(a, b):

return a \* b

**<Easy Unpack>**

Q : Your mission here is to create a function that gets a tuple and returns a tuple with 3 elements - the first, third and second element from the last for the given array.

A :

def easy\_unpack(elements: tuple) -> tuple:

return (elements[0], elements[2], elements[-2])

**<First Word (simplified)>**

Q : You are given a string where you have to find its first word.

* Input string consists of only english letters and spaces.
* There aren’t any spaces at the beginning and the end of the string.

A :

def first\_word(text: str) -> str:

text = text.split()

return text[0]

**<Acceptable Password Ⅰ>**

Q : You are the beginning of a password series. Every mission will be based on the previous one. Going forward the missions will become slightly more complex.

In this mission you need to create a password verification function.

Those are the verification conditions:

* the length should be bigger than 6.

A :

def is\_acceptable\_password(password: str) -> bool:

if len(password) > 6:

return True  
 else:

return False

**<Number Length>**

Q : You have a positive integer. Try to find out how many digits it has?

A :

def number\_length(a: int) -> int:

a = str(a)

return len(a)

**<End Zeros>**

Q : Try to find out how many zeros a given number has at the end.

A :

def end\_zeros(num: int) -> int:

num = str(num)

num\_rev = num[::-1]

num\_revList = list(num\_rev)

num\_revList

count = 0

for i in range(len(num\_revList)):

if num\_revList[i] == '0':

count += 1

else:

break

return count

**<Backward String>**

Q : You should return a given string in reverse order.

A :

def backward\_string(val: str) -> str:

return val[::-1]

**<Remove All Before>**

Q : Not all of the elements are important. What you need to do here is to remove from the list all of the elements before the given one.

We have two edge cases here: (1) if a cutting element cannot be found, then the list shoudn't be changed. (2) if the list is empty, then it should remain empty.

A :

from typing import Iterable

def remove\_all\_before(items: list, border: int) -> Iterable:

if border in items:

while True:

if items[0] != border:

del items[0]

else:

break

return items

else:

return items

**<All Upper Ⅰ>**

Q : Check if a given string has all symbols in upper case. If the string is empty or doesn't have any letter in it - function should return True.

A :

def is\_all\_upper(text: str) -> bool:

if text.isupper() == True:

return True

elif text == '':

return True

elif text.isspace() == True:

return True

elif text.isdigit() == True:

return True

else:

return False

**<Replace First>**

Q : In a given list the first element should become the last one. An empty list or list with only one element should stay the same.

A :

from typing import Iterable

def replace\_first(items: list) -> Iterable:

if len(items) > 1:

for i in range(len(items)-1):

\_tmp = items[i]

items[i] = items[i+1]

items[i+1] = \_tmp

return items

else:

return items

**<Max Digit>**

Q : You have a number and you need to determine which digit in this number is the biggest.

A :

def max\_digit(number: int) -> int:

num\_list = list(str(number))

max\_num = max(num\_list)

return int(max\_num)

**<Split Pairs>**

Q : Split the string into pairs of two characters. If the string contains an odd number of characters, then the missing second character of the final pair should be replaced with an underscore ('\_').

A :

def split\_pairs(a):

a\_list = list(a)

new\_list = []

if (len(a\_list) % 2) == 0:

z = 0

for i in range(len(a\_list)//2):

word = a\_list[z] + a\_list[z+1]

z += 2

new\_list.append(word)

return new\_list

elif (len(a\_list) % 2) == 1:

z = 0

for i in range(len(a\_list)//2):

word = a\_list[z] + a\_list[z+1]

z += 2

new\_list.append(word)

last\_word = a\_list[-1] + '\_'

new\_list.append(last\_word)

return new\_list

else:

return None

**<Beginning Zeros>**

Q : You have a string that consist only of digits. You need to find how many zero digits ("0") are at the beginning of the given string.

A :

def beginning\_zeros(number: str) -> int:

num\_list = list(number)

count = 0

for i in range(len(num\_list)):

if num\_list[i] == '0':

count += 1

else:

break

return count

**<Nearest Value>**

Q : Find the nearest value to the given one.

You are given a list of values as set form and a value for which you need to find the nearest one.

A few clarifications:

* If 2 numbers are at the same distance, you need to choose the smallest one;
* The set of numbers is always non-empty, i.e. the size is >=1;
* The given value can be in this set, which means that it’s the answer;
* The set can contain both positive and negative numbers, but they are always integers;
* The set isn’t sorted and consists of unique numbers.

A :

def nearest\_value(values: set, one: int) -> int:

if one in values:

return one

else:

val\_list = list(values)

val\_list.append(one)

val\_list.sort()

one\_idx = val\_list.index(one)

if one\_idx == 0:

return val\_list[one\_idx+1]

elif one\_idx == len(val\_list)-1:

return val\_list[one\_idx-1]

else:

rGap = abs(one - val\_list[one\_idx-1])

lGap = abs(one - val\_list[one\_idx+1])

if rGap == lGap:

return val\_list[one\_idx-1]

elif rGap < lGap:

return val\_list[one\_idx-1]

else:

return val\_list[one\_idx+1]

**<Between Markets>**

Q : You are given a string and two markers (the initial one and final). You have to find a substring enclosed between these two markers. But there are a few important conditions.

This is a simplified version of the Between Markers mission.

* The initial and final markers are always different.
* The initial and final markers are always 1 char size.
* The initial and final markers always exist in a string and go one after another.

A :

def between\_markers(text: str, begin: str, end: str) -> str:

if begin in text:

begin\_index = text.index(begin) + 1

if end in text:

end\_index = text.index(end)

return text[begin\_index:end\_index]

**<Correct Sentence>**

Q : For the input of your function, you will be given one sentence. You have to return a corrected version, that starts with a capital letter and ends with a period (dot).

Pay attention to the fact that not all of the fixes are necessary. If a sentence already ends with a period (dot), then adding another one will be a mistake.

A :

def correct\_sentence(text: str) -> str:

capi = text[0].upper()

text = text.replace(text[0], capi, 1)

if text[-1] != '.':

text = text + '.'

return text

**<Is Even>**

Q : Check is the given number is even or not. Your function shoudl return True if the number is even, andFalse if the number is odd.

A :

def is\_even(num: int) -> bool:

if num % 2 == 0:

return True

else:

return False