Python Formelark – Videregående skole

Organisert etter fag — LK20

1P – Praktisk matematikk

Prosentregning

```
# Finne prosent av tall
prosent = 25
tall = 800
resultat = tall * prosent / 100

# Prosentvis endring
gammelt = 500
nytt = 600
endring = (nytt-gammelt)/gammelt*100
```

Vekstfaktor

```
# Økning p%: vekstfaktor = 1+p/100
start = 1000
vekst = 1.05  # 5% økning
etter_n_aar = start * vekst**n

# Nedgang p%: vekstfaktor = 1-p/100
vekst = 0.92  # 8% nedgang
```

Lån og renter

```
def annuitet(laan, rente, terminer):
    r = rente / 100
    return laan*r/(1-(1+r)**(-
     terminer))

mnd = annuitet(2000000, 0.4, 300)
```

1T – Teoretisk matematikk

Funksjoner

```
def f(x):
    return x**2 - 4*x + 3

# Funksjonsverdi
y = f(5)

# Tabell
for x in range(-2, 6):
    print(f"f({x}) = {f(x)}")
```

Halveringsmetoden

```
def halvering(f, a, b, tol=0.0001):
    while b - a > tol:
        m = (a + b) / 2
        if f(a) * f(m) < 0:
            b = m
        else:
            a = m
    return (a + b) / 2

nullpunkt = halvering(f, 2, 3)
```

Lineær regresjon

```
x = [1, 2, 3, 4, 5]
y = [2.1, 4.0, 5.8, 8.1, 9.9]
n = len(x)
sx, sy = sum(x), sum(y)
sxy = sum(x[i]*y[i] for i in range(n)
    )
sx2 = sum(xi**2 for xi in x)

a = (n*sxy - sx*sy)/(n*sx2 - sx**2)
b = (sy - a*sx) / n
# y = ax + b
```

2P – Praktisk matematikk

Statistiske mål

```
data = [4, 7, 2, 9, 5, 8, 3, 6]

# Sentralmål
snitt = sum(data) / len(data)
sortert = sorted(data)
n = len(sortert)
if n % 2 == 0:
    median = (sortert[n//2-1] +
              sortert[n//2]) / 2
else:
    median = sortert[n//2]

# Spredningsmål
variasjon = max(data) - min(data)
```

Standardavvik

```
def standardavvik(data):
    n = len(data)
    snitt = sum(data) / n
    avvik = [(x-snitt)**2 for x in
     data]
    varians = sum(avvik) / n
    return varians ** 0.5

sd = standardavvik([4, 7, 2, 9, 5])
```

Simulering

```
import random

kast = random.randint(1, 6)
mynt = random.choice(["kr", "mynt"])

# Simuler 1000 forsøk
suksess = 0
for _ in range(1000):
    if random.randint(1, 6) == 6:
        suksess += 1
sannsynlighet = suksess / 1000
```

S1 – Samfunnsfaglig

Kombinatorikk

```
import math

# Fakultet: n!
math.factorial(5)  # 120

# Permutasjoner: P(n,r)=n!/(n-r)!
def P(n, r):
    return (math.factorial(n) //
            math.factorial(n-r))

# Kombinasjoner: C(n,r)=n!/[r!(n-r)!]
def C(n, r):
    return (math.factorial(n) //
            (math.factorial(r) *
             math.factorial(n-r)))
```

Binomialfordeling

```
def binom(n, k, p):
    """P(X=k) for X ~ Bin(n,p)"""
    C_nk = (math.factorial(n) //
            (math.factorial(k) *
             math.factorial(n-k)))
    return C_nk * p**k * (1-p)**(n-k)

# P(X = 4) for Bin(10, 0.3)
sann = binom(10, 4, 0.3)
```

R1 – Realfag

Numerisk derivasjon

```
def f(x):
    return x**3 - 2*x + 1

def derivert(f, x, h=0.0001):
    return (f(x+h) - f(x)) / h

# Sentral differanse (mer presist)
def derivert_s(f, x, h=0.0001):
    return (f(x+h) - f(x-h)) / (2*h)
```

Newtons metode

```
def newton(f, x0, tol=1e-8, maks=100)
    :
    x = x0
    for i in range(maks):
        fx = f(x)
        fpx = derivert(f, x)
        x_ny = x - fx / fpx
        if abs(x_ny - x) < tol:
            return x_ny
        x = x_ny
    return x

nullpunkt = newton(f, 1.5)
```

Ekstremalpunkter

```
def finn_ekstremum(f, a, b, n=1000):
    min_x, min_y = a, f(a)
    max_x, max_y = a, f(a)
    for i in range(n+1):
        x = a + i*(b-a)/n
        y = f(x)
        if y < min_y:
            min_x, min_y = x, y
        if y > max_y:
            max_x, max_y = x, y
    return (min_x,min_y), (max_x,
     max_y)
```

S2 – Samfunnsfaglig

Konfidensintervall

```
# 95% konfidensintervall
def konfidensint(data, z=1.96):
    n = len(data)
    snitt = sum(data) / n
    avvik = [(x-snitt)**2 for x in
     data]
    sd = (sum(avvik)/n) ** 0.5
    feil = z * sd / (n ** 0.5)
    return (snitt-feil, snitt+feil)
```

Hypotesetesting

```
def z_test(utv, pop, sd, n):
    """z-verdi for en-utvalgs test"""
    return (utv - pop) / (sd / n
     **0.5)

# Forkast H0 hvis |z| > 1.96 (5%)
z = z_test(52, 50, 10, 100)
forkast = abs(z) > 1.96
```

Monte Carlo

```
# Estimere pi
def estimer_pi(n):
    inne = 0
    for _ in range(n):
        x = random.random()
        y = random.random()
        if x**2 + y**2 <= 1:
            inne += 1
    return 4 * inne / n
```

Lineær regresjon med R2

```
def regresjon(x, y):
    n = len(x)
    sx, sy = sum(x), sum(y)
    sxy = sum(x[i]*y[i] for i in
     range(n))
    sx2 = sum(xi**2 for xi in x)

    a = (n*sxy - sx*sy)/(n*sx2 - sx
     **2)
    b = (sy - a*sx) / n

    # R2 (forklaringsgrad)
    y_pred = [a*xi + b for xi in x]
    ss_tot = sum((yi-sy/n)**2 for yi
     in y)
    ss_res = sum((y[i]-y_pred[i])**2
                 for i in range(n))
    r2 = 1 - ss_res/ss_tot
    return a, b, r2
```

R2 – Realfag

Rektangelmetoden

```
def rektangel(f, a, b, n=1000):
    dx = (b - a) / n
    total = 0
    for i in range(n):
        x = a + i * dx
        total += f(x) * dx
    return total
```

Trapesmetoden

```
def trapes(f, a, b, n=1000):
    dx = (b - a) / n
    total = (f(a) + f(b)) / 2
    for i in range(1, n):
        total += f(a + i * dx)
    return total * dx
```

Simpsons regel

```
def simpson(f, a, b, n=1000):
    if n % 2 == 1:
        n += 1
    dx = (b - a) / n
    total = f(a) + f(b)
    for i in range(1, n):
        x = a + i * dx
        if i % 2 == 0:
            total += 2 * f(x)
        else:
            total += 4 * f(x)
    return total * dx / 3
```

Rekursive følger

```
# Aritmetisk: a_n = a_1 + (n-1)*d
def aritmetisk(a1, d, n):
    return a1 + (n-1) * d

# Geometrisk: a_n = a_1 * k^(n-1)
def geometrisk(a1, k, n):
    return a1 * k**(n-1)

# Rekursiv definisjon
def rekursiv(a1, n, formel):
    a = a1
    for _ in range(n-1):
        a = formel(a)
    return a
```

Eulers metode

```
def euler(f, x0, y0, x_slutt, n):
    """Løs y' = f(x,y), y(x0) = y0"""
    h = (x_slutt - x0) / n
    x, y = x0, y0
    punkter = [(x, y)]

    for _ in range(n):
        y = y + h * f(x, y)
        x = x + h
        punkter.append((x, y))
    return punkter

# y' = x + y, y(0) = 1
losn = euler(lambda x,y: x+y, 0, 1,
    2, 100)
```

Areal mellom kurver

```
def areal_mellom(f, g, a, b, n=1000):
    """Areal mellom f(x) og g(x)"""
    def diff(x):
        return abs(f(x) - g(x))
    return simpson(diff, a, b, n)
```

Volum ved rotasjon

```
import math

def volum_x(f, a, b, n=1000):
    """Rotasjon om x-aksen"""
    def f_kv(x):
        return f(x)**2
    return math.pi * simpson(f_kv, a,
      b, n)

def volum_y(f, a, b, n=1000):
    """Rotasjon om y-aksen (
     skallmetoden)"""
    def integrand(x):
        return 2 * math.pi * x * f(x)
    return simpson(integrand, a, b, n
    )
```

Differensiallikninger

```
# Separabel: dy/dx = g(x)*h(y)
# Løs ved separasjon og integrasjon

# Lineaer: dy/dx + P(x)*y = Q(x)
# Løs med integrerende faktor

# Numerisk med Euler (se over)
```

Nyttige tips

- Bruk `import math` for matematiske funksjoner
- `math.pi`, `math.e`, `math.sqrt()`
- `math.sin()`, `math.cos()`, `math.log()`
- Husk: `math.log()` er ln, `math.log10()` er log