**ПРАКТИЧНА РОБОТА 05**

**Елементи об’єктно-орієнтованого програмування мовою Python**

**Система оцінювання**

|  |  |  |
| --- | --- | --- |
| **№** | **Тема** | **К-ть балів** |
| 1. | Вбудовані структури даних у мові Python | 3 |
| 2. | Об’єктно-орієнтоване програмування | 2 |
|  | **Всього за практичну** | **5** |
| 3. | ІНДЗ-1. Jupyter Notebook | 0,6 |
| 4. | ІНДЗ-2. Комплексні основи ООП | 1 |
| 5. | ІНДЗ-3. Стек | 0,7 |
| 6. | ІНДЗ-4. Черга | 0,7 |
|  | **Всього** | **8** |

***Вбудовані структури даних у мові Python***

***(Списки)*** Багато в чому списки є основою для інших вбудованих структур даних у мові Python. Зсередини вони реалізуються як С-масиви та поводять себе як масиви вказівників. У мові Python списки можуть містити як однотипні, так і різнотипні значення. Наприклад, введемо кілька списків:

emptyList = [] # список довільного розміру  
nElementList = [None]\*5 # список заданого розміру  
  
names = ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']  
something = [1, "a", "string", 1+2]  
print(something) #[1, 'a', 'string', 3]

Доступ до елементів списку здійснюється аналогічно до доступу до елементів масиву в мові С, проте допускається індексація як від початку списку, так і від його кінця (від’ємний індекс, починаючи з -1):

# доступ до елементів  
print(names[2])  
print(names[-2])

Виведеться наступне:

Сергій

Єлизавета

Вивід окремих елементів списку в консоль зазвичай здійснюється за допомогою циклів як поелементно, так і поіндексно. Поелементно:

# виведення елементів списку  
for name in names:  
 print(name)

У консолі з’явиться таке:

Олександр

Ірина

Сергій

Вікторія

Владислав

Єлизавета

Олег

Поіндексний вивід може відбуватись кількома способами:

i = 0  
while i < len(names):  
 print(names[i])  
 i += 1

або

for i in range(len(something)):  
 print(something[i])

де len() – це функція, що визначає кількість елементів переданого їй списку. Також зверніть увагу, що дана функція не обмежується тільки списками в якості вхідного параметру.

Для паралельної роботи з індексами та значеннями може бути зручно застосувати функцію enumerate(), яка дозволятиме циклу ітерувати як по індексах, так і по значенням, повертаючи кортежі (незмінювані списки у вигляді «індекс, значення»):

for num, name in enumerate(names, start=1):  
 print("Name {}: {}".format(num, name))

Результатом виконання даного коду стане наступний вивід:

Name 1: Олександр

Name 2: Ірина

Name 3: Сергій

Name 4: Вікторія

Name 5: Владислав

Name 6: Єлизавета

Name 7: Олег

Формувати свій список можна на основі вже існуючих даних. Доступне використання спискових включень, які є синтаксичним цукром для компактного запису ітерування по списку за допомогою циклів. Також маємо можливість «дописати» елементи в кінець існуючого списку:

# формування нового списку на базі вже існуючих даних  
names2 = [name+name for name in names] # спискове включення  
print(names2)  
  
extendedList = ['ПершийЕлемент']  
extendedList.extend(names) # дописування списку до списку  
print(extendedList)

На екран буде виведено таке:

['ОлександрОлександр', 'ІринаІрина', 'СергійСергій', 'ВікторіяВікторія', 'ВладиславВладислав', 'ЄлизаветаЄлизавета', 'ОлегОлег']

['ПершийЕлемент', 'Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']

Схожим чином працює вставка елементів у список за допомогою методів append() та insert(), проте результат може бути дещо відмінним:

appendedList = []  
appendedList.append(17) # вставка елементу в кінець списку  
appendedList.append(names) # вставка списку в список  
appendedList.insert(2, something[3]) # вставка елементу в задану позицію у списку  
print(appendedList)

У консолі буде відображено:

[17, ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег'], 3]

Зверніть увагу, що в ході вставки списку в список відбувається не дописування окремих елементів, як у випадку з extend(), а вставка єдиного елементу-списку, на що вказують додаткові квадратні дужки навколо переліку імен.

При роботі з ітерованими об’єктами доступна можливість робити зрізи (slices), зокрема «вирізати» значення з одного списку в інший. Для цього у квадратних дужках через двокрапку записуються ліва межа, права межа та крок зрізу:

print(names[0:5:2]) # ['Олександр', 'Сергій', 'Владислав']

Популярними є подібні зрізи:

* [:]/[::] — «весь список»;
* [::2] — «непарні по порядку елементи»;
* [1::2] — «парні по порядку елементи»;
* [::-1] — «всі елементи в зворотному порядку»;
* [2:] — «всі елементи, починаючи з третього»;
* [:3] — «всі елементи, не включаючи ті, що після другого».

Поширеними при роботі зі списками є агрегатні функції: sum(), count(), min(), max(), а також функція index() для знаходження позиції першої зустрічі заданого значення в списку. Застосуємо функцію sum() до списку з числами:

numbers = [78.45, 152, 21.654, -86.4, 9999]  
print(sum(numbers)) #10164.704

Очевидно, що для списків, у яких є нечислові значення, функція переважно викидатиме помилки, оскільки не знатиме, як додавати такі об’єкти.

Аналогічно підрахуємо, скільки разів зустрічається імена «Олег» та «Ольга» в списку names за допомогою методу count():

print(names.count('Олег')) # 1  
print(names.count('Ольга')) # 0

Метод index() дозволить визначити позицію імені в списку, починаючи з 0:

print(names.index("Олег")) # 6  
print(names.index('Ольга'))

Оскільки імені Ольга в списку немає, отримаємо помилку при виконанні:

ValueError: 'Ольга' is not in list

Визначення мінімуму та максимуму тісно пов’язане з порівнянням та сортуванням елементів набору. При цьому слід звернути увагу, що при сортуванні слів у стандартному для Python кодуванні UTF-8 літери [українського алфавіту](https://unicode-table.com/ru/alphabets/ukrainian/) не повністю впорядковані за своїми кодами. Зокрема, кирилиця в UTF-8 записується в діапазоні кодів 0410-0474, проте літера Є має шістнадцятковий код 0404, І – 0406, Ї – 0407, Ґ – 0490, ґ – 0491. Це можна перевірити за допомогою вбудованої функції ord(), яка повертає код переданого символу:

print(hex(ord('Є'))) #0x404  
print(hex(ord('І'))) #0x406  
print(hex(ord('А'))) #0x410  
print(hex(ord('Б'))) #0x411

Тому сортування імен може мати дещо несподіваний ефект:

print(sorted(names))

даватиме вивід

['Єлизавета', 'Ірина', 'Владислав', 'Вікторія', 'Олег', 'Олександр', 'Сергій']

У функції sorted є необов’язкові параметри key (функція, яка визначатиме критерій сортування) та reverse (порядок сортування від меншого до більшого – False, від більшого до меншого – True). Передача функції в якості параметру іншої функції є проявом можливостей функціонального програмування, мова про яке піде набагато пізніше. У якості частинного виправлення проблеми можна запропонувати свій власний критерій сортування. Для цього створимо окрему функцію, яка міститиме в собі український алфавіт та повертатиме індекс (номер в алфавіті) літери.

def ukrABC(ch):  
 abc = 'абвгґдеєжзиіїйклмнопрстуфхцчшщьюя'  
 return abc.index(ch)

Тепер можемо задати цю функцію як критерій сортування за допомогою параметру key:

print(sorted(names, key=lambda word: ukrABC(word[0].lower())))

Зверніть увагу, що в даному випадку сортування відбувається лише за першою літерою кожного з рядків. Також перші букви зводяться до маленьких літер, щоб відповідати записаному в функції ukrABC() алфавіті за допомогою методу lower(). Для передачі функції в якості параметру використовується лямбда-вираз, який вже стосується функціонального програмування та розглядатиметься пізніше. Результат роботи буде таким:

['Вікторія', 'Владислав', 'Єлизавета', 'Ірина', 'Олександр', 'Олег', 'Сергій']

З іншого боку, відсортувати імена можна, наприклад, за довжиною – функція len():

print(sorted(names, key=len)) #['Олег', 'Ірина', 'Сергій', 'Вікторія', 'Олександр', 'Владислав', 'Єлизавета']

Параметр key є необов’язковим параметром і для функцій min() і max().

print(min(names, key=len)) # Олег  
print(max(names, key=len)) # Олександр

Зверніть увагу, що максимальний елемент не є останнім у відсортованому за цим критерієм списку. Це перший елемент з максимальною довжиною. Тут маємо три дев’ятибуквенних імені, тому автоматично вони стають в порядку їх появи в первинному списку.

Для списків характерний метод sort(), який працює за аналогією до функції sorted(), проте має дещо іншу сигнатуру:

print("Початковий список: \t\t" + str(names))  
print(sorted(names, key=len))  
print("Список після sorted(): \t" + str(names))  
print(names.sort(key=len))  
print("Список після sort(): \t" + str(names))

Вивід такий:

Початковий список: ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']

['Олег', 'Ірина', 'Сергій', 'Вікторія', 'Олександр', 'Владислав', 'Єлизавета']

Список після sorted(): ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']

None

Список після sort(): ['Олег', 'Ірина', 'Сергій', 'Вікторія', 'Олександр', 'Владислав', 'Єлизавета']

Як бачимо, початковий список після застосування sorted() насправді не змінюється. На бажання можна зберегти відсортовану послідовність в іншу змінну. Проте метод sort() застосовується напряму до початкового списку та нічого не повертає (вивело None). Звідси, маємо перетворення первинного варіанту списку після сортування. Також зверніть увагу, що sort() визначений лише для списків, тоді як функція sorted() працює і з іншими колекціями.

За аналогічною ідеєю працюють функція reversed() і метод reverse() для зміни порядку елементів списку від кінця до початку:

print(list(reversed(names)))  
names.reverse()  
print(names)

Функція reversed() повертає спеціальний об’єкт-ітератор, тому результат потрібно явно звести до типу list().

Мова Python також пропонує методи pop() або remove() та ключове слово del для видалення елементів списку. Приклад роботи з методом pop():

print(names) # ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']  
print(names.pop()) # Олег  
print(names.pop()) # Єлизавета  
print(names) # ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета']  
print(names.pop(3)) # Вікторія  
print(names) # ['Олександр', 'Ірина', 'Сергій', 'Владислав', 'Єлизавета']  
print(names.pop(2)) # Сергій  
print(names) # ['Олександр', 'Ірина', 'Владислав']

Назва методу повторює назву аналогічної операції для стеку, тому принцип «останній прийшов – перший пішов» працює й тут: елементи видаляються з кінця списку. Проте при передачі індексу в метод видалення відбувається відносно початку списку з індексом 0. Якщо індекс виходить за межі списку, буде викинута відповідна помилка:

IndexError: pop index out of range

У результаті роботи методу повертається не змінений список, а лише видалений елемент. Водночас, метод remove() дозволяє видаляти елементи списку не за індексом, а за значенням:

print(names) # ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']  
print(names.remove("Ірина")) # None  
print(names) # ['Олександр', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']

Ключове слово del працює на більш глобальному рівні. Даний оператор мови програмування (інструкція) дозволяє видаляти не просто окремі елементи колекцій, а й цілі об’єкти загалом:

print(names) # ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']  
  
del names[2]  
print(names) # ['Олександр', 'Ірина', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']  
  
del names  
print(names) # NameError: name 'names' is not defined

***(Кортежі)*** Кортежі (tuple, вимовляється як «тапл») Працюють у мові Python так же, як і списки, проте за своєю природою є незмінюваними (immutable). Унаслідок цього методи для вставки та видалення елементів недоступні для кортежів, проте можна конкатенувати (поєднувати) кортежі за допомогою оператора «+»:

emptyTuple = ()  
someTuple = 78, 16  
namesTuple = tuple(names)  
  
print(hex(id(emptyTuple))) # 0x298dbb00040  
print(hex(id(someTuple))) # 0x298dc1cd300  
print(hex(id(namesTuple))) # 0x298dc1c99a0  
  
emptyTuple += namesTuple  
someTuple += namesTuple  
  
print(hex(id(emptyTuple))) # 0x298dc1c99a0  
print(hex(id(someTuple))) # 0x298dc196eb0

print(hex(id(namesTuple))) # 0x298dc1c99a0

Простежити за незмінюваністю кортежів можна за ідентифікаторами їх програмних об’єктів. Для демонстрації було створено три кортежі: порожній, з двома цілими значеннями (дужки при створенні не є обов’язковими) та перетворений зі списку з попереднього блоку. Всі вони мають різні ідентифікатори.

Тепер спробуємо конкатенувати останній кортеж з першими двома та переглянемо ідентифікатори сформованих об’єктів. З’єднання з порожнім кортежом призвело до того, що emptyTuple та namesTuple отримали однакові ідентифікатори. Це означає, що інтерпретатор оптимізував розміщення кортежів у пам’яті, оскільки зараз вони однакові. Проте за результатами конкатенації непорожніх кортежів утворився новий об’єкт someTuple, оскільки його ідентифікатор відрізняється від ідентифікаторів старих кортежів. Це означає, що незмінюваність формує наступну політику: при внесенні змін у початковий кортеж відбувається формування нового зміненого кортежу замість коригування його початкової версії.

Водночас, багато можливостей списків застосовні до кортежів. Зокрема, зрізи, функція len(), за умови сумісності типів елементів кортежу функції sorted(), min(), max(), а також інструкція del щодо всього об’єкта-кортежа.

***(Множини)*** Програмні об’єкти, подібні до математичних множин. Множини (sets) тримають в собі лише унікальні елементи, причому невпорядковано, на відміну від списків та кортежів. Подібно до співвідношення між списками й кортежами в контексті незмінюваності, існують звичайні (змінювані) та заморожені (незмінювані) множини.

Створити порожню множину можна тільки за допомогою конструктора set(), оскільки літерал «{}» використовується не лише множинами, а й словниками.

Операції з множинами в Python побудовані на принципах теорії множин:

* len(s) – розмір (кількість елементів) множини;
* in – оператор приналежності елемента множині;
* set1.isdisjoint(set2) – повертає True, якщо set1 та set2 не мають спільних елементів;
* set1 == set2 – рівність множин: всі елементи set1 належать set2, і навпаки;
* set1.issubset(set2) або set1 <= set2 – всі елементи set1 належать set2;
* set1.issuperset(set2) або set1 >= set2 – всі елементи set2 належать set1;
* set1.union(set2, ...) або set1 | set2 | ... – об’єднання кількох множин;
* set1.intersection(set2, ...) або set1 & set2 & ... – переріз кількох множин;
* set1.difference(set2, ...) або set1-set2-... – різниця множин, множина всіх елементів set1, які не належать жодним іншим set;
* set1.symmetric\_difference(set2); set1 ^ set2 – симетрична різниця множин, множина з елементів, які зустрічають лише в одній з множин set;
* set.copy() – копіює множину.

emptySet = set()  
namesSet = set(names)  
wordSet = set('hello')  
someSet1 = {6, 12, -5, 18, 7}  
someSet2 = {5, 7, 23, 56, 12, 11, 10}  
  
print(type({})) # <class 'dict'>  
print(names) # ['Олександр', 'Ірина', 'Сергій', 'Вікторія', 'Владислав', 'Єлизавета', 'Олег']  
print(namesSet) # {'Олександр', 'Ірина', 'Єлизавета', 'Олег', 'Владислав', 'Вікторія', 'Сергій'}  
print(wordSet) # {'h', 'l', 'o', 'e'}  
  
print("Олег" in namesSet) # True  
print(someSet1.union(someSet2)) # {5, 6, 7, 10, 11, 12, 18, 23, 56, -5}  
print(someSet1.intersection(someSet2)) # {12, 7}  
print(someSet1.difference(someSet2)) # {18, -5, 6}  
print(someSet1.symmetric\_difference(someSet2)) # {5, 6, 10, 11, 18, 23, 56, -5}

frozenSet = frozenset(someSet1)  
print(someSet1 == frozenSet) # True  
print(type(frozenSet)) # <class 'frozenset'>  
print(type(frozenSet - someSet2)) # <class 'frozenset'>  
print(type(someSet2 - frozenSet)) # <class 'set'>

Також для звичайних множин доступний набір операцій, які можуть безпосередньо змінювати ці множини.

* set.update(other, ...); set |= other | ... – об’єднання;
* set.intersection\_update(other, ...); set &= other & ... – переріз;
* set.difference\_update(other, ...); set -= other | ... – різниця;
* set.symmetric\_difference\_update(other); set ^= other – симетрична різниця;
* set.add(elem) – вставка елемента в множину;
* set.remove(elem) – видалення елемента з множини. KeyError, якщо такого елемента немає.
* set.discard(elem) – видаляє елемент, якщо він присутній у множині;
* set.pop() – видаляє перший елемент з множини. Через невпорядкованість множин спрогнозувати, яким саме буде цей елемент, важко;
* set.clear() – очистка множини.

***(Словники)*** Зберігають дані у вигляді пар «ключ-значення» (entry, item), де ключі (keys) виступають в якості унікальних ідентифікаторів значень (values) та спрощують їх пошук у великих наборах даних. За своєю суттю, словник без значень – це незмінювана множина, тому вони обидва використовують фігурні дужки для формування та представлення свого наповнення. Загальний синтаксис:

dictionary = {ключ : значення }

numbers = {'one':1, 'two':2, 'three':3}

Доступ до значення відбувається за його ключем. Водночас такий же синтаксис дозволяє здійснювати вставку нових пар за ключем

print(numbers['two']) #2

numbers['ninety'] = 90

Видалення елементів покладається на методи pop (повертає видалене значення), popitem (повертає кортеж «ключ, значення» видаленого елементу) та clear (очищає словник повністю):

my\_dict = {'First': 'Python', 'Second': 'Java', 'Third': 'Ruby'}  
a = my\_dict.pop('Third')   
print('Value:', a) # Value: Ruby  
print('Dictionary:', my\_dict) # Dictionary: {‘First’: ‘Python’, ‘Second’: ‘Java’}  
b = my\_dict.popitem()   
print('Key, value pair:', b) # Key, value pair: (‘Second’, ‘Java’)  
print('Dictionary', my\_dict) # Dictionary {‘First’: ‘Python’}  
my\_dict.clear()   
print('n', my\_dict) # {}

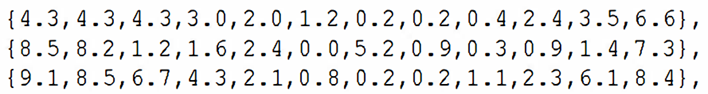
Поширені методи для отримання окремо наборів ключів, значень чи пар «ключ-значення»

print(my\_dict.keys()) # dict\_keys(['First', 'Second', 'Third'])  
print(my\_dict.values()) # dict\_values(['Python', 'Java', 'Ruby'])  
print(my\_dict.items()) # dict\_items([('First', 'Python'), ('Second', 'Java'), ('Third', 'Ruby')])

**Практичні завдання**

Оберіть задачі для виконання в сумі до 3 балів. ***Після здачі задачі шістьма особами з підгрупи подальша її подача іншими учасниками підгрупи не приймається!***

1. *0,2 бала* При аналізі даних, зібраних у результаті експерименту, часто може виникати потреба у видаленні найбільш екстремальних (занадто великих чи малих) значень до виконання обчислень. Ваша програма має видалити зі списку на n (ціле додатнє число, введене користувачем) елементів по троє найбільших та найменших значень і вивести решту значень (порядок не важливий). Також забезпечте коректну обробку ситуації, коли у списку не вистачатиме елементів для видалення.
2. *0,2 бала* Напишіть програму, що буде обчислювати ітогові дані по роках, щорічні та щомісячні середні значення опадів за період у декілька років





1. *0,2 бала* Напишіть програму, яка зчитує від користувача слова, кожне з нового рядка. Ввід закінчується порожнім рядком, після чого виводиться перелік всіх унікальних слів, які були введені. Наприклад, для вводу

first

second

first

third

second

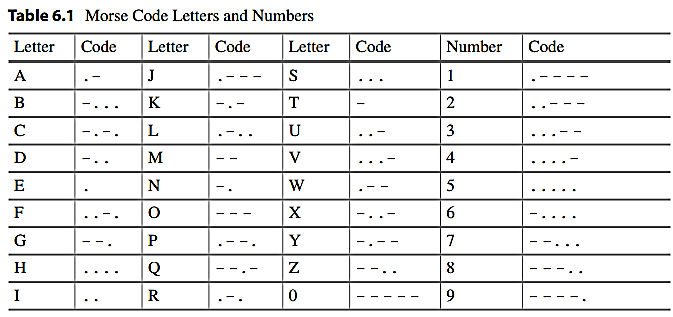
програма може вивести

first

second

third

1. *0,3 бала* Код Морзе – схема кодування, яка використовує точки та тире для представлення цифр і букв. Напишіть програму, яка буде використовувати словник для зберігання відображення букв і цифр на код Морзе (таблиця).



Ваша програма має читати повідомлення користувача та перетворювати його в код Морзе з наступним виводом закодованого тексту. Пробіли між словами слід залишати, а решту символів, що не є цифрою або буквою – ігнорувати. Наприклад, Hello, World! буде мати код

**.... . .-.. .-.. --- .-- --- .-. .-.. -..**

1. *0,3 бала* Напишіть додаток, який використовує генерування випадкових чисел для створення речень. Використовуйте 4 списки рядків: article, noun, verb та preposition. Створіть речення, обравши з кожного масиву слова в такому порядку: артикль, іменник, дієслово, прийменник, артикль та іменник. Після вибору кожного слова виконайте його конкатенацію з реченням. Слова мають відокремлюватись пробілами. Загальне речення має починатись з великої літери та закінчуватись крапкою. Додаток має згенерувати та відобразити 10 речень.

*Артиклі*: "the", "a", "one", "some" та "any";

*Іменники*: "boy", "girl", "dog", "town" та "car";

*Дієслова*: "drove", "jumped", "ran", "walked" та "skipped";

*Прийменники*: "to", "from", "over", "under" та "on".

1. *0,3 бала* На більшості кнопочних телефонів для набору текстового повідомлення використовують цифрову клавіатуру. Оскільки кожна кнопка прив’язана до кількох букв, її потрібно натискати декілька разів для вибору потрібної відповідно до таблиці. Натиснення цифри 2, 3, 4 або 5 разів генерує другу, третю, четверту або п’яту букву для цієї кнопки.

|  |  |
| --- | --- |
| **Кнопка** | **Символи** |
| 1 | . , ? ! : |
| 2 | A B C |
| 3 | D E F |
| 4 | G H I |
| 5 | J K L |
| 6 | M N O |
| 7 | P Q R S |
| 8 | T U V |
| 9 | W X Y Z |
| 0 | пробіл |

Напишіть програму, яка показує, скільки та яких натиснень кнопок потрібно, щоб набрати текст користувача. Сконструюйте словник, який відображає кожну букву чи символ на кнопку, яку потрібно натиснути. Наприклад, для тексту Hello, World! програма має вивести 4433555555666110966677755531111. Забезпечте коректну роботу як з малими, так і великими літерами. Ігноруйте символи, не представлені в таблиці.

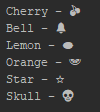
1. *0,3 бала* Багато фінансових документів передбачають запис суми грошей як у числовій, так і в текстовій формі. Напишіть програму, яка буде конвертувати суми від 0 до 10 000 грн. у текстове представлення. Наприклад, 3726 грн – три тисячі сімсот двадцять шість гривень. Використовуйте списки для переліку можливих текстових представлень для одиниць, десятків, сотень та тисяч гривень.
2. *0,5 бала* Створіть плейліст, кожна пісня якого має назву, виконавця, альбом та тривалість. Плейліст дозволяє додавати та видаляти пісню, отримувати розмір та очищати його, а також виконувати форматований вивід композицій і повну тривалість його відтворення.
3. *0,3 бала* Невелика авіакомпанія потребує систему автоматизації резервування місць у літаку на 12 місць. Ваша програма має пропонувати наступні альтернативи:

Натисніть 1 для першого класу (місця 1-6)

Натисніть 2 для економного класу (місця 7-12).

Потім Ваш застосунок повинен відображати посадкові місця, вказуючи на замовлене місце та його клас.

Використовуйте список булевих значень для представлення посадкових місць у літаку. Ваш застосунок має ніколи не давати резервувати вже зарезервовані місця, а повинен видавати перше вільне місце в класі. Коли місця з економного класу закінчились, програма повинна запропонувати квиток першого класу. Якщо таке не підходить, програма виведе повідомлення «Наступний виліт через 3 години».

1. *0,4 бала* Напишіть гру «Однорукий бандит» для консолі. У процесі будуть відображатись три випадкових символи з переліку:

Cherry - u"\U0001F352",

Bell - u"\U0001F514",

Lemon - u"\U0001F34B",

Orange - u"\U0001F34A",

Star - u"\u2606",

Skull - u"\U0001F480"

Гравцю видається кредит у розмірі 100 грн., а кожний запуск коштує 5 грн. Якщо «однорукий бандит» показує два однакових символи, користувач виграє 10 грн, а якщо три – 25 грн. У випадку випадіння трьох дзвіночків (bells) сума виграшу складає 100 грн. Якщо випадає два черепи – гравець додатково втрачає 5 грн, а при отриманні трьох черепів вся сума згорає. Гравець може зняти вигране до початку нової спроби або продовжувати грати, поки на рахунку будуть гроші.

1. *0,4 бала* Напишіть програму, яка зчитує дату від користувача та обчислює дату наступного дня. Дата буде вводитись у числовій формі трьома окремими виразами (рік, місяць, день). Наприклад,
2. якщо користувач вводить значення, яке представляє день 2017-05-10, Ваша програма має показати повідомлення, яке говоритиме, що наступним днем буде 2017-05-11.
3. Якщо користувач вводить значення на зразок 2017-04-30, програма має вказати, що наступним днем буде 2017-05-01.
4. Якщо користувач вводить значення, що представляє дату 2016-12-31, програма має вивести наступний день - 2017-01-01.
5. Забезпечте коректну роботу для високосних років.

Не використовуйте вбудованих методів для обчислення наступного дня!

1. *0,3 бала* Число Капрекара – це невід’ємне ціле число, квадрат якого в цій системі можна розбити на дві частини, сума яких дає початкове число. Наприклад, 45 є числом Капрекара, оскільки 452 = 2025, причому 20 + 25 = 45. Напишіть функцію, яка отримуватиме число та повертатиме дві частини, якщо це число Капрекара. Інакше функція поверне -1. Протестуйте роботу функції на різних числах.
2. *0,2 бала* Генерування паролю шляхом вибору випадкових символів дає відносно безпечний, проте загалом складний для запам’ятовування пароль. Альтернативним способом є конструювання паролю з двох англійських слів з подальшою конкатенацією. Такий пароль простіше запам’ятати, хоча і зламати також.

Напишіть програму, яка містить список слів, обирає два випадковим чином та з’єднує їх в одне. При створенні паролю переконайтесь, що загальна його довжина знаходиться в діапазоні від 8 до 10 символів, а кожне використане слово складається принаймні з трьох букв. Зробіть так, щоб у паролі кожне слово починалось з великої літери та виведіть пароль для користувача.

1. *0,3 бала* Ціле число n називають досконалим, якщо сума всіх його дільників дорівнює цьому числу. Наприклад, 28 є досконалим, оскільки його дільники 1, 2, 4, 7 та 14 у сумі дають 28. Створіть функцію, яка перевірятиме, чи є число досконалим. Напишіть програму, яка визначатиме досконалі числа з проміжку від 1 до 10 000 і записуватиме їх у список. Після цього список потрібно вивести на екран.
2. 0,5 бала Редакційна відстань (edit distance) між двома рядками є мірою їх схожості. Чим менша редакційна відстань, тим більше схожі між собою рядки відносно мінімальної кількості вставок, видалень та замін при перетворенні одного рядка в інший.

Наприклад, рядки kitten та sitting. Перший рядок можна перетворити в другий, якщо: замінити k на s, замінити e на i, вставити g в кінець рядка. Три – це найменша кількість операцій, що можна виконати для перетворення kitten у sitting.

Напишіть рекурсивну функцію, яка обчислює редакційну відстань між 2 рядками. Використовуйте наступний алгоритм:

Візьмемо рядки s і t

If довжина s = 0

Повернути довжину t

Else if довжина t = 0

Повернути довжину s

Else

cost = 0

If останній символ в s не дорівнює останньому символу в t

cost = 1

d1 = (редакційна відстань між всіма символами, крім

останнього в s та всіма символами в t) + 1

d2 = (редакційна відстань між всіма символами в s та всіма

символами в t, крім останнього) + 1

d3 = (редакційна відстань між всіма символами, крім

останнього, в s та всіма

символами в t, крім останнього) + 1

Return min(d1, d2, d3)

Використайте дану рекурсивну функцію, щоб написати програму, яка зчитує 2 рядки від користувача і виводить редакційну відстань між ними.

1. *0,2 бала* ***(Спискове включення)*** Функція foo(C) потрібна, щоб повернути «зворотну» версію символьного рядка:

>>> foo("hawk")

'kwah'

>>> foo("amazing")

'gnizama'

Напишіть таку функцію відповідно до наступної стратегії. Спочатку перевірте, чи можете написати range-вираз, який проходить від індексу останньої букви до першого індексу. Наприклад, для рядка "thousand" range-вираз має повернути [7,6,5,4,3,2,1,0]. Використайте цей вираз, щоб зробити власний список, який виконуватиме схожий код:

[ C[i] for i in range( . . . ) ]

Це допоможе спростити foo(C), яка повертатиме цей кастомний список. Не використовуйте reverse(). Залишиться перетворити список символів у рядок. Використовуйте для цього метод join().

1. *0,3 бала* Секретна інформація часто редагується чи видаляється з документів перед їх публікуванням. Після публікації документів відредагований текст часто заміняється чорними прямокутниками.

Напишіть функцію, яка редагує всі секретні слова, заміняючи їх зірочками, навіть якщо слово з’являється всередині іншого слова. Передбачено існування списку секретних слів. Виведіть відредаговану версію початкового тексту на екран.

Корисним може бути метод replace() для роботи з рядками. Доповніть програму незалежністю від регістру літер у тексті, тобто слова exam, Exam, ExaM та EXAM будуть вважатись однаковими.

1. *0,2 бала* Напишіть програму, яка зчитує числа від користувача, поки не буде введено порожній рядок. Далі вона повинна знайти середнє значення із введених чисел та вивести 2 списки: введених чисел, що не перевищують середнє значення та, відповідно, більше або рівні за нього.
2. *0,2 бала* Нехай грає музика! Запрограмуйте комп’ютер на відтворення мелодії, заданої окремими списком частот та списком тривалостей відтворення відповідної частоти. Відповідні пари значень наведені нижче:

(659,250), (659,250), (659,300), (523,250), (659,250), (784,300), (392,300), (523,275), (392,275), (330,275), (440,250), (494,250), (466,275), (440,275), (392,275), (659,250), (784,250), (880,275), (698,275), (784,225), (659,250), (523,250), (587,225), (494,225).

Для підключення біперу потрібно імпортувати модуль winsound і викликати функцію

winsound.Beep(частота у Гц, тривалість у мс)

1. *0,2 бала* Якщо Вам сподобалась попередня задача, спробуйте тепер іншу мелодію, збережену в *двовимірному* масиві (списку списків):

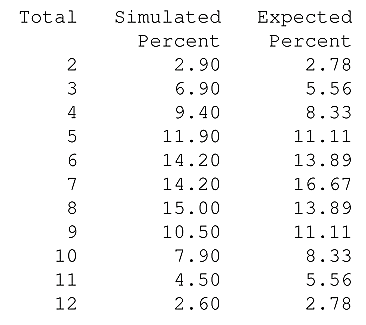
(440,500), (440,500), (440,500), (349,350), (523,150), (440,500), (349,350), (523,150), (440,1000), (659,500), (659,500), (659,500), (698,350), (523,150), (415,500), (349,350), (523,150), (440,1000).

1. *0,3 бала* Звичайна колода карт має 52 карти, кожна з яких описується мастю і значенням. Мастями можуть бути піки (spades), черви (hearts), трефи (diamonds) та бубни (clubs), а значеннями – числа від 2 до 10, валет (Jack), дама (Queen), король (King) і туз (Ace). Приклади позначень карт описані в таблиці.

|  |  |
| --- | --- |
| **Карта** | **Абревіатура** |
| Піковий валет (Jack of spades) | Js |
| Трефова четвірка (Four of clubs) | 4c |
| Бубнова дев’ятка (Nine of diamonds) | 9d |
| Чирвова десятка (Ten of hearts) | Th |

Напишіть програму, яка буде створювати колоду карт у вигляді списку. Перетасуйте колоду відповідно до алгоритму Сатолло, представленого [тут](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D1%81%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A4%D0%B8%D1%88%D0%B5%D1%80%D0%B0_%E2%80%94_%D0%99%D0%B5%D1%82%D1%81%D0%B0). Виведіть початкову колоду та перетасовану на екран.

1. *0,3 бала* Створивши та перетасувавши колоду в попередніх задачах, прийшов час здати їх гравцям. Кількість гравців та карт для кожного з них залежить від гри, тому користувач повинен після перетасовки ввести кількість карт на одного гравця та кількість гравців. За цими даними потрібно вивести інформацію зі списку про гравців та карти, що їм були здані. Забезпечте коректну роботу програми, коли карт для гравців потрібно більше, ніж є в колоді.
2. *0,3 бала* Напишіть програму, яка симулюватиме 1000 кидків пари костей та виводитиме статистику сум, які випадають у відсотках. Оформити результат можна так:



Очікуваний відсоток – стандартне значення, отримане з [розподілу Гауса](https://uk.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D1%80%D0%BE%D0%B7%D0%BF%D0%BE%D0%B4%D1%96%D0%BB).

1. *0,3 бала* Решето Ератосфена – алгоритм, який дозволяє знайти прості числа, що не перевищують деякого n. Він був придуманий понад 2000 років тому і описується так:

Запишіть всі числа від 0 до n

Викресліть 0 і 1, тому що вони не є простими

Задайте p = 2

While p < n do

Викреслити всі числа, кратні p (проте не сам p)

Перейти (змінна p) до наступного числа в списку, яке

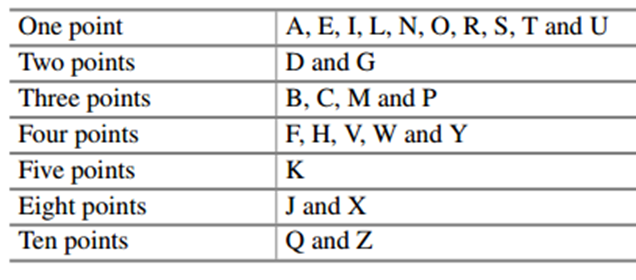
не буде закресленим

Вивести всі незакреслені числа.

Сформуйте список усіх чисел від 2 до введеного значення n, а потім видаляйте з нього всі числа, які викреслює алгоритм. У результаті повинен отриматись список із простих чисел для введеного діапазону.

Підказка. Пришвидшити роботу початкового алгоритму можна за рахунок неповного перебору чисел. Насправді, достатньо перевірити числа не до n, а лише до .

1. *0,3 бала* У грі Scrabble™ кожна буква оцінюється балами, як показано в таблиці. Загальна оцінка слова – це сумарна кількість очків за кожну букву:



Напишіть програму, яка обчислює та виводить Scrabble™-оцінку для слова. Створіть словник, який відображає букви на бальні значення, а потім з його допомогою аналізуйте введені слова, виводячи їх оцінку в балах.

У грі Scrabble™ включено деякі squares, які домножають значення букви чи всього слова на певні значення. Це правило в задачі ігнорується.

1. *0,4 бала* Припустимо, на залізничному вокзалі присутня автоматизована система з управління камер зберігання для багажу пасажирів. Інформація про камери зберігається у вигляді переліку «Прізвище пасажира-час здачі багажу-час видачі багажу». На вокзалі присутні 12 комірок, пасажиру відкривається вільна комірка з мінімальним номером.

Вважаємо, що робота програми обмежується одним днем, а час здачі та видачі багажу вводить сам пасажир. Для роботи з програмою спочатку треба ввести кількість пасажирів, а потім їх дані, наприклад, так:

3

Іваненко 09:45 12:00

Петренко 10:00 11:00

Сидоренко 12:00 13:12

На екран має бути виведено, яку комірку відкривати кожному з пасажирів.

Іваненко 1

Петренко 2

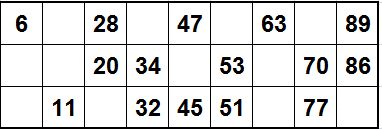
Сидоренко 1

1. *0,3 бала* Введіть з клавіатури деякий текст та підрахуйте в ньому кількість слів, букв, цифр,малих літер, великих літер, пробілів.
2. *0,3 бала* Лінійна регресія дозволяє побудувати пряму, яка найкраще наближає набір з n точок даних (data points). Це дає можливість, наприклад, встановити тренд цін на біржі в деякому проміжку часу. Кожна точка з набору має координати (x; y) – хвилину з початку роботи біржі та ціну за акцію. Символами та позначаємо середнє значення по окремим координатам. Лінія тренду представляється рівнянням , де a і b обчислюються за формулами

Напишіть програму, яка зчитує колекцію точок від користувача покоординатно, поки не буде введено порожній рядок. У результаті потрібно вивести формулу для прямої, наприклад, y = 0.95x + 0.1 для координат (1, 1), (2, 2.1) та (3, 2.9).

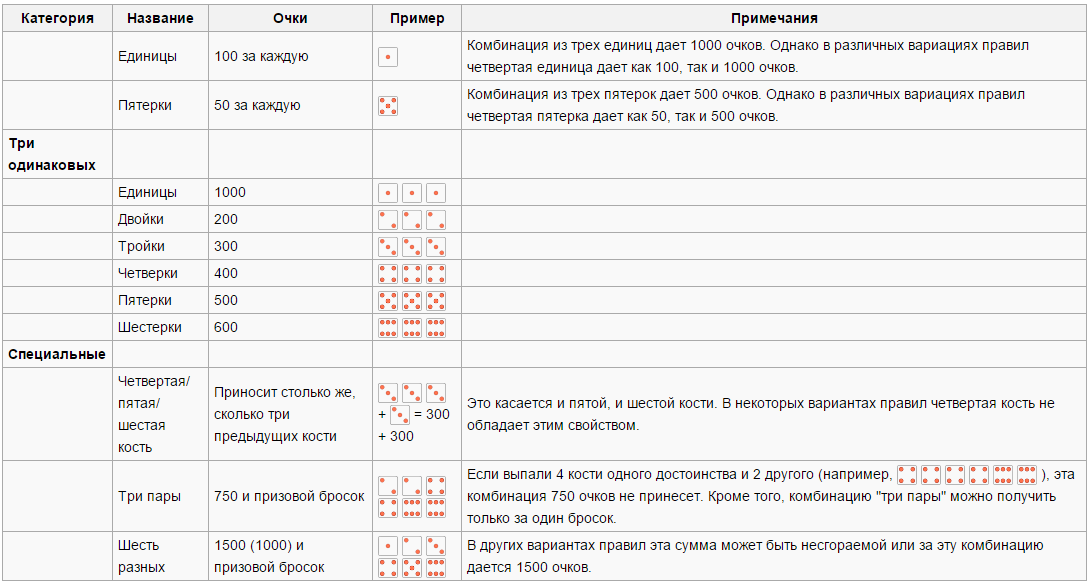
1. *0,5 бала* Виграшна картка лото містить лінію з 5 чисел, які до цього були витягнуті з мішка. Гравці закривають витягнуті числа фішками. У нашому додатку будемо заміняти число на 0 у лото-словнику, якщо воно було оголошено.

Напишіть програму, яка працює зі словниками, що представляють картки лото. Якщо картка містить рядок з 5 нулів, вона вважається виграшною. Програма має формувати 2 картки лото (9 стовпців чисел, кожен з яких відповідає десятку числа, 3 рядка, по 5 чисел у кожному), генерувати випадкове число та заново виводити картку після цього. Як тільки якась із карток виграє, програма завершує роботу з повідомленням про виграш відповідного гравця.



Підказка. Оскільки всі числа додатні, знаходження 5 нулів на картці відповідає задачі рівності суми 5 входжень нулю. Можливо, Вам так буде простіше моделювати гру.

1. *0,5 бала* Для гри в зонк потрібно 6 шестигранних кубиків, які будуть кидатись одночасно. Доступні наступні комбінації:



Напишіть підпрограму, яка аналізує кидок одного гравця. При цьому генерується 6 випадкових чисел (від 1 до 6). Виведіть на екран доступні комбінації за результатом кидка та очки, які за них набере користувач. Спеціальні комбінації можна не розглядати.

1. *0,4 бала* Нижче наведена таблиця описує октави музичних нот, починаючи з middle C (С4) разом з їх частотами.

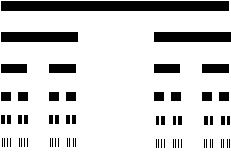
|  |  |
| --- | --- |
| **Нота** | **Частота (Гц)** |
| C4 | 261,63 |
| D4 | 293,66 |
| E4 | 329,63 |
| F4 | 349,23 |
| G4 | 392,00 |
| A4 | 440,00 |
| B4 | 493,88 |

Програма має коректно виводити частоти для нот всіх октав – від C0 до C8. Для того, щоб їх знайти, потрібно враховувати наступне правило: частота ноти в октаві n – половина частоти ноти в октаві n+1. Додаток має підтримувати 2 варіанти роботи:

1) На вхід програми має надходити нота відповідної октави, а на виході виводитись її частота.

2) За введеною частотою визначити ноту.

*Підказка: для виконання завдання потрібно виділити окремо символи з двосимвольної назви ноти. Далі можна визначити частоту ноти в четвертій октаві, а потім поділити її на 24−x, де x – номер октави, введеної користувачем.*

1. *0,4 бала* (Пил Кантора) Множину Кантора отримують шляхом поділу відрізку на 3 частини та видалення середньої з них. Напишіть рекурсивну функцію, що дозволятиме відобразити аналогічне зображення в консольному додатку. Для візуалізації -того рівня множини Кантора використовують правило n = 0: на початку для [0, 1];

n > 0: застосувати правило n разів

1. *0,3 бала* Створіть програму, яка підраховує суму всіх введених користувачем чисел, пропускаючи некоректно введені (нечислові) дані. Програма має виводити поточну суму після кожного введеного числа. При вводі нечислових даних виводиться повідомлення про помилку, проте робота програми продовжується. Вихід з програми здійснюється після вводу порожнього рядка. Переконайтесь, що програма працює коректно як для цілих, так і дробових значень.

Підказка: вправа вимагає використання винятків.

1. *0,3 бала* Один з перших відомих прикладів шифрування використовував Юлій Цезар. Йому потрібно було надсилати письмові інструкції своїм генералам, проте він не хотів, щоб вороги дізнались про його плани, якщо повідомлення потрапить їм у руки. Тому Цезар запропонував просту ідею: кожна літера первинного повідомлення зсувалась на 3 позиції в алфавіті. Замість A записували D, замість B – E, C ставало F, D – G і т. д. Останні 3 букви в алфавіті мали замінники з його початку (X = A, Y = B, Z = C). Небуквенні символи не зашифровувались.

Напишіть програму, яка реалізує шифр Цезаря. Дозвольте користувачеві ввести повідомлення та розмір зсуву, а потім виведіть «зсунуте» повідомлення. Переконайтесь, що Ваша програма коду як великі, так і маленькі букви. Додаток також має підтримувати від’ємний зсув, щоб мати можливість як зашифрувати, так і розшифрувати повідомлення.

1. *0,6 бала* Реалізуйте гру «Хрестики-нулики». Крім головної функції, виділіть окремі підпрограми для
   * + ініціалізації матриці 3х3 «порожніми» символами,
     + обробки ходу гравця,
     + обробки ходу комп’ютера,
     + перевірки виграшу,
     + виводу результатів.

Нехай комп’ютер під час свого ходу заповнює першу «порожню» клітинку, а гравець – задає координати (X,Y), які треба перевірити на валідність.

1. *0,5 бала* Подумайте про Ваших одногрупників, які зараз знаходяться в кабінеті. Напишіть повідомлення, яке вітається з кожним з них. Збережіть їх імена та прізвища у списку та виведіть привітання кожного по цьому списку в алфавітному порядку за іменем. Також надрукуйте повідомлення, звертаючись до кожного в алфавітному порядку за прізвищем.

Зверніть увагу на дні народження одногрупників зі списку. Виведіть на екран їх імена та дні народження від наймолодшого до найстаршого. Якщо день народження когось зі списку відбудеться в наступні 30 днів від нинішньої дати, покажіть спеціальне повідомлення для цієї особи. Збережіть інформацію про людей у словнику, якщо досі цього не зробили.

Примітка. Не забувайте, що для [сортування](http://python-3.ru/page/sorted) може використовуватися

* метод sort(), який переставляє елементи в самій структурі даних
* метод sorted(), який повертає відсортовану копію

фунція attrgetter(), яка генерує функцію, що бере об’єкт і повертає певний атрибут з цього об’єкта.

1. *0,3 бала* Інформація щодо пакету акцій зберігається в наборі атрибутів – даті придбання, ціні придбання кількості та символу на біржі. Ми можемо записати цю інформацію в кортеж для кожного пакету акцій та виконувати з цими пакетами прості операції. Нехай у нас є портфоліо:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Дата придбання** | **Ціна придбання** | **Кількість** | **Символ на біржі** | **Поточна ціна** |
| 25 Jan 2018 | 43.50 | 25 | CAT | 92.45 |
| 25 Jan 2018 | 42.80 | 50 | DD | 51.19 |
| 25 Jan 2018 | 42.10 | 75 | EK | 34.87 |
| 25 Jan 2018 | 37.58 | 100 | GM | 37.58 |

Ми можемо представити цю інформацію у вигляді списку кортежів

portfolio= [ ( "25-Jan-2018", 43.50, 25, 'CAT', 92.45 ),

( "25-Jan-2018", 42.80, 50, 'DD', 51.19 ),

( "25-Jan-2018", 42.10, 75, 'EK', 34.87 ),

( "25-Jan-2018", 37.58, 100, 'GM', 37.58 )

]

Напишіть функцію, яка переглядає кожен пакет акцій та множить кількість на ціну їх покупки, щоб визначити загальну ціну портфоліо на момент придбання. Реалізуйте аналогічну функцію для обчислення поточної вартості портфоліо.

1. *0,6 бала* Реалізуйте задачі 21-22 про колоду карт з застосуванням об’єктно-орієнтованої парадигми програмування. Структуру даних, що представляє колоду карт, називають деком (двоголовою чергою). Це дає змогу брати карти як згори колоди, так і знизу. Ознайомтесь з [реалізацією деку](http://aliev.me/runestone/BasicDS/ImplementingaDequeinPython.html) в Python та сформуйте класи, що описуватимуть карту, колоду та гравця. Здайте карти за правилами гри «Дурень» та виведіть карти з рук гравців на екран.
2. *0,6 бала* ***(Задача на структури даних)*** Змоделюйте чергу в ресторані самообслуговування. Час обслуговування генеруватиметься випадковим чином у заданих далі діапазонах. Сумарний час замовлення їжі складається з трьох частин:
3. *Підбору свого меню* – збиранні на підносі бажаних страв. Нехай існує великий стіл зі стравами, які пропонує заклад харчування, причому кількість порцій цих страв обмежена. Вважатимемо, що людина витрачає від 2 до 5 секунд на підбір однієї страви для себе, кількість таких страв обмежена до 6. Зібравши свій піднос, клієнт стає в чергу.
4. *Очікування в черзі*, поки ще не обслужені клієнти попереду.
5. *Обслуговування на касі*, яке може займати від 5 до 35 секунд.

Створіть чергу з 12 клієнтів, кожен з яких має власне меню. Виведіть по кожному з них наступну інформацію:

1. Номер клієнта
2. Номер клієнта у черзі
3. Час, коли клієнт став у чергу
4. Тривалість очікування на розрахунок за страви
5. Тривалість обслуговування на касі
6. Повний час обслуговування

Наприклад, у такому форматі:

# клієнта | # у черзі | Час | Тривалість | Тривалість | Час |

----------|-----------|-----|------------|------------|-----|

4 | 6 |00:21| 25 секунд | 31 секунда |01:16|

**Об’єктно-орієнтоване програмування**

1. *0,3 бала* ***(Класи та об’єкти)*** Класичною вправою для людей, які вчаться писати класи є моделювання особи (***клас Person***). Ми знайомі з рисами та поведінкою людей, тому визначте клас Person(), в якому функція \_\_init()\_\_ встановлює кілька атрибутів для особи. Тут можна розглядати ім’я, вік, місце народження та ін. інформацію, що хочеться знати про людину.

Напишіть метод, який буде представляти цю людину, наприклад, виводити привітання «Привіт, мене звати \_\_\_\_\_\_\_\_\_\_». Також можете доповнити клас методом, який буде збільшувати вік особи на 1. У більш складній версії цього методу потрібно зберігати дату народження, а не вік, а потім обчислювати вік як різницю між поточною датою та датою народження.

Створіть об’єкт-особу, заповніть інформацію про неї та виведіть її на екран.

1. *0,3 бала* ***(Класи та об’єкти)*** Створіть клас Ship, у якому визначено метод \_\_init\_\_(), який встановлює для кожного об’єкта цього класу початкові значення координат (0; 0). Також у ньому записано метод move\_up(), який збільшує y-координату корабля на 1.

Створіть корабель (об’єкт класу Ship) та виведіть на екран інформацію про об’єкт та його у-координату. Пересуньте корабель вгору та знову виведіть його у-координату.

Створіть корабельний флот з 6 кораблів, кожен з яких знаходиться в різних координатах та покажіть, що кожен корабель – це окремий об’єкт. Перемістіть кілька з них.

Доповніть клас Ship методом, який буде обчислювати відстань між кораблями. Виведіть відстані від першого корабля флотилії до решти кораблів.

Придумайте та введіть у клас додаткові атрибути, що описують корабель, наприклад, довжину корпусу корабля, розмір команди, назву корабля, його максимальну швидкість тощо. Виведіть розширений опис наявних кораблів.

1. *0,4 бала* ***(Наслідування)*** Якщо Ви хочете змоделювати космічний корабель, можна написати абсолютно новий клас. Проте космічний корабель є спеціальним видом корабля, тому можна успадкувати атрибути та поведінку класу Ship, а потім доповнити наш новий клас додатковими атрибутами та методами, що стосуються космічного корабля.

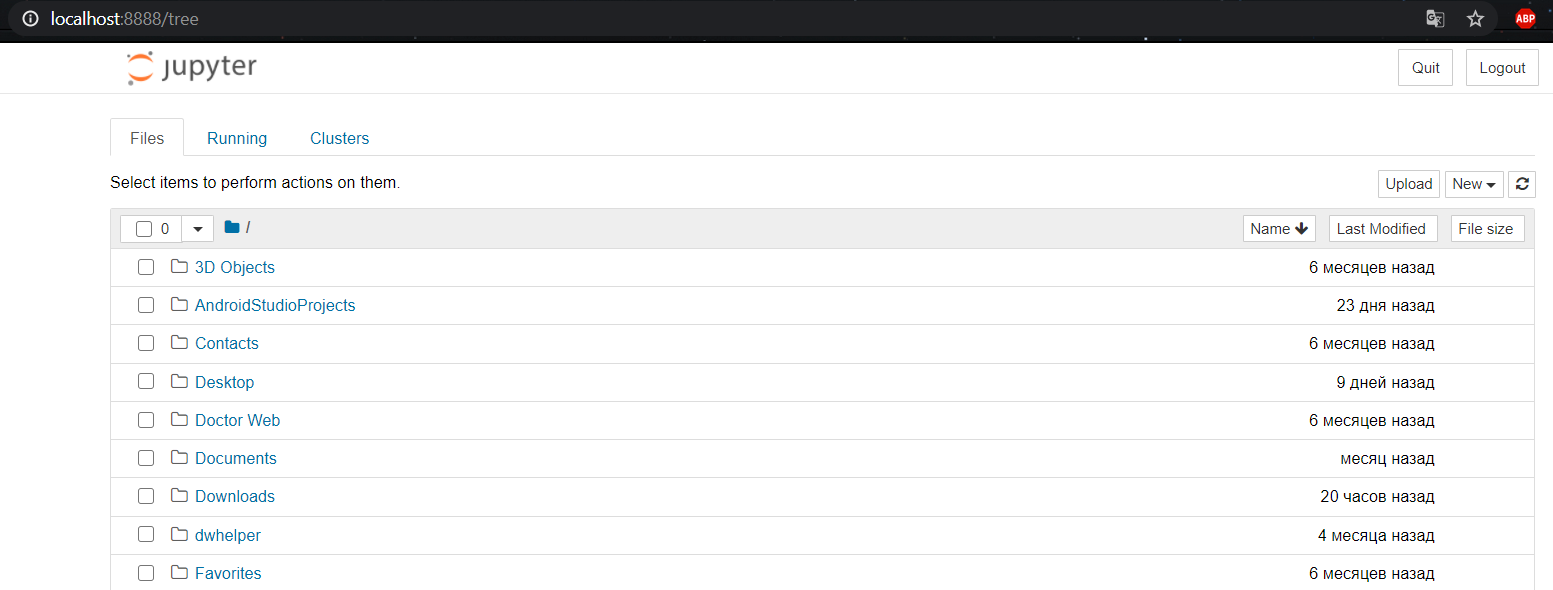
У фантастиці розглядають кілька понять для космічних суден, зокрема [зореліт](https://uk.wikipedia.org/wiki/%D0%97%D0%BE%D1%80%D0%B5%D0%BB%D1%96%D1%82) (starship, starcraft або interstellar spacecraft) та [космічний апарат](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D1%81%D0%BC%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BF%D0%B0%D1%80%D0%B0%D1%82) (Spacecraft). Спрощено кажучи, зореліт є підкатегорією космічних апаратів. Таким чином, можна вибудувати ієрархію наслідування: Корабель – Космічний апарат – Зореліт. Реалізуйте цю ієрархію наслідування у вигляді відповідних класів. Створіть об’єкти кожного із запропонованих класів.

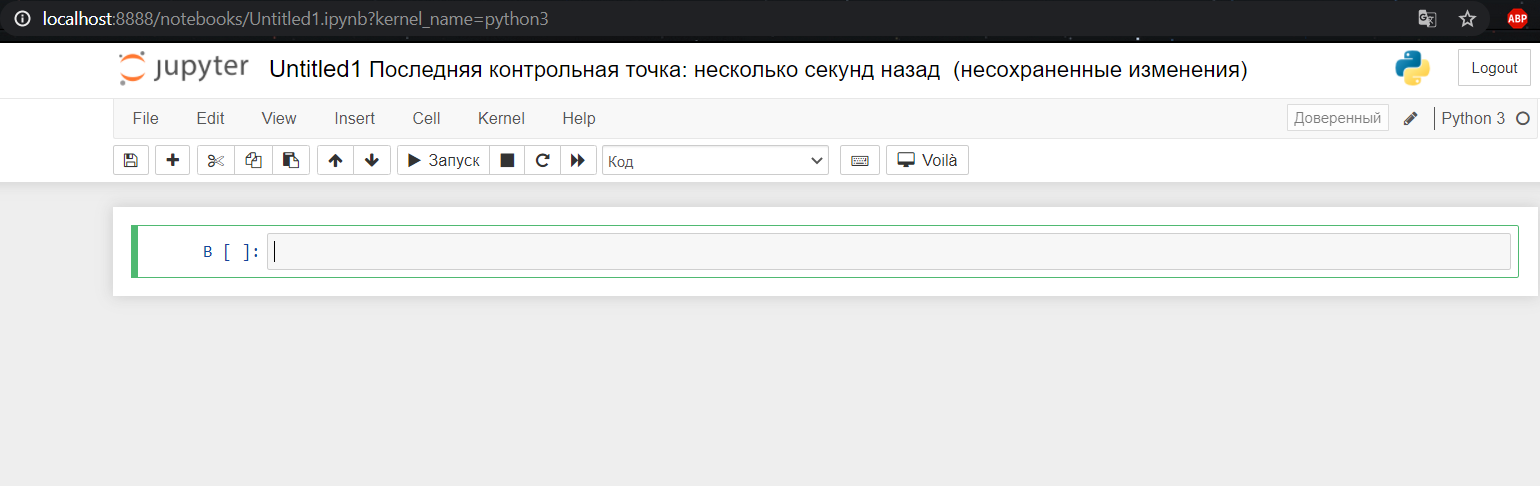
1. *0,8 бала* ***(Ієрархія наслідування)*** Подумайте над тим, що кожен корабель може мати свою спеціалізацію. Знайдіть класифікацію кораблів та спробуйте перенести її в код. Зокрема, це можуть бути транспортні судна, екскурсійний лайнер, дослідницький або бойовий корабель. Побудуйте схему, що відображатиме, на Вашу думку, класифікацію кораблів, в основі якої буде лежати базове поняття «корабель». Додайте діаграму класів до звіту. Виділіть для кожного елементу схеми атрибути та методи, що будуть характерні саме для нього та його дочірніх елементів. Спробуйте перенести дану схему в код.
2. *0,2 бала* ***(Модулі та класи)*** Класи з часом розростаються новим функціоналом, тому з ними стає важче працювати. Мова Python дозволяє зберігати Ваші класи в окремий файл – *модуль*, а потім імпортувати його до Вашої програми. Створіть окремий модуль, що міститиме вище описані класи, а потім імпортуйте його до Вашої програми, в яку перенесеться код роботи з об’єктами цих класів.

**ІНДЗ**

1. *0,6 бала* ***(Jupyter Notebook)*** Ознайомтесь з можливостями роботи з інструментом Jupyter Notebook. Для початку роботи Ви маєте кілька варіантів його інсталяції:

* Встановити [окремий інструмент JupyterLab](https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html). Відповідно до цього способу Ви працюватимете в браузерному режимі з окремими блокнотами:





* [налаштувати](https://www.jetbrains.com/help/pycharm/editing-jupyter-notebook-files.html#create-notebook-file) підтримку JupyterLab у PyCharm Professional (безкоштовна версія Community Edition не підтримує роботу з блокнотами Jupyter, починаючи з 2019.х). Майте на увазі, що версія Professional працюватиме в trial-режимі протягом визначеного періоду. Або розгляньте можливість отримання [безкоштовної ліцензії](https://www.jetbrains.com/ru-ru/community/education/#students) інструментів JetBrains для студентів, зокрема за програмою GitHub Student Developer Pack;
* [Налаштувати](https://docs.anaconda.com/anaconda/user-guide/tasks/use-jupyter-notebook-extensions/) підтримку Jupyter Notebook в Anaconda Navigator.

Дослідіть питання практичної потреби та використання даного інструменту, опишіть у звіті переваги від його використання, а також продемонструйте інсталяцію та інтерактивну роботу з блокнотом Jupyter (файл .ipynb) на власній машині. За основу можете взяти [туторіал](https://www.dataquest.io/blog/jupyter-notebook-tutorial/).

1. *1 бал* ***(Комплексні основи ООП)*** Створіть новий клас Student, породжений класом Person. Визначте атрибути, які має лише студент, і не мають інші люди. Наприклад, студент відвідує навчальний заклад, має рік вступу, перелік поточних/пройдених курсів з оцінками, середній бал за кожну сесію (GPA) та ін.

Створіть об’єкт класу Student та покажіть, що наслідування виконується коректно. Задайте деяким атрибутам для студента значення, записані лише в класі Person або лише в класі Student. Виведіть значення цих атрибутів.

Доповніть архітектуру додатку класами, які описують кафедру навчання студента. Кожна кафедра має назву та перелік викладачів, які на ній працюють. Кожен з викладачів має свій набір курсів для викладання. Загалом кафедра пропонує набір курсів (потрібен клас для відповідного опису), які студент має завершити протягом свого навчання. Кожен курс мінімально має свій код, назву, кількість навчальних кредитів (від 1 до 6), прикріплених до нього викладачів (достатньо ПІБ), час початку та час завершення.

Збережіть класи Person та Student в окремому файлі (person.py), а код, що використовує ці класи, в чотирьох окремих файлах:

1. Тут використовуйте синтаксис «from module\_name import ClassName», щоб запустити Вашу програму.
2. Тут використовуйте синтаксис «import module\_name».
3. Використовуйте синтаксис «import module\_name as different\_local\_module\_name».
4. Використовуйте синтаксис «import \*».
5. *0,7 бала* Напишіть програму, яка створюватиме стек та дозволятиме виконувати операції з ним. На її базі реалізуйте можливість перевірки синтаксису програм мовою С: встановлення правильності послідовності фігурних дужок ({…}). Програма повинна отримувати текст програми мовою С, при знаходженні символу “{“ додавати його в стек, а при появі “}” – виконувати зняття елементу зі стеку. Перевірка стеку на пустоту встановить, чи правильно вжиті фігурні дужки в програмі.
6. *0,7 бала* Напишіть клас, який описуватиме створення черги та дозволятиме виконувати операції з нею. Створіть чергу з чисел та продемонструйте роботу даних операцій.