Pogosian S A_KVBO-07-23_WorkBook 1

September 22, 2024

[1]:
```
x = 5 >= 2
A = {1, 3, 5, 7}
B = {2, 4, 5, 10, 'apple'}
C = A & B
df = '          ', 34, ' '
z = 'type'
D = [1, 'title', 2, 'content']

print(f"{x} | {type(x)}\n{A} | {type(A)}\n{B} | {type(B)}\n{C} |␣
 ↪{type(C)}\n{df} | {type(df)}\n{z} | {type(z)}\n{D} | {type(D)}")
```

```
True | <class 'bool'>
{1, 3, 5, 7} | <class 'set'>
{2, 4, 5, 10, 'apple'} | <class 'set'>
{5} | <class 'set'>
('          ', 34, ' ') | <class 'tuple'>
type | <class 'str'>
[1, 'title', 2, 'content'] | <class 'list'>
```

[2]:
```
x = int(input())

if x < -5:
    print("X is less then -5")
elif -5 <= x <= 5:
    print("X is between -5 and 5")
else:
    print("X is higher then 5")
```

```
 10
```

```
X is higher then 5
```

[3]:
```
x = 10
while x >= 1:
    print(x)
    x -= 3
```

```
10
7
```

```
4
1
```

[4]:
```
a = ["    ", "   ", "    ", "      ", "   ", "  ", "   "]

print(*a)
```

[5]:
```
a = [i for i in range(2, 16)]
print(*a, sep=", ")
```

```
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
```

[6]:
```
for i in range(105, 4, -25):
    print(i)
```

```
105
80
55
30
5
```

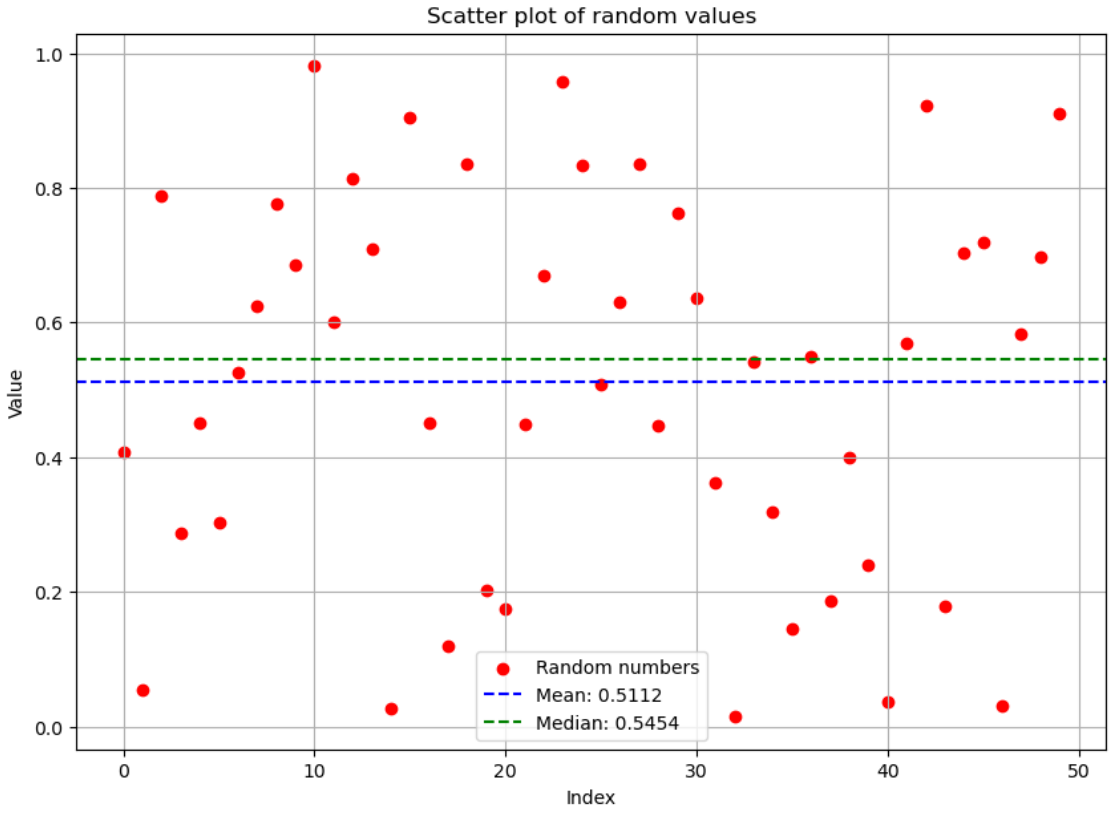
[7]:
```
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

even_x = x[-2::-2]

x[::2] = even_x

print(x)
```

```
[8, 1, 6, 3, 4, 5, 2, 7, 0, 9]
```

[9]:
```
import numpy as np
import matplotlib.pyplot as plt

n = int(input("              : "))

np.random.seed(40)

arr = np.random.rand(n)

mean_val = np.mean(arr)
median_val = np.median(arr)

print("Array: ", arr)

print(f"Mean value: {mean_val}, median value: {median_val}")

plt.figure(figsize=(10, 7))
```

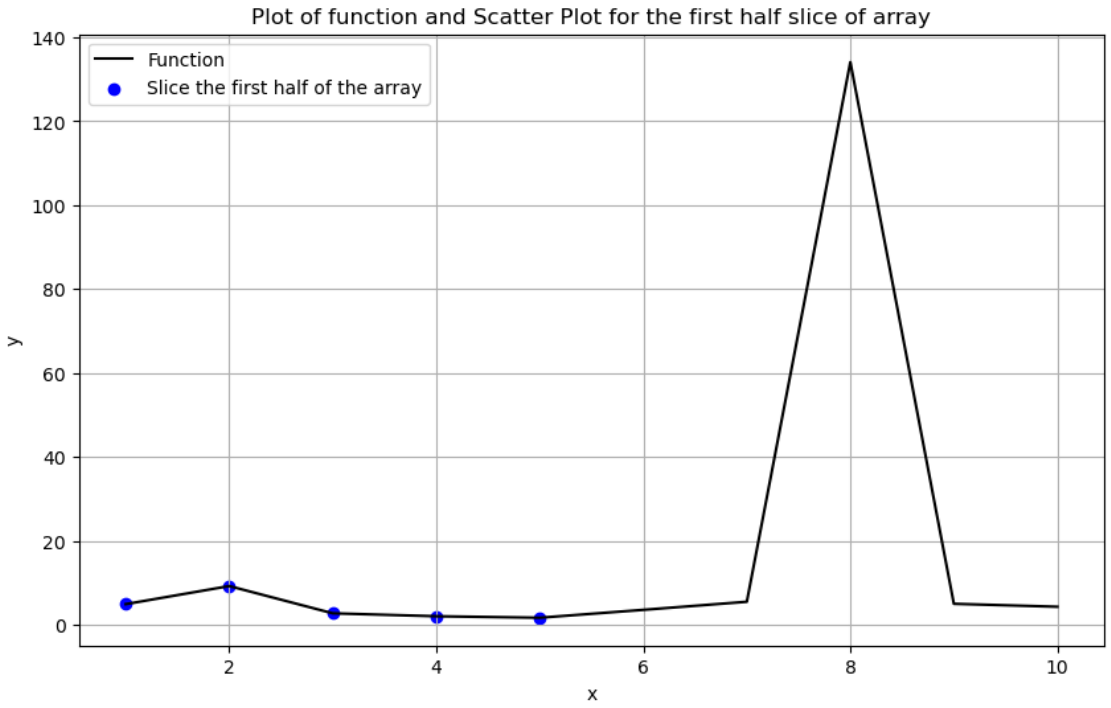
```
plt.scatter(range(n), arr, color="red", label="Random numbers")
plt.axhline(mean_val, color="blue", linestyle="--",
            label=f"Mean: {mean_val:.4f}")
plt.axhline(median_val, color="green", linestyle="--",
            label=f"Median: {median_val:.4f}")

plt.title("Scatter plot of random values")
plt.xlabel("Index")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()
```

```
              :  50
Array:  [0.40768703 0.05536604 0.78853488 0.28730518 0.45035059 0.30391231
 0.52639952 0.62381221 0.77677546 0.68624165 0.98093886 0.60081609
 0.81396852 0.70864515 0.02753468 0.90426722 0.44990485 0.11892465
 0.83530018 0.20224823 0.17420267 0.44914708 0.66979478 0.95739911
 0.83333325 0.50830996 0.63002355 0.83503469 0.44733165 0.76229047
 0.63692224 0.36229589 0.01457455 0.54198489 0.31815548 0.14499035
 0.54889195 0.18748127 0.39898148 0.24003821 0.03665485 0.56854476
 0.9227648  0.17905511 0.70307767 0.7200128  0.03106402 0.58282531
 0.6976776  0.90985347]
Mean value: 0.5112329435957045, median value: 0.5454384191188367
```



[10]:
```python
from math import sqrt, e, pow, cos, sin
import matplotlib.pyplot as plt
import numpy as np


def func(x: int):
    numerator = sqrt(1 + pow(e, sqrt(x)) + pow(cos(x), 2))
    denominator = abs(1 - pow(sin(x), 3))
    result = numerator / denominator
    return result


x_values = np.arange(1, 11)
y_values = np.array([func(i) for i in x_values])

half_x_values = x_values[:len(x_values) // 2]
half_y_values = y_values[:len(y_values) // 2]

plt.figure(figsize=(10, 6))

plt.plot(x_values, y_values, label="Function", color="black")
```

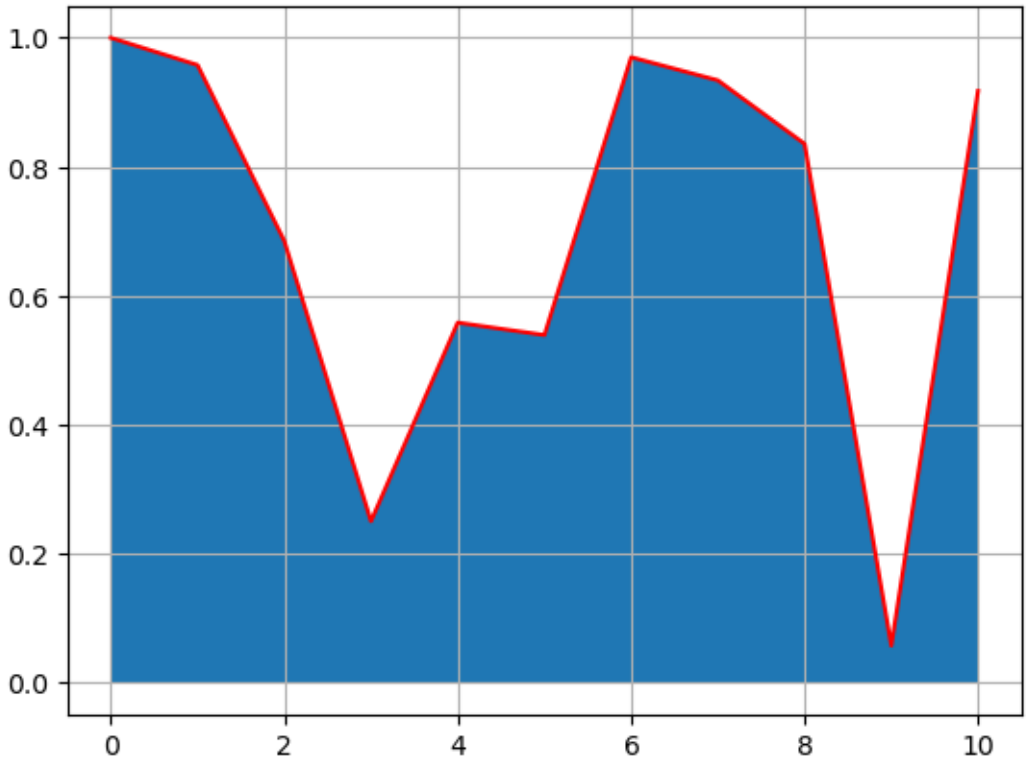
```
plt.scatter(half_x_values, half_y_values, color="blue",
            label="Slice the first half of the array")

plt.title(
    "Plot of function and Scatter Plot for the first half slice of array")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```



[11]:
```
from math import e, cos, log, pow
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps


def func(val: int):
    result = abs(cos(val * pow(e, cos(val) + log(val + 1))))
    return result


x_value = np.arange(0, 11)
```

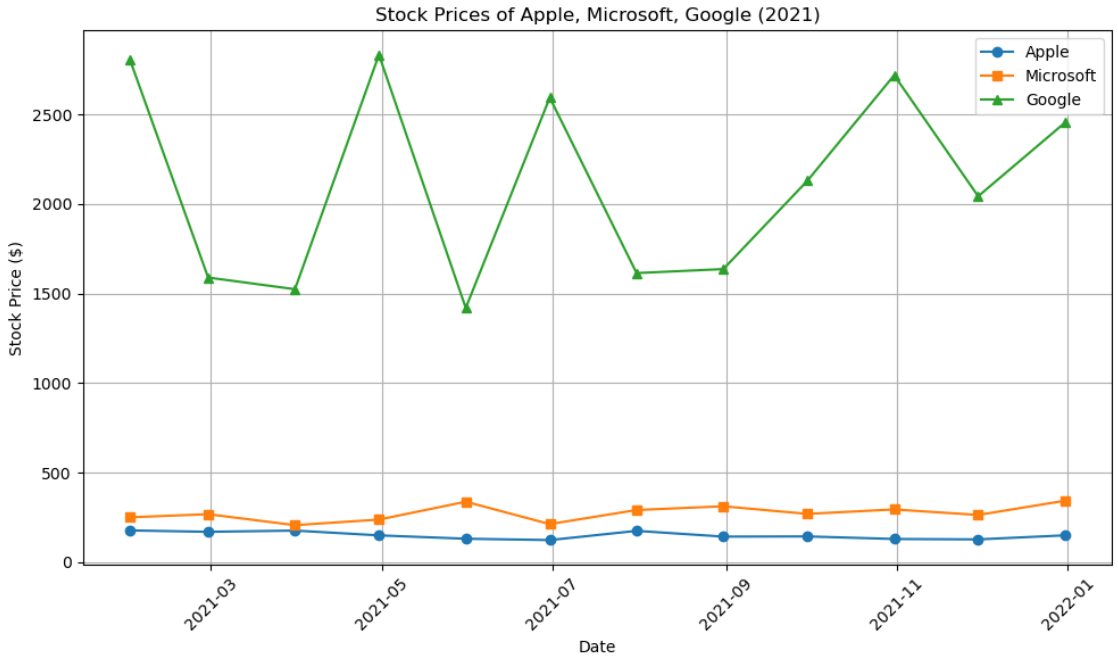
```
y_value = np.array([func(i) for i in x_value])
plt.grid()
plt.plot(x_value, y_value, c='r')
plt.fill_between(x_value, y_value)

area = np.trapz(y_value)
print(area)
```

```
6.748183214657723
```



[12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


dates = pd.date_range(start="2021-01-01", end="2021-12-31", freq='M')


apple_prices = np.random.uniform(low=120, high=180, size=len(dates)).round(2)
microsoft_prices = np.random.uniform(low=200, high=350, size=len(dates)).round(
    2)
google_prices = np.random.uniform(low=1400, high=3000, size=len(dates)).round(
```

```
    2)


stock_data = pd.DataFrame({
    'Date': dates,
    'Apple': apple_prices,
    'Microsoft': microsoft_prices,
    'Google': google_prices
})


plt.figure(figsize=(10, 6))
plt.plot(stock_data['Date'], stock_data['Apple'], label="Apple", marker='o')
plt.plot(stock_data['Date'], stock_data['Microsoft'], label="Microsoft",
         marker='s')
plt.plot(stock_data['Date'], stock_data['Google'], label="Google", marker='^')


plt.title('Stock Prices of Apple, Microsoft, Google (2021)')
plt.xlabel('Date')
plt.ylabel('Stock Price ($)')
plt.grid(True)
plt.legend()


plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[13]:
```
from math import e, sin, cos, pow

x = int(input("x: "))
y = int(input("y: "))

s = input(
    """
    Choose the operator (+, -, *, /,
    e ^ (x + y),
    sin(x + y),
    cos(x + y),
    x ^ y\n
    Operator: """)

match s:
    case "+":
        print("x + y = ", x + y)
    case "-":
        print("x - y = ", x - y)
    case "*":
        print("x * y = ", x * y)
    case "/":
        if y == 0:
            print("Divide by zero!")
        else:
            print("x / y = ", x / y)
    case "e ^ (x + y)":
        print("e ^ (x + y) = ", pow(e, x + y))
    case "sin(x + y)":
        print("sin(x + y) = ", sin(x + y))
    case "cos(x + y)":
        print("cos(x + y) = ", cos(x + y))
    case "x ^ y":
        print("x ^ y = ", pow(x, y))
    case _:
        print("Sorry, calculator does not understand you(")
```

```
x:  10
y:  12

    Choose the operator (+, -, *, /,
    e ^ (x + y),
    sin(x + y),
    cos(x + y),
    x ^ y
```

```
    Operator:  +
x + y =  22
```

[ ]: