BÀI 3: KỸ THUẬT LẬP TRÌNH PYTHON

1. Mục Tiêu

- Xử lý dãy số, ma trận trên Numpy
- Hiển thị đồ thị dùng Matplotlib

2. Bài tập thực hành

Bài 1. Tính toán trên dãy số

1. Tạo một dãy số a có N phần tử (N = 10)

```
import numpy as np

print("1. Tạo một dãy số a có N phần tử (N = 10)")
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print("Day so a: ", a)

# 1. Tạo một dãy số a có N phần tử (N = 10)
# Day so a:  [120 183 160 175 145 162 190 160 152 162]
```

2. Các thao tác trên dãy số a
   a. Bình phương các phần tử trong dãy số
   b. Tìm số lượng các phần tử của dãy số
   c. In ra giá trị lớn nhất, giá trị nhỏ nhất và giá trị trung bình
   d. Tính phương sai của dãy số $\delta^2 = \frac{\sum(a_i-\mu)^2}{N-1}$ với $\mu = \frac{\sum a_i}{N}$ và độ lệch chuẩn $var = \sqrt{\delta^2}$

```
print("2. Các thao tác tính toán đơn giản với dãy số a")
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print("a. Binh phuong a: ", a2)
print("\nb. Do dai a: ", na)

""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print(f'\nc. Gia tri lon nhat: [{v_max}], gia tri nho nhat: [{v_min}], gia tri trung binh
[{v_avg}]')

""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print(f'\nd. Phuong sai: [{v_sigma2: .2f}] va do lech chuan: [{v_var: .2f}]')

# 2. Các thao tác tính toán đơn giản với dãy số a
# a. Binh phuong a:  [[38809 15129 12996 23409]
#  [10000 36481 10201 21904]
#  [13456 23716 18225 24025]
#  [29929 22801 18769 23104]
#  [31329 31684 11449 30276]
#  [28561 38025 18496 17689]]
#
# b. Do dai a:  6
#
# c. Gia tri lon nhat: [197], gia tri nho nhat: [100], gia tri trung binh [148.70833333333334]
#
# d. Phuong sai: [ 959276.67] va do lech chuan: [ 979.43]
```

3. Toán tử trên hai dãy số
   a. Tạo dãy số x có 10 phần tử từ 1 đến 10 (dùng hàm np.arange)

```
print("3. Xử lý khác trên dãy số")
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print(f'a. Day so x: {x}')

# 3. Xử lý khác trên dãy số
# a. Day so x: [ 1  2  3  4  5  6  7  8  9 10]
```

b. Liệt kê và đếm số lượng phần tử chẵn và lẻ có trong dãy a và x

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print( "b. Cac phan tu chan trong day a, b: ")
print(f"+ Day a: {a}")
print(f"  - Loc chan: {a_la_chan}")
print(f"  - Day chua phan tu chan trong a: {a_chan}, so luong {n_a_chan} phan tu.")
print(f"+ Day x: {x}")
print(f"  - Loc chan: {x_la_chan}")
print(f"  - Day chua phan tu chan trong a: {x_chan}, so luong {n_x_chan} phan tu.")

# b. Cac phan tu chan trong day a, b:
# + Day a: [120 183 160 175 145 162 190 160 152 162]
#   - Loc chan: [ True False  True False False  True  True  True  True]
#   - Day chua phan tu chan trong a: [120 160 162 190 160 152 162], so luong 7 phan tu.
# + Day x: [9 7 4 ... 1 4 1]
#   - Loc chan: [False False  True ... False  True False]
#   - Day chua phan tu chan trong a: [ 4  6  6 ...  4 10  4], so luong 5042 phan tu.
```

c. Tính khoảng cách giữa các phần tử ở vị trí lẻ của mảng a và x với $s = \sqrt{\sum(a_i - x_i)^2}$ với i lẻ

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print( "c.Tính hiệu các phần tử ở vị trí lẻ của mảng a và x: ")
print(f"+ Vi tri le trong a va x: {v_le}")
print(f"+ Day so a: {a} va cac so o vi tri le: {a_le}")
print(f"+ Day so x: {x} va cac so o vi tri le: {x_le}")
print(f"+ Hieu cua a va x o vi tri le: {hieu_le}")
print(f"+ Khoang cach cac phan tu cua a va x o vi tri le: {kc_le: .2f}")

# c.Tính hiệu các phần tử ở vị trí lẻ của mảng a và x:
# + Vi tri le trong a va x: [1, 3, 5, 7, 9]
# + Day so a: [120 183 160 175 145 162 190 160 152 162] va cac so o vi tri le: [183 175 162
160 162]
# + Day so x: [9 7 4 ... 1 4 1] va cac so o vi tri le: [7 6 5 1 7]
# + Hieu cua a va x o vi tri le: [176 169 157 159 155]
# + Khoang cach cac phan tu cua a va x o vi tri le:  365.37
```

d. Tìm khoảng cách nhỏ nhất giữa tập a và x. Ví dụ: $min(\{1,3,5\},\{2,10,8\}) = 1$

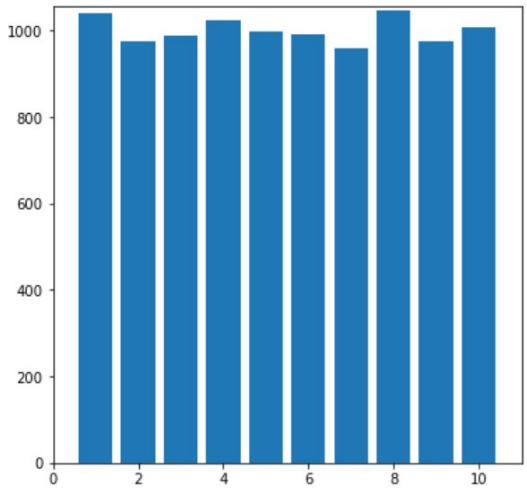
```
print("d. Tìm khoảng cách nhỏ nhất giữa tập a và x. Ví dụ: min({1, 3, 5}, {2, 10, 8}} = 1")
"""
Gợi ý: Áp dụng toán tử broatcasting
[[1],                    [[1, 1, 1],         [[2, 10, 8],    =   [[-1, -9, -7],
 [3],   - [2, 10, 8] =    [3, 3, 3],   -      [2, 10, 8],         [ 1, -7, -5],  -->
np.min(np.abs(...)) = 1
 [5]]                     [5, 5, 5]]          [2, 10, 8]]         [ 3, -5, -3]]
"""
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# Tăng số chiều dùng tại vị trí chiều cuối dùng np.newaxis như sau: a[:, np.newaxis]
# …

print(f"+ Day a: {a}")
print(f"+ Day x: {x}")
print(f"+ Khoang cach nho nhat: {v_min}")

# d. Tìm khoảng cách nhỏ nhất giữa tập a và x. Ví dụ: min({1, 3, 5}, {2, 10, 8}} = 1
# + Day a: [120 183 160 175 145 162 190 160 152 162]
```

```
# + Day x: [ 1  2  3  4  5  6  7  8  9 10]
# + Khoang cach nho nhat: 110
```

4. Sinh ngẫu nhiên dãy số

   a. Sinh ngẫu nhiên dãy số có 100 phần tử theo phân bố đều với giá trị trong đoạn [1, 10] và vẽ đồ thị tần số xuất hiện

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# Sử dụng hàm np.random.randint để sinh dãy số ngẫu nhiên
# Sử dụng hàm np.unique để trả về tần số xuất hiện
# …

print("a. Sinh ngẫu nhiên theo phân bố đều")
print(f"+ Gia tri: {value}")
print(f"+ So lan xuat hien: {cnt}")
print("+ Do thi ham so xuat hien")
plt.figure(figsize=(6,6))       #  kich thuoc ban ve
plt.bar(value, cnt)             #  ve cac khoi chu nhat theo day x, y
plt.xlim(0, 11)                 #  gioi han truc x
plt.ylim(0, np.max(cnt) + 10)   #  gioi han truc y
plt.savefig("4a.png")           #  luu do thi
plt.show()                      #  hien thi do thi

# a. Sinh ngẫu nhiên theo phân bố đều
# + Gia tri: [ 1  2  3  4  5  6  7  8  9 10]
# + So lan xuat hien: [1041  974  988 1024  997  990  958 1047  974 1007]
# + Do thi ham so xuat hien
```



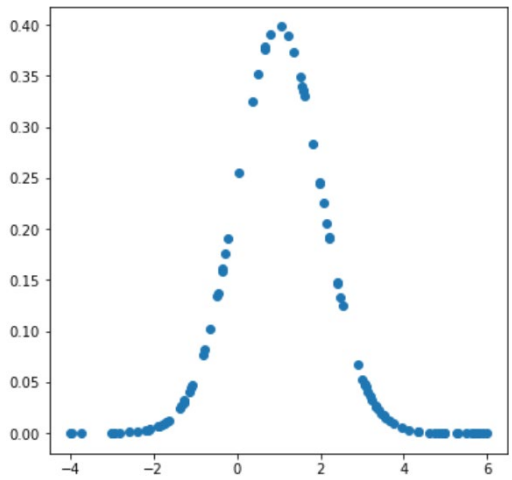
   b. Sinh ngẫu nhiên N diem (x, y) với y là hàm phân phối chuẩn $N(\mu = 1, \sigma^2 = 1.0)$:

$$y = f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

   nơi mà $x \in [\mu - 5\sigma, \mu + 5\sigma]$.

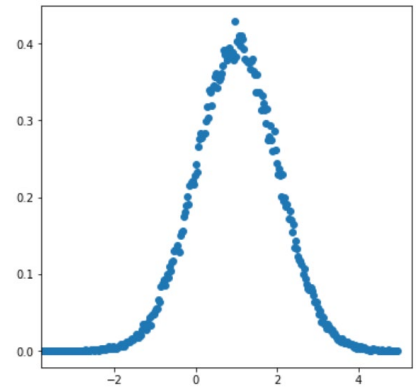
```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# Sử dụm hàm np.random.rand() sinh số ngẫu nhiên trong [0,1) --> [0,1) * (b - a) + a --> [a, b)
# Sử dụng hàm sau để tính hàm xác suất chuẩn N(m, s): np.sqrt, np.exp
# …

print("b. Sinh cặp (x, y) với y làm hàm phân phối chuẩn")
print(f"+ x: {x_min} <= min [{np.min(x): .2f}] <= max [{np.max(x): .2f}] <= {x_max}")
print(f"+ Ve ham so y(x)")
plt.figure(figsize=(6,6))       # kich thuoc ban ve
plt.scatter(x, y)               # ve cac diem (xi, yi)
plt.savefig("4b.png")           #  luu do thi
plt.show()                      #  hien thi do thi

# b. Sinh cặp (x, y) với y làm hàm phân phối chuẩn
```

```
# + x: -4.0 <= min [-4.00] <= max [ 5.99] <= 6.0
# + Ve ham so y(x)
```



c. Sinh ngẫu nhiên dãy số có 100000 phần tử theo phân bố chuẩn $N(\mu = 1, \sigma^2 = 1.0)$ và vẽ đồ thị tần số xuất hiện

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# Sử dụng hàm np.random.randn() sinh phân phối chuẩn N(0,1) --> N(m,s^2) = m + N(0,1) * s
# Sử dụng hàm np.histogram với density=True để thống kê số lần xuất hiện với số bins = 300 (do dãy số thực)
# …

print( "c. Sinh day so theo phan phoi chuan")
print(f"+ x: len = {len(x)}, min [{np.min(x)}], max [{np.max(x)}]")
print(f"+ Mat so xac suat cua day so x")
plt.figure(figsize=(6,6))       # kich thuoc ban ve
plt.xlim(np.min(val), np.max(val))
plt.scatter(val, cnt)
plt.show()

# c. Sinh day so theo phan phoi chuan
# + x: len = 100000, min [-3.8857659357819045], max [4.984806319391394]
# + Mat so xac suat cua day so x
```



Bài 2. Tính toán trên ma trận

1. Tạo ma trận a có M = 6 dòng, N = 4 cột

```
# a = np.random.randint(100, 200, (6, 4))
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print("Ma trận a: \n", a)

# Ma trận a:
#  [[197 123 114 153]
#  [100 191 101 148]
#  [116 154 135 155]
```

```
#  [173 151 137 152]
#  [177 178 107 174]
#  [169 195 136 133]]
```

2. Các thao tác trên ma trận a
    a. In ra ma trận chuyển vị
    b. In ra phần tử ở vị trí dòng 2 cột 3
    c. Trích xuất dòng đầu, dòng cuối, dòng 2
    d. Trích xuất cột kế cuối, cột đầu
    e. Đảo các giá trị trên từng cột
    f. Tính tổng các dòng, trung bình các cột

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print(f"a. In ra ma trận chuyển vị: \n{a_t}")
print(f"\nb. In ra phần tử ở vị trí dòng 2 cột 3: {v_23}")
print(f"\nc. Trích xuất các dòng: ")
print(f"   + Dòng đầu : {d_0}")
print(f"   + Dòng cuối: {d_last}")
print(f"   + Dòng hai : {d_2}")
print(f"\nd. Trích xuất các cột: ")
print(f"   + Cột kế cuối: {c_l1}")
print(f"   + Cột đầu: {c_0}")
print(f"\ne. Đảo các giá trị trên từng cột: \n {dao_cot}")
print(f"\nf. Phép tính theo trục:")
print(f"   + Tổng dòng: {tong_dong}")
print(f"   + Trung bình cột: {tb_cot}")

# a. In ra ma trận chuyển vị:
# [[197 100 116 173 177 169]
#  [123 191 154 151 178 195]
#  [114 101 135 137 107 136]
#  [153 148 155 152 174 133]]
#
# b. In ra phần tử ở vị trí dòng 2 cột 3: 155
#
# c. Trích xuất các dòng:
#    + Dòng đầu : [197 123 114 153]
#    + Dòng cuối: [169 195 136 133]
#    + Dòng hai : [116 154 135 155]
#
# d. Trích xuất các cột:
#    + Cột kế cuối: [114 101 135 137 107 136]
#    + Cột đầu: [197 100 116 173 177 169]
#
# e. Đảo các giá trị trên từng cột:
#  [[169 195 136 133]
#  [177 178 107 174]
#  [173 151 137 152]
#  [116 154 135 155]
#  [100 191 101 148]
#  [197 123 114 153]]
#
# f. Phép tính theo trục:
#    + Tổng dòng: [587 540 560 613 636 633]
#    + Trung bình cột: [155.33333333 165.33333333 121.66666667 152.5        ]
```

3. Các toán tử trên hai ma trận
    a. Tạo hai ma trận A và B có 4 dòng x 3 cột và X có 3 dòng x 4 cột
    b. Tính tổng, hiệu, tích của từng phần tử trên hai ma trận A và B
    c. Tính phép nhân ma trận $Y = A \times X$

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print("a. Tạo hai ma trận A và B có 4 dòng x 3 cột và X có 3 dòng x 4 cột")
print(f"+ Ma tran A: \n{A}")
print(f"+ Ma tran B: \n{B}")
print(f"+ Ma tran X: \n{X}")

print("\nb. Tính tổng, hiệu, tích của từng phần tử trên hai ma trận A và B")
print(f"A + B = \n{AB_add}")
print(f"A - B = \n{AB_sub}")
print(f"A . B = \n{AB_mul}")

print("\nc. Tính phép nhân ma trận Y = A x X")
print(f"Y = A x X --> shape A ({A.shape}) x shape X ({X.shape}) = shape Y ({Y.shape}) \n{Y}")

# a. Tạo hai ma trận A và B có 4 dòng x 3 cột và X có 3 dòng x 4 cột
# + Ma tran A:
# [[5 5 4]
#  [2 2 5]
#  [5 5 5]
#  [2 5 2]]
# + Ma tran B:
# [[1 4 4]
#  [3 1 2]
#  [3 1 2]
#  [5 1 2]]
# + Ma tran X:
# [[5 2 5 5]
#  [5 2 2 3]
#  [4 5 3 4]]
#
# b. Tính tổng, hiệu, tích của từng phần tử trên hai ma trận A và B
# A + B =
# [[6 9 8]
#  [5 3 7]
#  [8 6 7]
#  [7 6 4]]
# A - B =
# [[ 4  1  0]
#  [-1  1  3]
#  [ 2  4  3]
#  [-3  4  0]]
# A . B =
# [[ 5 20 16]
#  [ 6  2 10]
#  [15  5 10]
#  [10  5  4]]
#
# c. Tính phép nhân ma trận Y = A x X
# Y = A x X --> shape A ((4, 3)) x shape X ((3, 4)) = shape Y ((4, 4))
# [[66 40 47 56]
#  [40 33 29 36]
#  [70 45 50 60]
#  [43 24 26 33]]
```

4. Giải hệ phương trình tuyến tính

$$\begin{cases} 4x_1 + 3x_2 - 5x_3 = 2 \\ -2x_1 - 4x_2 + 5x_3 = 5 \\ 8x_1 + 8x_2 = -3 \end{cases}$$

```
""" CÁC BẠN LÀM BÀI Ở ĐÂY """
# …

print(f"+ Ma tran A: \n{A}")
print(f"+ Vector y: {y}")
print(f"+ Nghiem x: {x}")
print(f"+ Kiem tra: yy = Ax = {yy} ==> ||yy -y ||_2 = {diff: .2f}")
```

```
# + Ma tran A:
# [[ 4  3 -5]
#  [-2 -4  5]
#  [ 8  8  0]]
# + Vector y: [ 2  5 -3]
# + Nghiem x: [ 2.20833333 -2.58333333 -0.18333333]
# + Kiem tra: yy = Ax = [ 2.  5. -3.] ==> ||yy -y ||_2 =  0.00
```

3. Bài tập áp dụng

Bài 3. Tính gần đúng số pi dùng phương pháp Monte Carlo

Vẽ một đường tròn C có bán kính r nội tiếp một hình vuông C, và đặt ngẫu nhiên N chấm lên hình vuông. Tỉ lệ các chấm nằm trong hình tròn trên tổng số chấm xấp xỉ bằng diện tích của hình tròn chia cho hình vuông như sau:

$$\frac{N_C}{N} \approx \frac{S_C}{S_N}$$

nơi mà N_C là các chấm trong hình tròn, N là tổng số chấm, S_C và S_N lần lượt là diện tích của hình tròn và hình vuông.

$$S_C = \pi * r^2$$

$$S_N = (2 * r)^2 = 4r^2$$

Do đó, công thức trên trở thành:

$$\frac{N_C}{N} \approx \frac{\pi * r^2}{4r^2} = \frac{\pi}{4}$$

Suy ra, ta có giá trị số π được tính như sau:

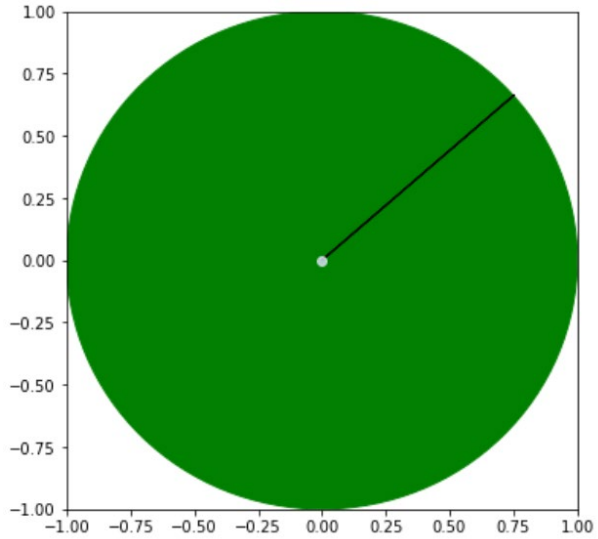
$$\pi \approx \frac{4N_C}{N}$$

Các bạn hãy lập trình tính sấp xỉ số π dùng phương pháp Monte Carlo. Cho biết các sai số với $N = 100$, $N = 10000$, và $N = 1000000$. Vẽ đồ thị minh họa cho chương trình.

Hướng dẫn: Viết chương trình phát sinh N điểm (x_i, y_i) với $x_i, y_i \in [-1, -1]$. (x_i, y_i) thuộc hình tròn tâm $(0,0)$ bán kính 1 nội tiếp hình vuông có tâm tại $(0,0)$ độ dài cạnh 2 khi và chỉ khi $x_i^2 + y_i^2 \leq 1$.

- Hiển thị hình ảnh minh họa hình tròn tâm (0,0) bán kính 1 nội tiếp hình vuông tâm (0,0) có độ dài 2

```
import matplotlib.pyplot as plt # khai báo thư viện vẽ pyplot
import math

circle = plt.Circle((0, 0), 1, color='g') # tạo đối tượng vòng tròn

fig, ax = plt.subplots(figsize=(6, 6)) # lấy figure và vùng vẽ

plt.plot(0, 0, 'o', color=(0.9, 0.9, 1.0), alpha=0.8) # vẽ điểm tại tâm (0,0)
ax.add_patch(circle) # thêm vòng tròn vào vùng vẽ

x = 0.75; y = math.sqrt(1 - (x ** 2)) # (x = 0.75, y = căn bậc 2 của 1 - x^2)
plt.arrow(0, 0, x, y)

plt.xlim(-1, 1) # giới hạn trục x
plt.ylim(-1, 1) # giới hạn trục y

plt.savefig('pi.png') # lưu đồ thị xuống tập tin
plt.show()            # hiển thị đồ thị
```



- Cài đặt hàm tính toán số π

```python
import numpy as np


def calc_pi_monte_carlo(n=100):
    pi = 0

    # gợi ý: sử dụng np.random.rand(s1, s2, ...) --> sinh ngẫu nhiên các số trong nửa đoạn [0.0, 1.0)
    """ CÁC BẠN LÀM BÀI Ở ĐÂY """
    # …

    return pi
# calc_pi_monte_carlo
```

- Kiểm tra hàm tính toán số π

```python
import math

print("epsilon(n=100): ", calc_pi_monte_carlo(n = 100) - math.pi)
print("epsilon(n=100): ", calc_pi_monte_carlo(n = 100) - math.pi)
print("epsilon(n=10000): ", calc_pi_monte_carlo(n = 10000) - math.pi)
print("epsilon(n=10000): ", calc_pi_monte_carlo(n = 10000) - math.pi)
print("epsilon(n=1000000): ", calc_pi_monte_carlo(n = 1000000) - math.pi)
print("epsilon(n=1000000): ", calc_pi_monte_carlo(n = 1000000) - math.pi)

# epsilon(n=100):  0.0984073464102071
# epsilon(n=100):  -0.061592653589793045
# epsilon(n=10000):  -0.04559265358979303
# epsilon(n=10000):  -0.009192653589793043
# epsilon(n=1000000):  -0.0020086535897929636
# epsilon(n=1000000):  -0.0008486535897929137
```

--- Hết ---