**АРИТМЕТИКА И ОПЕРАТОРИ**

16. Написати програм који са стандардног улаза чита температуру у степенима Целзијуса (реалан број), а на стандардном излазу исписује исту у степенима Фаренхајта (реалан број), заокружену на две децимале. Формула за претварање је *F=1.8C+32*.

23. Помножити број са 4 не служећи се оператором \* (ни \*=).

**ГРАНАЊА**

4. Написати програм који исписује највећи од три унета броја.

21. За унете коефицијенте *a*, *b* и *c* пронаћи реална решења квадратне једначине *ax^2+bx+c=0*. Ако не постоје реална решења, исписати поруку.

22. За унети вишецифрени број, проверити да ли има више од 3 цифре (исписати ако има), а ако нема више од 3 цифре, исписати да ли је број цифара паран или непаран.

24. За унети број, ако је број између 0 и 9, исписати да ли је дељив са три; ако је између 10 и 99, исписати да ли је дељив са 5, а ако је већи од 99, исписати остатак при дељењу са 15.

25. За унети редни број дана у недељи исписати који је то дан (1 - 7 = понедељак - недеља).

26. За унети месец и годину исписати колико тај месец у тој години има дана.

**ПЕТЉЕ**

1. Написати програм који исписује све бројеве од 1 до унетог броја.

2. Написати програм који исписује збир цифара унетог броја.

15. Написати програм који са стандардног улаза чита стринг и на стандардни излаз исписује *ASCII* кодове слова унетог стринга раздвојене зарезима.

27. Исписати минимум *n* унетих бројева. Уноси се *n* - број бројева, а затим *n* целих бројева.

28. Исписати аритметичку средину унетих бројева. Уноси се *n* - број бројева, а затим *n* реалних бројева.

29. Исписати збир цифара унетог целог броја.

**ГРАНАЊА И ПЕТЉЕ**

5. Написати програм који учитава бројеве све док њихов збир не пређе 100, а онда исписује број унетих бројева и њихов збир.

9. Написати функцију која проверава да ли је унети број Армстронгов. Број је Армстронгов ако је једнак збиру кубова својих цифара. Исписати све Армстронгове бројеве од 1 до 1000.

17. Написати програм који исписује све просте бројеве из унетог интервала са стандардног улаза. Бројеви треба да буду раздвојени размацима.

20. Написати програм који учитава један по један број са стандардног улаза све док се не унесе 0, а онда исписује збир свих парних и производ свих непарних унетих бројева.

**СТРИНГОВИ И ПЕТЉЕ**

6. Написати функцију која унету реч исписује на следећи начин:

Улаз: ПРИМЕР

Излаз:

П

ПР

ПРИ

ПРИМ

ПРИМЕ

ПРИМЕР

7. Написати функцију која унету реч исписује на следећи начин:

Улаз: ПРИМЕР

Излаз:

ПРИМЕР

РИМЕР

ИМЕР

МЕР

ЕР

Р

8. Написати функцију која унету реч исписује на следећи начин:

Улаз: ПРИМЕР

Излаз:

Р

РЕ

РЕМ

РЕМИ

РЕМИР

РЕМИРП

**СТРИНГОВИ, ПЕТЉЕ И ФУНКЦИЈЕ**

3. Написати функцију која рачуна *n*-ти члан Фибоначијевог низа итеративно.

10. Написати функцију која за унети текст избацује све размаке, претвара га у велика слова и дели у групе од по четири слова и враћа тако добијени стринг. Затим додати други аргумент функцији који омогућује групе произвољне величине (уместо 4 слова).

Улаз: Ово је тест пример за задатак

Излаз: ОВОЈ ЕТЕС ТПРИ МЕРЗ АЗАД АТАК

14. Написати програм који чита стринг са стандардног улаза и у излазну датотеку исписује само речи дуже од 5 слова.

**ДАТОТЕКЕ, ПЕТЉЕ, ГРАНАЊА**

11. Написати програм који из улазне датотеке чита бројеве, а у излазну датотеку исписује само просте бројеве, у засебним линијама.

**ДАТОТЕКЕ, СТРИНГОВИ**

12. Написати функцију која проверава да ли се у фајлу (на датој путањи) налази дата реч.

18. У датотеци је у свакој линији дат по један низ целих бројева, непознате дужине, раздвојених размацима. Написати програм који у излазну датотеку исписује у засебним линијама аритметичку средину сваког низа бројева заокружену на три децимале.

**ДАТОТЕКЕ, СТРИНГОВИ, СТРУКТУРЕ**

13. Написати програм који чита фајл и формира и исписује на стандардни излаз стринг који се састоји само од његових различитих слова (по редоследу по коме су прочитана).

19. Написати програм који за задати низ слова са стандардног улаза и задату улазну датотеку исписује на стандардни излаз речи из датотеке које се могу саставити од унетих слова.

**КЛАСЕ, ДЕКОРАТОРИ, ГЕНЕРАТОРИ, ИТЕРАТОРИ, УГРАЂЕНЕ ФУНКЦИЈЕ**

1. Написати генератор простих бројева из неког интервала. Генератор треба да прима почетак и крај као опционе аргументе и да има следеће понашање:
   1. Ако није прослеђен ниједан аргумент, генератор генерише све просте бројеве од 2 до бесконачности
   2. Ако је прослеђен један аргумент, генератор генерише све просте бројеве од 2 до тог броја
   3. Ако су прослеђена два аргумента, генератор генерише све просте бројеве од првог до другог. Ако прослеђени аргументи нису цели или реални бројеви, бацити TypeError изузетак са поруком о грешци. Ако су реални бројеви, заокружити их на први ближи цео број и тако их користити.
2. Написати класу SortedDict која има следећа својства:
   1. Наслеђује object (тј. не наслеђује ниједну колекцију)
   2. У својој имплементацији не сме да користи dict, ни било коју сличну “мапу”
   3. Подржава индексирање на следећи начин: може да се ради додела и читање са одређеног кључа; ако кључ не постоји у колекцији а покушано је читање, баца KeyError; ако је кључ у колекцији, придружује му нову вредност. Сви кључеви морају бити цели или реални бројеви - у случају да је покушано нешто друго, баца TypeError.
   4. Подржава итерацију тако да се враћају парови (кључ, вредност) у сортираном поретку по кључу; за ову потребу направити интерну класу SortedDictIterator
   5. Има методу keys(reverse=False), која враћа н-торку кључева у сортираном поретку, а ако јој се проследи True као аргумент, враћа је сортирану у обрнутом поретку.
3. Направити декоратор log\_exc за обраду изузетака. Декоратор треба да хвата изузетак из декорисане методе, додаје га у датотеку error\_log.txt и затим баца исти изузетак у спољашњи контекст.

**30.10.2018. (пета недеља)**

1. Дата је датотека са корисничким именима и лозинкама, таква да је у сваком реду један запис. Корисничка имена и лозинке су раздвојене табом, корисничко име не садржи специјалне карактере. Лозинке су записане као md5 хешеви стрингова (лозинки) кодираних у UTF-8 (подразумевано кодовање стрингова), записаних као hex digest. Датотека се зове passwords.txt.

а) Направити генератор свих могућих корисничких лозинки од задате дужине до задате дужине, састављених од прослеђене азбуке (азбука се задаје као стринг). За лакше генерисање комбинација дозвољено је користити itertools.product (проучити).

б) Направити rainbow таблицу која за сваки хеш памти листу могућих лозинки. Нека обухвата све лозинке над азбуком великих и малих слова енглеског алфабета и цифара које су дужине од 1 до 3 (укључујући оба броја). Користити hashlib.md5 објекте (проучити).

в) Искористити rainbow таблицу и за свако корисничко име из датотеке покушати декрипцију лозинке. Ако је лозинка откључана, исписати комбинацију корисничког имена и откључане лозинке у датотеци unlocked\_passwords.txt, раздвојене табом, за свако корисничко име у посебном реду. Ако се хеш слаже са више лозинки, исписати сваку (раздвојене табовима у истом реду)

г) Пребројати колико је откључаних лозинки и исписати статус на стандардни излаз:

- укупан број лозинки у rainbow таблици

- укупан број откључаних лозинки из датотеке

- укупан број неоткључаних лозинки из датотеке

Датотека passwords.txt је приложена уз материјале.

2. Направити структуру података (класу) вектор. Као аргументе у конструктору може да прими итерабилни објекат и тип података. Сви елементи у прослеђеном објекту морају бити конвертибилни у прослеђени тип. Подржани типови су int и float. Ако се деси било каква грешка при валидацији, баца се одговарајући изузетак.

Класа треба да имплементира следећа понашања:

а) сабирање (лево и десно асоцијативно - \_\_radd\_\_!) – сабирање може да се изврши или са вектором који је исте величине (када се елементи на одговарајућим индексима сабирају) или са скаларом (када се исти број додаје сваком елементу)

б) множење (лево и десно асоцијативно) – такође може да се изврши или са вектором исте величине (када се елементи на одговарајућим индексима множе) или са скаларом (када се цео вектор множи тим бројем)

в) „матрично” множење (\_\_matmul\_\_) – могуће га је извести само са вектором исте величине, а резултат је скалар који се добије скаларни производ вектора (тј. збир производа по компонентама)

Направити две инстанце вектора (нпр. [1, 3, 5], [2, 4, 6]) и демонстрирати свих 5 операција у главном програму, користећи операторе +, \* и @.