#1

У нас є список показань заборгованостей з комунальних послуг наприкінці місяця. Заборгованості можуть бути від'ємними — у нас переплата, чи додатними, якщо необхідно сплатити за рахунками. Напишіть функцію amount\_payment, яка приймає на вхід список платежів, підсумовує додатні значення та повертає суму платежу наприкінці місяця.

def amount\_payment(payment):

sum = 0

for value in payment:

if value > 0:

sum = sum + value

else:

continue

return sum

print(amount\_payment([1, -3, 4]))

#2

При аналізі даних часто виникає необхідність позбавитися екстремальних значень, перш ніж почати працювати з даними далі. Напишіть функцію prepare\_data, яка видаляє з переданого списку найбільше та найменше значення, сортує його в порядку зростання і повертає змінений список як результат.

def prepare\_data(data):

removed\_item = data.remove(min(data))

removed\_item = data.remove(max(data))

new\_data = sorted(data)

return new\_data

print(prepare\_data([1, -3, 4, 0, 1000, -500]))

#3

Ми розробляємо кулінарний блог. І в рецептах, при написанні списку інгредієнтів, ми розділяємо їх комами, а перед останнім ставимо союз "and", як у прикладі нижче:

2 eggs, 1 liter sugar, 1 tsp salt and vinegar

Напишіть функцію format\_ingredients, яка прийматиме на вхід список з інгредієнтів ["2 eggs", "1 liter sugar", "1 tsp salt", "vinegar"] та повертатиме рядок зібраний з його елементів в описаний вище спосіб. Ваша функція має вміти обробляти списки будь-якої довжини.

def format\_ingredients(items):

value = len(items)

if value == 0:

last\_item = ""

return last\_item

elif value == 1:

last\_item = items[0]

return last\_item

elif value == 2:

last\_item = items.pop(-1)

prelast\_item = items.pop(-1)

new\_last\_item = (prelast\_item + " and " + last\_item)

return new\_last\_item

else:

last\_item = items.pop(-1)

prelast\_item = items.pop(-1)

new\_last\_item = (prelast\_item + " and " + last\_item)

count = len(items)

string = "\_"

for count in items:

string = string + ", " + count

string = string + ", " + new\_last\_item

string\_copy = string[3::]

return string\_copy

print(format\_ingredients(

[]))

#4

Сучасна система оцінок в університеті має такий вигляд:

| **Оцінка** | **Бали** | **Оцінка ECTS** | **Пояснення** |
| --- | --- | --- | --- |
| 1 | 0-34 | F | Unsatisfactorily |
| 2 | 35-59 | FX | Unsatisfactorily |
| 3 | 60-66 | E | Enough |
| 3 | 67-74 | D | Satisfactorily |
| 4 | 75-89 | C | Good |
| 5 | 90-95 | В | Very good |
| 5 | 96-100 | A | Perfectly |

Реалізуйте дві функції. Перша буде використовуватись у бухгалтерії при розрахунку стипендії, get\_grade приймає ключ у вигляді оцінки **ECTS**, і має повертати відповідну п'ятибальну оцінку (перший стовпчик таблиці). Друга get\_description теж приймає ключ у вигляді оцінки **ECTS**, але повертатиме пояснення оцінки в текстовому форматі (останній стовпчик таблиці) і буде використана в електронній заліковій книжці студента. На відсутній ключ функції повинні повертати значення None .

*def* get\_grade(*key*):

    points = {"F": 1, "FX": 2, "E": 3, "D": 3, "C": 4, "B": 5, "A": 5}

    grade = points.get(*key*)

    return grade

*def* get\_description(*key*):

    points = {"F": "Unsatisfactorily", "FX": "Unsatisfactorily", "E": "Enough",

              "D": "Satisfactorily", "C": "Good", "B": "Very good", "A": "Perfectly"}

    description = points.get(*key*)

    return description

print(get\_grade("B"))

print(get\_description("B"))

#5

Як ми знаємо, ключ у словнику має бути унікальним, тоді як значення його ні. Реалізуйте функцію lookup\_key для пошуку всіх ключів за значенням у словнику. Першим параметром у функцію ми передаємо словник, а другим — значення, що хочемо знайти. Таким чином, результат може бути як список ключів, так і порожній список, якщо ми нічого не знайдемо.

def lookup\_key(data, values):

key = data

count = values

search = []

for kies, value in data.items():

if value == count:

search.append(kies)

return search

#6

У нас є список показників студентів групи – це список з отриманими балами з тестування. Необхідно поділити список на дві частини. Напишіть функцію split\_list, яка приймає список (цілі числа), знаходить середнє значення бала у списку та ділить його на два списки. У перший потрапляють значення менше середнього, включаючи середнє значення, тоді як у другий — строго більше від середнього. Функція повертає кортеж цих двох списків. Для порожнього списку повертаємо два порожні списки.

def split\_list(grade):

sum\_num = 0

low\_grade = []

high\_grade = []

if len(grade) > 0:

for num in grade:

sum\_num = sum\_num + num

aver\_num = sum\_num / len(grade)

for num in grade:

if num <= aver\_num:

low\_grade.append(num)

else:

high\_grade.append(num)

return low\_grade, high\_grade

#7

Є чотирикутна схема польотів дронів з координатами (1, 2, 3, 4). У нас є словник points, ключі якого — кортежі, точки польоту між координатами чотирикутника, вигляду (1, 2). Значення словника — це відстані між вказаними точками.

Приклад:

points = {(0, 1): 2, (0, 2): 3.8, (0, 3): 2.7, (1, 2): 2.5, (1, 3): 4.1, (2, 3): 3.9}

Напишіть функцію calculate\_distance, яка на вхід приймає список координат чотирикутника зі словника виду [0, 1, 3, 2, 0]. Функція повинна підрахувати, використовуючи вказаний словник, яку загальну відстань пролетить дрон, рухаючись між точками польоту.

Примітки:

* коли беремо у словника points значення, у ключі кортежі завжди має бути першою координата з меншим значенням — (2, 3), але не (3, 2);
* для порожнього списку та списку з однією координатою функція calculate\_distance має повертати 0.

points = {

    (0, 1): 2,

    (0, 2): 3.8,

    (0, 3): 2.7,

    (1, 2): 2.5,

    (1, 3): 4.1,

    (2, 3): 3.9,

}

*def* calculate\_distance(*coordinates*):

    distance = 0

    num\_points = len(*coordinates*)

    flag = True

    if num\_points == 0:

        return distance

    else:

        for num in range(num\_points-1):

            p\_1 = *coordinates*[num]

            p\_2 = *coordinates*[num+1]

            coord\_1 = (p\_1, p\_2)

            if p\_1 > p\_2:

                p\_1, p\_2 = p\_2, p\_1

                coord\_1 = (p\_1, p\_2)

            dis = points[coord\_1]

            distance = distance + dis

    return distance

calculate\_distance([0, 1, 3, 2, 0])

#8

Потрібно написати функцію реалізації наступного ігрового алгоритму. На вхід функції game подається два аргументи: список, що складається зі списків, та початкове значення power - енергія гравця. Внутрішні списки — це списки з числовим значенням енергії, які може поглинути гравець, якщо вони менші або дорівнюють його енергії. Після поглинання елементу списку він рухається за списком далі та, або поглинає список повністю до кінця, або, якщо знаходить енергію вище за власну, залишає його і переходить до наступного списку. Наприкінці обходу всіх списків функція повинна повернути загальну отриману енергію гравця.

Приклад списку:

[[1, 1, 5, 10], [10, 2], [1, 1, 1]]

Для цього списку і початкової енергії рівної 1 гравець поглине з першого списку перші два значення і залишить його, зустрівши значення 5, через те, що на цей момент матиме енергію в 3. Другий список пропустить відразу, а третій повністю поглине та отримає остаточну енергію в 6.

def game(terra, power):

for list in terra:

for forse in list:

if power >= forse:

power += forse

else:

break

return power

print(game([[1, 1, 5, 10], [10, 2], [1, 1, 1]], 1))

№9

Всім відомо, що для доступу до кредитної картки банку потрібний пін-код. Класично склалося, що це поєднання чотири цифри. Нам необхідно вирішити наступне програмістське завдання. Є підготовлений перелік пін-кодів. Напишіть функцію is\_valid\_pin\_codes, яка буде приймати як параметр список цих пін-кодів — рядок з чотирьох цифр і повертати логічне значення — валідний список чи ні. Переконайтеся, що серед цих пін-кодів у списку не буде дублікатів, всі вони зберігаються у вигляді рядків, їх довжина дорівнює 4 символам і містять вони тільки цифри.

Приклад аргументу для функції is\_valid\_pin\_codes:

['1101', '9034', '0011']

Якщо список відповідає всім поставленим умовам, функція повертає логічне значення True. Якщо хоч одну з умов порушено, повертається значення — False. Передбачити перевірку на порожній список в аргументі функції та повернути при цьому значення False.

def is\_valid\_pin\_codes(pin\_codes):

res = False

once\_set = set(pin\_codes)

for pin in pin\_codes:

if len(pin\_codes) != len(once\_set):

res = False

break

if len(pin) != 4:

res = False

break

else:

for i in pin:

if 48 <= ord(i) <= 57:

res = True

else:

res = False

break

return res

print(is\_valid\_pin\_codes(['001a', '0011']))

№10

Браузер Chrome пропонує нам згенеровані випадкові паролі для сайтів та вебзастосунків. Ми потренуємось розв'язувати подібні завдання. Розіб'ємо завдання на три етапи. Перший етап — створіть функцію get\_random\_password, яка буде генерувати випадковий рядок пароля.

Вимоги:

* у пароля має бути 8 символів.
* для шифрування пароля будемо використовувати наступний набір символів:

()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~

Ці символи лежать у межах від 40 до 126 коду в таблиці ASCII включно, і доступ до них можна отримати за допомогою функції chr.

chr(40) # (

chr(126) # ~

Щоб отримати випадкове ціле значення із заданого діапазону, ми використовуємо стандартний модуль Python random та його функцію randint. Вона має виклик виду randint(A, B) і повертає випадкове ціле число N, A ≤ N ≤ B.

from random import randint

random\_num = randint(40, 126)

Після виконання коду в змінній random\_num буде знаходитися випадкове ціле число від 40 до 126 включно.

Таким чином функція get\_random\_password має випадковим чином вибрати із запропонованого діапазону 8 символів та повернути згенерований пароль у вигляді рядка.

from random import randint

def get\_random\_password():

passw = ""

for i in range(0,8):

random\_num = randint(40, 126)

passw = passw + str(chr(random\_num))

return passw

№11

Другий етап. Необхідно написати функцію is\_valid\_password, яка перевірятиме отриманий параметр — пароль на надійність.

Критерії надійного пароля:

1. Довжина рядка пароля вісім символів.
2. Містить хоча б одну літеру у верхньому регістрі.
3. Містить хоча б одну літеру у нижньому регістрі.
4. Містить хоча б одну цифру.

Функція is\_valid\_password повинна повернути True, якщо переданий параметр пароль відповідає вимогам на надійність. Інакше повернути False.

def is\_valid\_password(password):

res\_A = False

res\_a = False

res\_1 = False

res = False

for ch in password:

if len(password) != 8:

break

if ord(ch) in range(65, 90):

res\_A = True

if ord(ch) in range(97, 122):

res\_a = True

if ord(ch) in range(48, 57):

res\_1 = True

if res\_A and res\_a and res\_1:

res = True

return res

print(is\_valid\_password('NmlDl1V0'))

№12

І, нарешті, третій, останній етап. Використовуючи рішення із попередніх двох завдань, напишіть функцію get\_password, яка згенерує нам випадковий надійний пароль та поверне його. Алгоритм простий — ми генеруємо пароль за допомогою функції get\_random\_password і, якщо він проходить перевірку на надійність функцією is\_valid\_password, повертаємо його, якщо ні — повторюємо ітерацію знову.

Примітка: Хорошою практикою буде обмежити кількість спроб (наприклад, до 100), щоб не отримати нескінченний цикл.

from random import randint

def get\_random\_password():

result = ""

count = 0

while count < 8:

random\_symbol = chr(randint(40, 126))

result = result + random\_symbol

count = count + 1

return result

def is\_valid\_password(password):

has\_upper = False

has\_lower = False

has\_num = False

for ch in password:

if "A" <= ch <= "Z":

has\_upper = True

elif "a" <= ch <= "z":

has\_lower = True

elif "0" <= ch <= "9":

has\_num = True

if len(password) == 8 and has\_upper and has\_lower and has\_num:

return True

return False

def get\_password():

i = 0

while i < 100:

i = i + 1

password = get\_random\_password()

if is\_valid\_password(password) == False:

continue

else:

return password

print(get\_password())

#13

Напишіть функцію parse\_folder, вона приймає єдиний параметр path, який є об'єктом Path. Функція повинна просканувати директорію path та повернути кортеж із двох списків. Перший — це список файлів усередині директорії, другий — список директорій.

Приклад виводу функції:

(['.gitignore', 'readme.md'],

['.git', '.idea', '.vscode', 'module-01', 'module-02', 'module-03', 'module-04', 'module-05', 'module-06', 'module-07',

'module-08', 'module-09', 'module-10', 'module-11', 'module-12'])

from pathlib import Path

*def* parse\_folder(*path*):

    #path = Path(path) #но теперь не работает в программе

    files = []

    folders = []

    for x in *path*.iterdir():

        if x.is\_dir():

            folders.append(x.name)

        else:

            files.append(x.name)

    return files, folders

print(parse\_folder("C:\Python\Go\_IT\Marathon"))

№14

Створіть функцію parse\_args, яка повертає рядок, складений з аргументів командного рядка, розділених пробілами. Наприклад, якщо скрипт був викликаний командою: python run.py first second, то функція parse\_args повинна повернути рядок наступного виду 'first second'.

import sys

def parse\_args():

result = ""

path = sys.argv[1:]

for args in path:

result = result + " " + args

result = result.lstrip()

return result