Универзитет у Београду

Електротехнички факултет

Одабрана поглавља нумеричке анализе

A picture containing text, watch

Description automatically generated

Пројектни задатак 1

Најбоље рационалне апроксимације реалних бројева

Студент:

Александра Богићевић 0390/17

Београд, школска година 2021/2022

**Задатак**:

Нека је дат позитиван реалан број *α* са коначним децималским записом и нека су дати природни бројеви *n* и *m*, тако да *n* < *m*. Формирати низ разломака *p*/*q* таквих да за именилац *q* важи *n* ≤ *q* ≤ *m* (тј. *q = n, n + 1, . . . ,m*) и при том имениоцу *q* придружујемо бројилац *p* који одређујемо заокруживањем на најближи природан број производа *α · q*. Представити сваки разломак *p/q* у облику верижног разломака. У низу разломака *p/q* издвојити:

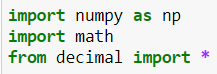
• најбоље рационалне апроксимације I врсте,

• најбоље рационалне апроксимације II врсте,

• сортирати разломке *p/q* по услову минималности апсолутне грешке | *α* − *p/q*|.

**Напомена**:

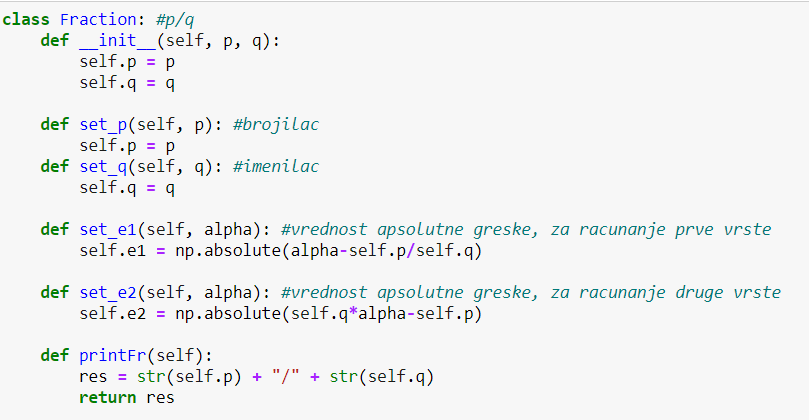
За рачунање решења коришћен је програмски језик *Python* и његове библиотеке *numpy,math* и *decimal*. Радно окружење је *Jupуter notebook*. На дну документа се налази прилог са целокупним кодом.

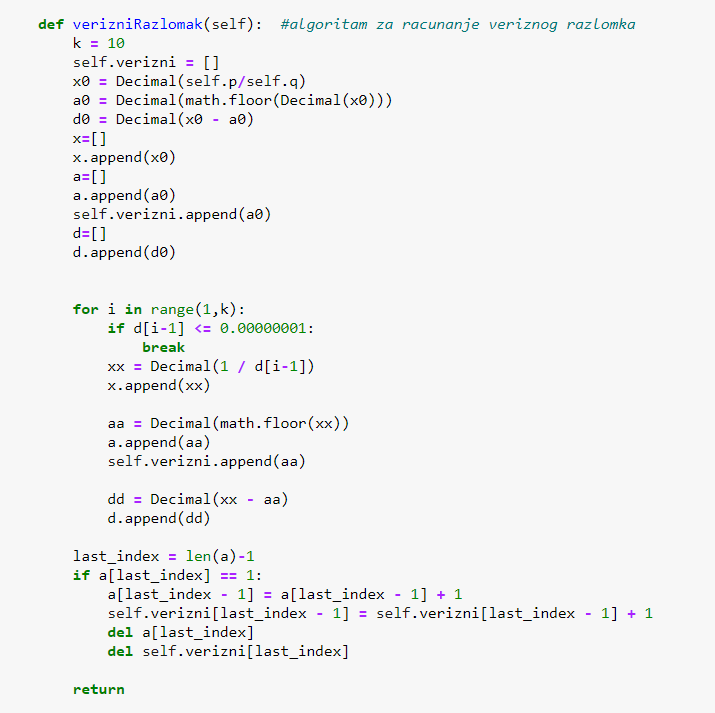


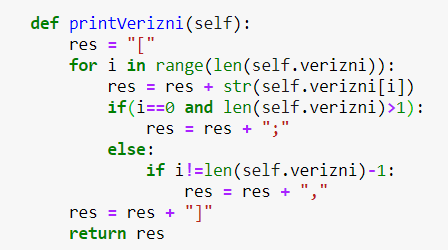
**Решење:**

За решавање овог проблема, пре свега потребно је формирати класу којом се представља разломак и све њене методе.

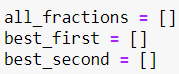
Конструктору се прослеђују вредности имениоца и бројиоца. Осим конструктора, класа садржи методе које израчунавају вредност апсолутне грешке, методу која представља разломак у верижном облику, и методе које олакшавају испис.







Такође је потребно увести и низове за смештање свих разломака, најбоље рационалне апроксимације прве врсте и најбоље рационалне апроксимације друге врсте.



На почетку програма тражимо унос података (*α, m, n*).

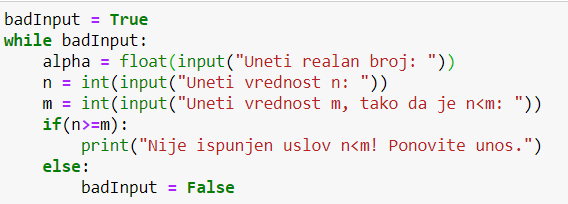
Даље у раду ће се разматрати пример универзалне параболичке константе (***universal parabolic constant***) која износи **2.2955871**.

Наши улазни параметри су:

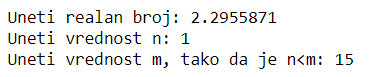
**α = 2.2955871**

**n = 1**

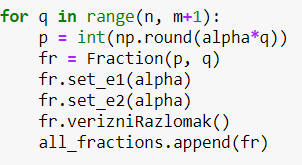
**m = 15**



Излаз програма:

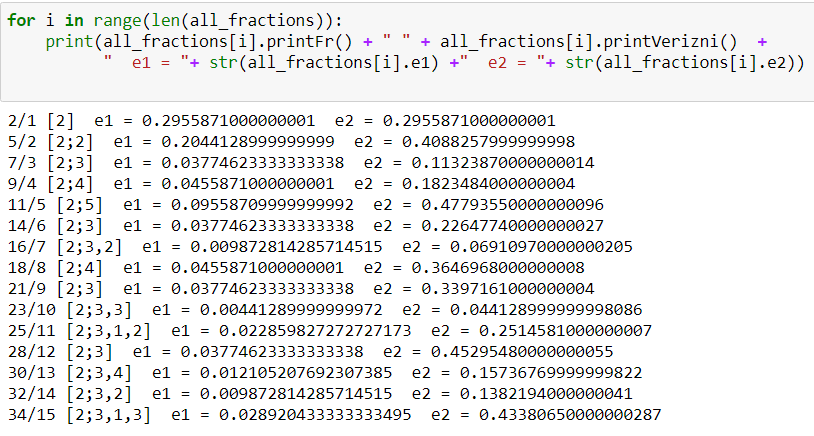


Када имамо унете вредности можемо да формирамо низ разломака *p*/*q* који је тражен у задатку.

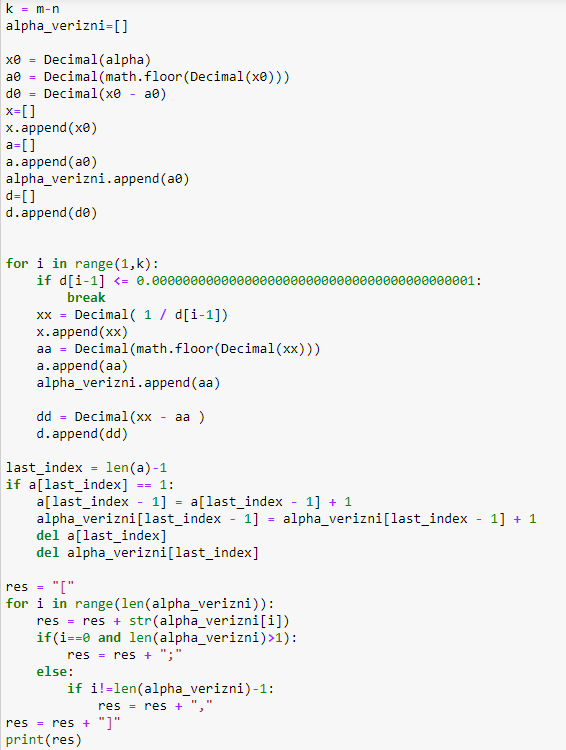


При формирању низа разломака може се приметити да се одмах и позивају методе које рачунају апсолутну грешку, као и метода која разломак представља у верижном облику.

У наставку следи изглед формираног низа.



Да бисмо испратили ток решења, можемо представити и број α у верижном облику.



Излаз програма:



Формиравши низ свих разломака, можемо из њега издвојити најбоље апроксимације прве и друге врсте.

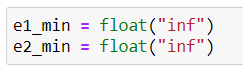
Рационални број *p/q* је **најбоља апроксимација прве врсте** броја *α* ако важи

за све разломке такве да је .

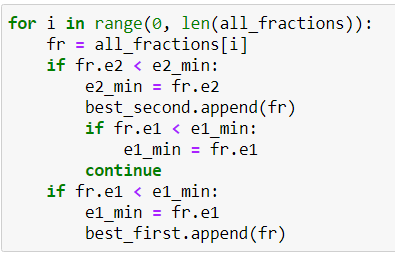
Рационални број *p/q* је **најбоља апроксимација друге врсте** броја *α* ако важи

за све разломке такве да је .

Уводимо променљиве у којима ћемо памтити досадашњи минимум грешке. Променљиве иницијализујемо на највећу могућу вредност.



На крају покрећемо алгоритам за разврставање разломака:

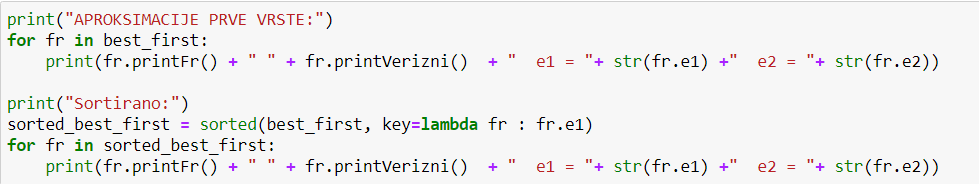


Пошто су најбоље рационалне апроксимације друге врсте уједно и најбоље апроксимације прве врсте, прво се врши провера да ли разломак припада другој врсти, и ако припада њој се приписује. Потом се врши провера да ли разломак припада првој врсти.

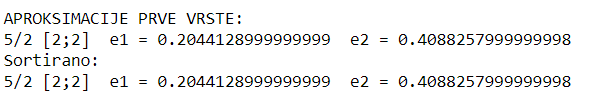
На крају овог алгоритма имамо формирана сва 3 низа.

Да бисмо испратили извршавање програма, исписаћемо садржај два новонастала низа.

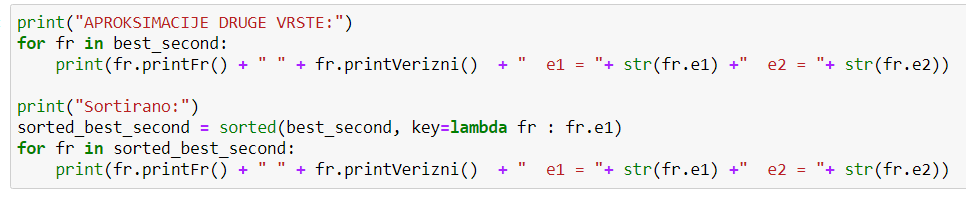
Апроксимације прве врсте:



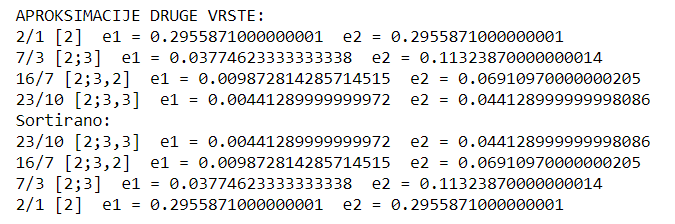
Излаз програма:



Апроксимације друге врсте:



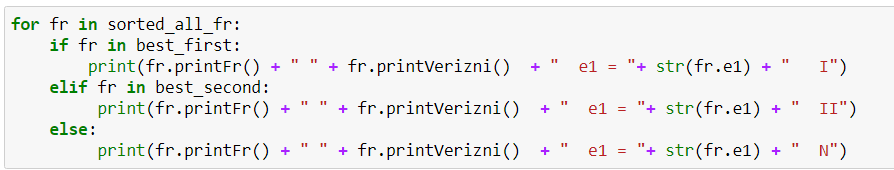
Излаз програма:



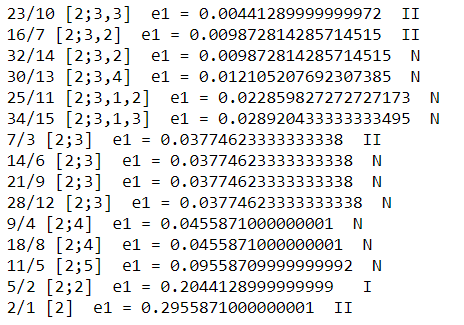
Како бисмо одговорили на задатак потребно је сортирати све разломке по услову минималности апсолутне грешке | *α* − *p/q*|.



Сада можемо да испишемо коначне резултате. При испису, последња колона означава којој врсти апроксимације разломак припада, при чему *I* означава најбоље апроксимације прве врсте, *II* означава најбоље апроксимације друге врсте, а *N* означава апроксимације које не припадају ни првој ни другој врсти.



Излаз програма:



**Прилог**:

У наставку следи приказ кода у целости.

#!/usr/bin/env python

# coding: utf-8

# In[1]:

**import** numpy **as** np

**import** math

**from** decimal **import** **\***

# Uvodimo klasu kojom predstavljamo razlomak (razlomak je oblika p/q):

# In[2]:

**class** **Fraction:** #p/q

**def** \_\_init\_\_**(**self**,** p**,** q**):**

self**.**p **=** p

self**.**q **=** q

**def** set\_p**(**self**,** p**):** #brojilac

self**.**p **=** p

**def** set\_q**(**self**,** q**):** #imenilac

self**.**q **=** q

**def** set\_e1**(**self**,** alpha**):** #vrednost apsolutne greske, za racunanje prve vrste

self**.**e1 **=** np**.**absolute**(**alpha**-**self**.**p**/**self**.**q**)**

**def** set\_e2**(**self**,** alpha**):** #vrednost apsolutne greske, za racunanje druge vrste

self**.**e2 **=** np**.**absolute**(**self**.**q**\***alpha**-**self**.**p**)**

**def** printFr**(**self**):**

res **=** str**(**self**.**p**)** **+** "/" **+** str**(**self**.**q**)**

**return** res

**def** verizniRazlomak**(**self**):** #algoritam za racunanje veriznog razlomka

k **=** 10

self**.**verizni **=** **[]**

x0 **=** Decimal**(**self**.**p**/**self**.**q**)**

a0 **=** Decimal**(**math**.**floor**(**Decimal**(**x0**)))**

d0 **=** Decimal**(**x0 **-** a0**)**

x**=[]**

x**.**append**(**x0**)**

a**=[]**

a**.**append**(**a0**)**

self**.**verizni**.**append**(**a0**)**

d**=[]**

d**.**append**(**d0**)**

**for** i **in** range**(**1**,**k**):**

**if** d**[**i**-**1**]** **<=** 0.00000001**:**

**break**

xx **=** Decimal**(**1 **/** d**[**i**-**1**])**

x**.**append**(**xx**)**

aa **=** Decimal**(**math**.**floor**(**xx**))**

a**.**append**(**aa**)**

self**.**verizni**.**append**(**aa**)**

dd **=** Decimal**(**xx **-** aa**)**

d**.**append**(**dd**)**

last\_index **=** len**(**a**)-**1

**if** a**[**last\_index**]** **==** 1**:**

a**[**last\_index **-** 1**]** **=** a**[**last\_index **-** 1**]** **+** 1

self**.**verizni**[**last\_index **-** 1**]** **=** self**.**verizni**[**last\_index **-** 1**]** **+** 1

**del** a**[**last\_index**]**

**del** self**.**verizni**[**last\_index**]**

**return**

**def** printVerizni**(**self**):**

res **=** "["

**for** i **in** range**(**len**(**self**.**verizni**)):**

res **=** res **+** str**(**self**.**verizni**[**i**])**

**if(**i**==**0 **and** len**(**self**.**verizni**)>**1**):**

res **=** res **+** ";"

**else:**

**if** i**!=**len**(**self**.**verizni**)-**1**:**

res **=** res **+** ","

res **=** res **+** "]"

**return** res

# Uvodimo nizove gde cemo smestati:

# all\_fractions - sve razlomke p/q

# best\_first - najbolje racionalne aproksimacije prve vrste

# best\_second - najbolje racionalne aproksimacije druge vrste

# In[3]:

all\_fractions **=** **[]**

best\_first **=** **[]**

best\_second **=** **[]**

# Uvodimo promenljivu u kojoj cemo cuvati minimalnu gresku i koji razlomak daje tu gresku

# In[4]:

e1\_min **=** float**(**"inf"**)**

e2\_min **=** float**(**"inf"**)**

# Unosimo vrednosti alfa, n i m.

# alfa=2.2955871

# n = 1

# m = 15

# In[5]:

badInput **=** **True**

**while** badInput**:**

alpha **=** float**(**input**(**"Uneti realan broj: "**))**

n **=** int**(**input**(**"Uneti vrednost n: "**))**

m **=** int**(**input**(**"Uneti vrednost m, tako da je n<m: "**))**

**if(**n**>=**m**):**

**print(**"Nije ispunjen uslov n<m! Ponovite unos."**)**

**else:**

badInput **=** **False**

# Algoritam za ispis razlomaka p/q:

# Formiramo niz razlomaka (all\_fractions) oblika p/q, tako da je:

# q = (n, n+1, n+2,..,m)

# p = alpha \* q, zaokruzeno na najblizi prirodan broj

# In[6]:

**for** q **in** range**(**n**,** m**+**1**):**

p **=** int**(**np**.**round**(**alpha**\***q**))**

fr **=** Fraction**(**p**,** q**)**

fr**.**set\_e1**(**alpha**)**

fr**.**set\_e2**(**alpha**)**

fr**.**verizniRazlomak**()**

all\_fractions**.**append**(**fr**)**

# Ispis razlomaka i greske:

# In[7]:

**for** i **in** range**(**len**(**all\_fractions**)):**

**print(**all\_fractions**[**i**].**printFr**()** **+** " " **+** all\_fractions**[**i**].**printVerizni**()** **+**

" e1 = "**+** str**(**all\_fractions**[**i**].**e1**)** **+**" e2 = "**+** str**(**all\_fractions**[**i**].**e2**))**

# Sortiranje i ispis liste po uslovu minimalnosti apsolutne greske |alpha-p/q|:

# In[8]:

sorted\_all\_fr **=** sorted**(**all\_fractions**,** key**=lambda** fr **:** fr**.**e1**)**

**print(**"Sortirana lista razlomaka po minimalnosti apsolutne greske e=|alpha-p/q| :"**)**

**for** i **in** range**(**len**(**sorted\_all\_fr**)):**

**print(**sorted\_all\_fr**[**i**].**printFr**()+** " " **+** sorted\_all\_fr**[**i**].**printVerizni**()** **+**" e1 = "**+** str**(**sorted\_all\_fr**[**i**].**e1**))**

# Sortiranje i ispis liste po uslovu minimalnosti apsolutne greske |alpha\*q-p|:

# In[9]:

sorted\_all\_fr2 **=** sorted**(**all\_fractions**,** key**=lambda** fr **:** fr**.**e2**)**

**print(**"Sortirana lista razlomaka po minimalnosti apsolutne greske e=|alpha\*q-p| :"**)**

**for** i **in** range**(**len**(**sorted\_all\_fr**)):**

**print(**sorted\_all\_fr2**[**i**].**printFr**()+** " " **+** sorted\_all\_fr2**[**i**].**printVerizni**()** **+**" e2 = "**+** str**(**sorted\_all\_fr2**[**i**].**e2**))**

# Trazimo verizne razlomke za datu konstantu alpha:

# In[10]:

k **=** m**-**n

alpha\_verizni**=[]**

x0 **=** Decimal**(**alpha**)**

a0 **=** Decimal**(**math**.**floor**(**Decimal**(**x0**)))**

d0 **=** Decimal**(**x0 **-** a0**)**

x**=[]**

x**.**append**(**x0**)**

a**=[]**

a**.**append**(**a0**)**

alpha\_verizni**.**append**(**a0**)**

d**=[]**

d**.**append**(**d0**)**

**for** i **in** range**(**1**,**k**):**

**if** d**[**i**-**1**]** **<=** 0.000000000000000000000000000000000000000001**:**

**break**

xx **=** Decimal**(** 1 **/** d**[**i**-**1**])**

x**.**append**(**xx**)**

aa **=** Decimal**(**math**.**floor**(**Decimal**(**xx**)))**

a**.**append**(**aa**)**

alpha\_verizni**.**append**(**aa**)**

dd **=** Decimal**(**xx **-** aa **)**

d**.**append**(**dd**)**

last\_index **=** len**(**a**)-**1

**if** a**[**last\_index**]** **==** 1**:**

a**[**last\_index **-** 1**]** **=** a**[**last\_index **-** 1**]** **+** 1

alpha\_verizni**[**last\_index **-** 1**]** **=** alpha\_verizni**[**last\_index **-** 1**]** **+** 1

**del** a**[**last\_index**]**

**del** alpha\_verizni**[**last\_index**]**

res **=** "["

**for** i **in** range**(**len**(**alpha\_verizni**)):**

res **=** res **+** str**(**alpha\_verizni**[**i**])**

**if(**i**==**0 **and** len**(**alpha\_verizni**)>**1**):**

res **=** res **+** ";"

**else:**

**if** i**!=**len**(**alpha\_verizni**)-**1**:**

res **=** res **+** ","

res **=** res **+** "]"

**print(**res**)**

# Trazimo aproksimacije prve i druge vrste:

#

# In[11]:

**for** i **in** range**(**0**,** len**(**all\_fractions**)):**

fr **=** all\_fractions**[**i**]**

**if** fr**.**e2 **<** e2\_min**:**

e2\_min **=** fr**.**e2

best\_second**.**append**(**fr**)**

**if** fr**.**e1 **<** e1\_min**:**

e1\_min **=** fr**.**e1

**continue**

**if** fr**.**e1 **<** e1\_min**:**

e1\_min **=** fr**.**e1

best\_first**.**append**(**fr**)**

# Ispis prve vrste:

# In[12]:

**print(**"APROKSIMACIJE PRVE VRSTE:"**)**

**for** fr **in** best\_first**:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+**" e2 = "**+** str**(**fr**.**e2**))**

**print(**"Sortirano:"**)**

sorted\_best\_first **=** sorted**(**best\_first**,** key**=lambda** fr **:** fr**.**e1**)**

**for** fr **in** sorted\_best\_first**:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+**" e2 = "**+** str**(**fr**.**e2**))**

# Ispis druge vrste:

# In[13]:

**print(**"APROKSIMACIJE DRUGE VRSTE:"**)**

**for** fr **in** best\_second**:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+**" e2 = "**+** str**(**fr**.**e2**))**

**print(**"Sortirano:"**)**

sorted\_best\_second **=** sorted**(**best\_second**,** key**=lambda** fr **:** fr**.**e1**)**

**for** fr **in** sorted\_best\_second**:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+**" e2 = "**+** str**(**fr**.**e2**))**

# Konacan ispis:

# In[14]:

**for** fr **in** sorted\_all\_fr**:**

**if** fr **in** best\_first**:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+** " I"**)**

**elif** fr **in** best\_second**:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+** " II"**)**

**else:**

**print(**fr**.**printFr**()** **+** " " **+** fr**.**printVerizni**()** **+** " e1 = "**+** str**(**fr**.**e1**)** **+** " N"**)**