Diszkrét matematika
5. előadás

MÁRTON Gyöngyvér
mgyongyi@ms.sapientia.ro

Sapientia Egyetem,
Matematika-Informatika Tanszék
Marosvásárhely, Románia

2023, őszi félév

Miről volt szó az elmúlt előadáson?

- Python:
  - a `strip`, `split`, `zip` függvények,
  - szövegállományok,
  - a `Fraction` típus, a `Decimal` típus,
- racionális számok, lánctörtek, irracionális számok,
- "híresebb" irracionális számok: $\sqrt{n}$,
- algoritmusok:
  - adott kritérium szerinti kiválogatás,
  - racionális számok irreducibilis alakja,
  - racionális számok sorba rendezése - két módszer,
  - racionális számok lánctört jegyei.

Miről lesz szó?

- Python: a `random`, a `numpy` és a `matplotlib` modulok, a `complex` típus,
- "híresebb" irracionális számok tízedesjegyei: $\sqrt{2}, \pi, e, \phi$,
- valós számok, komplex számok, számrendszerek,
- algoritmusok:
  - polinom helyettesítési értéke, függvények ábrázolása,
  - a `sin, log` függvények,
  - műveletek komplex számokkal: szorzat, hatványozás, stb
  - fraktálok: Mandelbrot, Julia

A π szám

- a kör kerületének és átmérőjének hányadosa az eukleidészi geometriában:

$$\pi = \frac{K}{2 \cdot R},$$

  ahol R a kör sugara, K a kör kerülete
- más definíciók is léteznek, melyek kihagyják a kört
- irracionális és transzcendens szám (nincs olyan egész együtthatós polinom amelynek gyöke lenne)

A π értékének meghatározására több lánctört-képlet is ismert:

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{7 + \ldots}}}}$$

$$\pi = 3 + \cfrac{1}{6 + \cfrac{3^2}{6 + \cfrac{5^2}{6 + \cfrac{7^2}{6 + \ldots}}}}$$

A π szám

1. feladat

Határozzuk meg π értékét lánctörtek segítségével.

```
def my_pi(p):
    getcontext().prec = p
    temp = Decimal(0)
    for x in range(1000, 0, -1):
        a = x * x
        b = 2 * x + 1
        temp = a / (b + temp)
    return 4 / (1 + temp)

>>> my_pi(100)
    Decimal('3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 7')
```

Az e szám

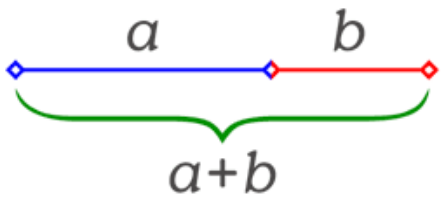
- irracionális és transzcendens szám
- többféleképpen lehet értelmezni:

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n.$$

$$e = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{2}{3 + \cfrac{3}{4 + \ldots}}}}$$

A φ szám, aranymetszés, aranyarány (Golden Ratio)

- két mennyiség, $a, b, a > b$ az aranymetszés szerint aránylik egymáshoz, ha fennáll:



$$\varphi \stackrel{def}{=} \frac{a}{b} = \frac{a+b}{a}$$

- a φ meghatározása érdekében felírhatjuk: $\frac{a+b}{a} = 1 + \frac{1}{\frac{a}{b}}$, azaz fennáll:

$$\varphi = 1 + \frac{1}{\varphi} \Leftrightarrow \varphi^2 = \varphi + 1$$

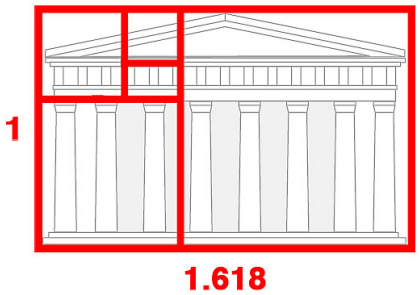
- megoldva a fenti egyenletet kapjuk, hogy

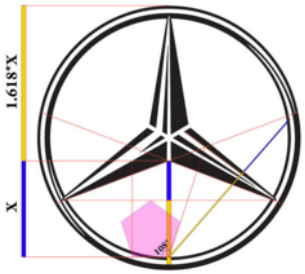
$$\varphi = \frac{1+\sqrt{5}}{2} = 1.61803\ldots \text{ és } \hat{\varphi} = \frac{1-\sqrt{5}}{2} = -0.61803\ldots$$

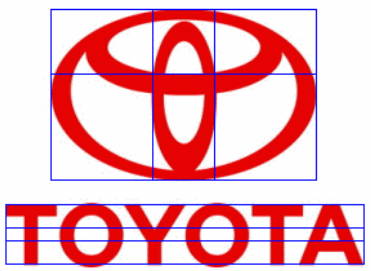
- a φ irracionális szám

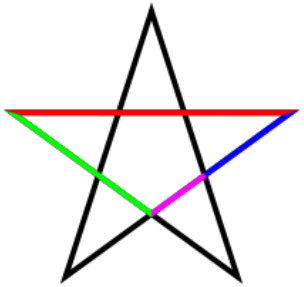
A φ szám, aranymetszés, aranyarány (Golden Ratio)

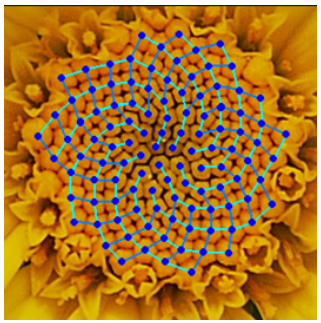
- építészet: Parthenon homlokzatának arányértékei:
- logok: Toyota, Mercedesz, stb.
- Pentagramma (szabályos ötszög): $\frac{piros}{zold} = \frac{zold}{kek} = \frac{kek}{lila} = \varphi$
- természet: napraforgó spiráljai











A φ szám, aranymetszés, aranyarány (Golden Ratio)

Kiindulva a $\varphi = 1 + \frac{1}{\varphi}$ összefüggésből, a φ értékét felírhatjuk a következőképpen:

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ddots}}}}$$

A φ szám, aranymetszés, aranyarány (Golden Ratio)

- két egymás utáni Fibonacci szám arányaként is felírhatjuk a φ értékét
- egy Fibonacci szám a Fibonacci számsorozat egy eleme
- a Fibonacci számsorozat: $0, 1, 1, 2, 3, 5, 8, 13...$, kiindulva a 0, 1 kezdőértékekből, a következő elemet az előző két elem összegéből kapjuk

Az `n` és `m` közötti, két egymásutáni Fibonacci számból képzett arányok meghatározása:

```
def fib_phi(n, m):
    f1 = 0
    f2 = 1
    for i in range(2, m):
        f = f1 + f2
        if i >= n:
            print("%4d%5d%5d%10.6f" % (i, f, f2, f/f2))
        f1 = f2
        f2 = f

>>> fib_phi(10, 14)
   10    55    34   1.617647
   11    89    55   1.618182
   12   144    89   1.617978
   13   233   144   1.618056
```

Valós számok

- a racionális és irracionális számok halmaza,
- halmazjelölés: $\mathbb{R} = \mathbb{Q} \cup \mathbb{Q}^*$,
- egy szám egyszerre nem lehet racionális és irracionális is,
- a valós számokhoz hozzárendelhető, egy mindkét irányban végtelen egyenes egy-egy pontja,
- kommutatívitás, asszociatívítás, disztributívítás,
- az egész számokkal ellentétben a valós számok halmaza nem megszámlálható,
- a számítástechnikában nem valós mennyiségekkel dolgozunk, ezek egy közelítő értéke lesz eltárolva: lebegőpontos ábrázolás

A $\sin(z)$ értéke

2. feladat

Írjunk egy Python függvényt, amely meghatározza $\sin(z)$ *p számú tizedes jegyét a következő lánctörtet alkalmazva:*

$$\sin(z) = \cfrac{z}{1 + \cfrac{z^2}{2 \cdot 3 - z^2 + \cfrac{2 \cdot 3 \cdot z^2}{4 \cdot 5 - z^2 + \cfrac{4 \cdot 5 \cdot z^2}{6 \cdot 7 - z^2 + \ldots}}}}$$

```
def my_sin_(z, p = 50):
    getcontext().prec = p
    temp = Decimal(0)
    for x in range(1000, 0, -2):
        a = (x + 2) * (x + 3) - z*z
        b = x * (x + 1) * z * z
        temp = b / (a + temp)
    temp = z*z / (2*3 - z*z + temp)
    return z / (1 + temp)
```

A sin(z) értéke

```
>>> my_sin_(60)
    Decimal('-0.304810621102216705625762041861313457514405658222l8')

>>> import math
>>> math.sin(60)
    -0.304810621102216 7

>>> math.sqrt(3)/2
    0.8660254037844386
```

Mi a probléma? Miért nem a helyes eredményt kapjuk? A paraméterként megadott értéket át kell alakítani radiánba:

```
>>> math.sin(60 * math.pi/180)
    0.8660254037844386

>>> math.sin(math.radians(60))
    0.8660254037844386

>>> my_sin_(Decimal(math.radians(60)))
    Decimal('0.866025403784438589345509030791826171935265533514 20')
```

A sin(z) értéke

az átalakított kódsor:

```
from math import radians, pi
def my_sin(z):
    getcontext().prec = 50
    #z = Decimal(z * pi/180)
    z = Decimal(radians(z))
    temp = Decimal(0)
    for x in range(1000, 0, -2):
        a = (x + 2) * (x + 3) - z * z
        b = x * (x + 1) * z * z
        temp = b / (a + temp)
    temp = z * z / (2 * 3 - z * z + temp)
    return z / (1 + temp)

>>> my_sin(60)
Decimal('0.86602540378443858934550903079182617193526553351420')
```

Az $\ln(z)$ értéke

A $\ln(z)$ meghatározásához a következő lánctörtet használhatjuk:

$$\ln(1+z) = \cfrac{z}{1+\cfrac{1^2 \cdot z}{2+\cfrac{1^2 \cdot z}{3+\cfrac{2^2 \cdot z}{4+\cfrac{2^2 \cdot z}{5+\cfrac{3^2 \cdot z}{6+\ldots}}}}}}$$

Használhatóak a következő összefüggések is:

$$\ln(z) = (z-1) - \frac{(z-1)^2}{2} + \frac{(z-1)^3}{3} - \frac{(z-1)^4}{4} \cdots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}(z-1)^n$$

$$\ln(z) = \lim_{n\to\infty} n \cdot (z^{1/n} - 1)$$

Polinom helyettesítési értéke

3. feladat

Írjunk egy Python függvényt, amely meghatározza egy adott polinom helyettesítési értékét, egy megadott értékre.

- két algoritmust adunk meg, ahol a `horner` hatékonyabb lesz, mint a `poly`,
- mindkét függvénynek két bemeneti értéket kell megadni, az első a *megadott* érték, a második egy lista, amelyben a polinom együtthatóit kell megadni, ahol a lista első eleme a polinom legnagyobb fokszámú eleme kell legyen. Ha $A(c) = a_n \cdot c^n + \cdots + a_1 \cdot c + a_0$, akkor a lista $= [a_n, a_{n-1}, \ldots, a_1, a_0]$.

```
def poly(c, A):
    res, p = 0, 1
    for a in A[::-1]:
        res += a * p
        p *= c
    return res
```

$$
\begin{array}{lcl}
res, p & = & 0, 1 \\
res, p & = & a_0 \cdot 1, c \\
res, p & = & a_1 \cdot c + a_0 \cdot 1, c^2 \\
res, p & = & a_2 \cdot c^2 + a_1 \cdot c + a_0 \cdot 1, c^3 \\
\ldots & &
\end{array}
$$

```
def horner(c, A):
    res = 0
    for a in A:
        res = res * c + a
    return res
```

$$
\begin{array}{lcl}
res & = & 0 \\
res & = & 0 \cdot c + a_n = a_n \\
res & = & a_n \cdot c + a_{n-1} \\
res & = & (a_n \cdot c + a_{n-1}) \cdot c + a_{n-2} \\
 & = & = a_n \cdot c^2 + a_{n-1} \cdot c + a_{n-2} \\
\ldots & &
\end{array}
$$

Polinom helyettesítési értéke

A következő `yValue` függvény több helyettesítési értékre is meghatározza a `L` listában megadott polinom értékeit, ahol a helyettesítési értéke az `xs` kezdőértéktől indul, `step` lesz a lépésköz és `xF` a végső helyettesítési érték:

```
def yValue(L, xS, xF, step)
    x = xS
    while x <= xF:
        print(x, horner(x, L))
        x += step

>>> L = [1, -1, -1]
>>> xS = -2, xF = 3, step = 0.5
>>> yValue()
    -2 5
    -1.5 2.75
    ...
```

A numpy és matplotlib csomagok

A függvények/polinomok ábrázolásához szükség lesz a numpy és matplotlib csomagokra, amelyek telepítésékor, illetve egyéb Python csomagok telepítésekor használjuk a pip telepítőt. A Python újabb verziói alapból tartalmazzák a pip-et.

- Windows alatti egyszerűbb használatához, be kell legyen állítva a PATH környezetváltozó (lásd részletesebben az 1. előadásban)
- adjuk ki a következő parancsokat:
  - `python -m pip install numpy`
  - `python -m pip install matplotlib`

Ha nem ismeri fel a pip-et, akkor a csomagok telepítése előtt azt is telepíteni kell:

- töltsük le egy mappába a *get-pip.py* állományt: *https://bootstrap.pypa.io/3.2/get-pip.py*
- nyissunk meg egy Command prompt ablakot, válasszuk ki azt a mappát amibe a get-pip állományt tettük, pl: `cd C:\Downloads`
- adjuk ki a következő parancsokat:
  - `python get-pip.py`
  - `python -m install --upgrade pip`

Polynomok/függvények ábrázolása

4. feladat

Írjunk egy Python függvényt, amely grafikusan ábrázolja a megadott függvényt/polinomot.

```
import numpy as np
import matplotlib.pyplot as plt

def main():
    X = np.linspace(-200, 200, 300, endpoint=True)
    F1 = horner(X, [1, -4, 3])
    F2 = horner(X, [2, 7, 7])
    #F1 = horner(X, [4, 11, 3, 5])
    #F2 = horner(X, [10, 3, 11, 19])
    plt.plot(X, F1, label = "F1")
    plt.plot(X, F2, label = "F2")

    plt.axvline(x = 0, c = "lightgray")
    plt.axhline(y = 0, c = "lightgray")
    plt.show()

main()
```

Polynomok/függvények ábrázolása

```
def my_sinF(z):
    temp = 0
    for x in range(30, 0, -2):
        a = (x + 2) * (x + 3) - z * z
        b = x * (x + 1) * z * z
        temp = b / (a + temp)
    temp = z * z / (2 * 3 - z * z + temp)
    return z / (1 + temp)

def main2():
    X = np.arange(0, 3*np.pi, 0.1)
    #plt.plot(X, np.sin(X))
    plt.plot(X, my_sinF(X))
    plt.axvline(x = 0, c = "red")
    plt.axhline(y = 0, c = "red")
    plt.show()

main2()
```

Komplex számok

A komplex számok a valós számhalmaz egy olyan bővítése, melyben negatív számok esetén is értelmezett a gyökvonás, halmazjelölés: $\mathbb{C}$.

- három modell alapján is értelmezhető: halmazelméleti, geometriai, algebrai modell alapján
- halmazelméleti modell: $\mathbb{C} = \{(a, b) | a \in \mathbb{R}, b \in \mathbb{R}\}$, azaz a számhalmazt rendezett számpárok alkotják, ahol az elemek valós számok,
- imaginárius rész: az a komplex szám amelynek négyzete -1, jele az `i`
- a komplex számok $a + b \cdot i$ alakban írhatóak fel, ahol a valós rész, b imaginárius egység
- ha $b = 0$, akkor valós számot kapunk
- a valós számok körében megismert műveleti tulajdonságok megmaradnak
- additív semleges elem: $z = 0 + 0 \cdot i$
- multiplikatív semleges elem: $z = 1 + 0 \cdot i$
- additív inverz elem: $-z = -a - b \cdot i$,
- multiplikatív inverz elem: $1/z = a/(a^2 + b^2) - b/(a^2 + b^2) \cdot i, z \neq 0$

Műveletek komplex számokkal

$$
\begin{aligned}
a &= a_1 + a_2 \cdot i \\
abs(a) &= \sqrt{a_1^2 + a_2^2} \\
b &= b_1 + b_2 \cdot i \\
a + b &= (a_1 + b_1) + (a_2 + b_2) \cdot i \\
a - b &= (a_1 - b_1) + (a_2 - b_2) \cdot i \\
a \cdot b &= (a_1 \cdot b_1 - a_2 \cdot b_2) + (a_2 \cdot b_1 + a_1 \cdot b_2) \cdot i \\
\frac{1}{b} &= \frac{b_1}{(b_1^2 + b_2^2)} - \frac{b_2}{(b_1^2 + b_2^2)} \cdot i \\
\frac{a}{b} &= a \cdot \frac{1}{b}
\end{aligned}
$$

```
>>> a = complex(2.5, 4.3)
>>> a.real
    2.5
>>> a.imag
    4.3
>>> abs(a)
    4.973932046178355
>>> a ** 3.2
    (-166.23299664047408-33.65354280875679j)

>>> b = complex(1.7,10.25)
>>> a + b
    (4.2+14.55j)
>>> a - b
    (0.8-5.95j)
>>> a * b
    (-39.824999999999996+32.935j)
>>> a / b
    (0.44765058706375493-0.1696795141381 63j)
```

Két komplex szám szorzata

5. feladat

Írjunk egy Python függvényt, amely meghatározza az $a = a_1 + a_2 i$ és $b = b_1 + b_2 i$ komplex számok szorzatát, ahol alkalmazzuk az
$a \cdot b = (a_1 \cdot b_1 - a_2 \cdot b_2) + (a_2 \cdot b_1 + a_1 \cdot b_2)i$ összefüggést.

```
def szorzatC(a, b):
    a1, a2 = a
    b1, b2 = b
    vResz = a1 * b1 - a2 * b2
    iResz = a2 * b1 + a1 * b2
    return (vResz, iResz)

>>> szorzatC((2.5, 5.4), (3.34, 101.5))
    (-539.75, 271.786)

>>> a = complex(2.5, 5.4)
>>> b = complex(3.34, 101.5)
>>> a * b
    (-539.75+271.786j)
```

Komplex számok hatványozása

6. feladat

Írjunk egy Python függvényt, amely meghatározza a^n-t, ahol $a = (a_1, a_2)$ az $a_1 + a_2 i$ komplex szám és n egy természetes szám. Alkalmazzuk a gyorshatványozás algoritmusát.

```
def my_powC (a, n):
    res = (1, 0)
    while True:
        if n % 2 == 1:
            res = szorzatC(res, a)
        if n == 1: break
        a = szorzatC(a, a)
        n = n // 2
    return res

>>> my_powC((2.5, 5.4), 3)
    (-203.07500000000002, -56.21400000000003)

>>> a = complex(2.5, 5.4)
>>> pow(a, 3)
    (-203.07500000000002-56.21400000000003j)
```

Mandelbrot fraktál

- Fraktálok: kiindulva egy komplex számból, egy iterációs folyamat eredményeként a képernyőre kirajzolt pontok fraktál alakzatokat hozhatnak létre
- minél nagyobb az iteráció szám, annál jobb a kirajzolt kép minősége
- a Mandelbrot halmaz iterációs képlete:

$$z_0 = c$$
$$z_{n+1} = (z_n)^2 + c,$$

  ahol a c komplex szám értékét a programozó állítja be
- a Mandelbrot halmaz azokat a c komplex számokat fogja tartalmazni, amelyekre a z_n sorozat nem tart a végtelenbe,
- a z_n sorozat nem tart a végtelenbe, ha az i-edik iteráció után z_i abszolútértéke kisebb vagy egyenlő lesz mint 2.

Mandelbrot fraktál

7. feladat

Írjunk egy Python függvényt, amely meghatározza a Mandelbrot halmaz elemeit. Egy halmazbeli elem esetén írjunk #-t a képernyőre, másképp space-t.

```
def fMan():
    L1 = [a*0.07 for a in range(-15, 16)]
    L2 = [a*0.04 for a in range(-50, 26)]
    for y in L1:
        L = ""
        for x in L2:
            c = complex(x, y)
            z = c
            for i in range(40):
                    z = z ** 2 + c
                    if abs(z) > 2:
                        L += " "
                        break
            if abs(z) <= 2: L += "#"
        print (L)
```

Julia fraktál

8. feladat

Írjunk egy Python függvényt, amely meghatározza a Julia halmaz elemeit. Egy halmazbeli elem esetén írjunk #-t a képernyőre, másképp space-t.

A Julia halmaz iterációs képlete ugyanaz, mint a Mandelbrot halmazé, azzal a különbséggel, hogy a c értéke itt konstans, legyen $c = -1 - 0.25i$.

```
def fJulia():
    L1 = [a*0.07 for a in range(-15, 16)]
    L2 = [a*0.04 for a in range(-40, 36)]
    c = complex (-1, -0.25)
    for y in L1:
        L = ""
        for x in L2:
            z = complex(x, y)
            for i in range(40):
                    z = z ** 2 + c
                    if abs(z) > 2:
                        L += " "
                        break
            if abs(z) <= 2: L += "#"
        print (L)
```

Megjegyzések

- az L1, L2 intervallumokat módosíthatjuk, a kirajzolt alakzat nagyobb lesz:

```
L1 = [a*0.05 for a in range(-20, 21)]
L2 = [a*0.02 for a in range(-80, 31)]
```

- a grafikus megjelenítéshez használjuk a `pygame` csomagot, telepítéséhez nyíssunk meg egy Command prompt ablakot, majd adjuk ki a következő parancsot: `python -m pip install pygame`
- a grafikus megjelenítés *valamikori* forrása: *http://fractalart.gallery/mandelbrot-set-in-python-pygame*
- a két fraktált grafikusan megjelenítő kódsorok a *4. labor* megoldott feladatai között találhatók