DSP Homework1

전자공학과 3학년 12191468_남국현

Unit sample sequence – generate

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

# 변수 설정
start = 0
end = 20
impulse = 3

# 배열 생성
dt_seq = np.arange(start, end+1)

# 변수와 배열 상태 출력
print(f"Start: {start}, End: {end}, Impulse: {impulse}")
print("Sequence length: ", end-start)
print("Discrete-time sequence: {}".format(dt_seq))
```

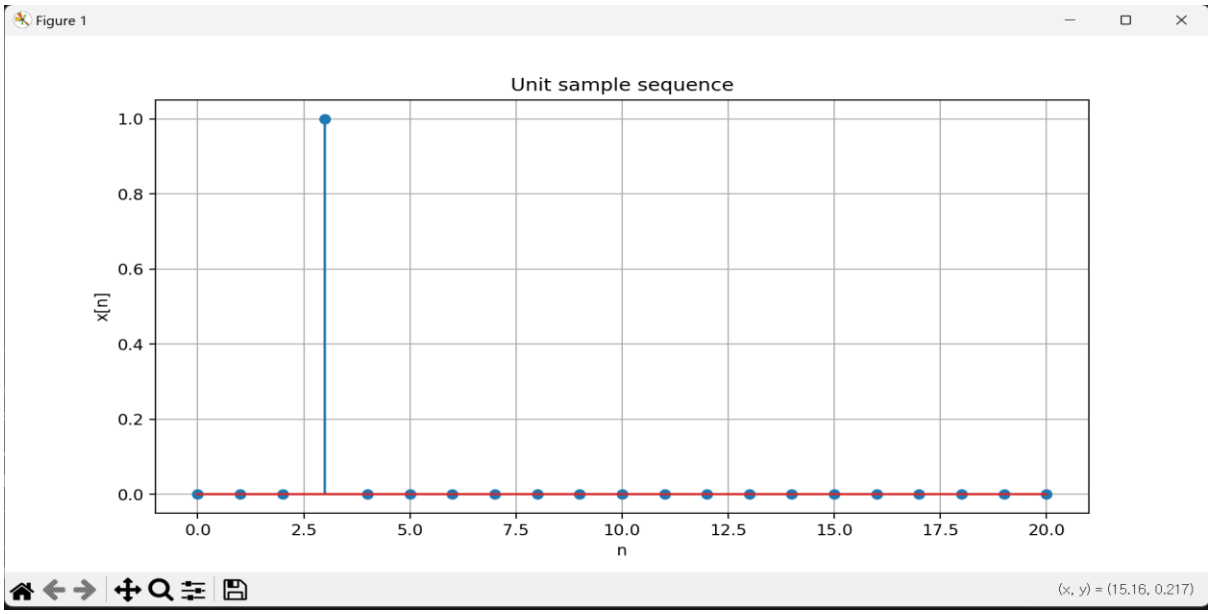
```
C:\Users\남국현\OneDrive - 인하대학교\바탕 화면\DSP\HW> cmd /C ""c:\Program Files\Python311
n32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher 63485 -- "c:\Users\남국현\OneDr
Start: 0, End: 20, Impulse: 3
Sequence length:  20
Discrete-time sequence: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

```python
# 리스트 표현식을 통해 Unit sample sequence 작성
unit_sample_list = [1 if (i+start) == impulse else 0 for i in range(end+1)]
unit_sample_array = np.array(unit_sample_list)
```

```
Unit sample sequence: [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Unit sample sequence – plot

```
# Unit sample sequence 값 출력
print(f"Unit sample sequence: {unit_sample_array}")

plt.figure(figsize = (10, 5)) # 그래프 크기를 10x5로 설정
plt.stem(dt_seq, unit_sample_array); plt.grid() # dt_seq와 unit_sample_array를 이용해 그래프를 그림, 격자선 추가
plt.title("Unit sample sequence") # 그래프 제목 설정
plt.xlabel("n"); plt.ylabel("x[n]") # 축 라벨 지정

plt.show() # 그래프를 화면에 표시
```



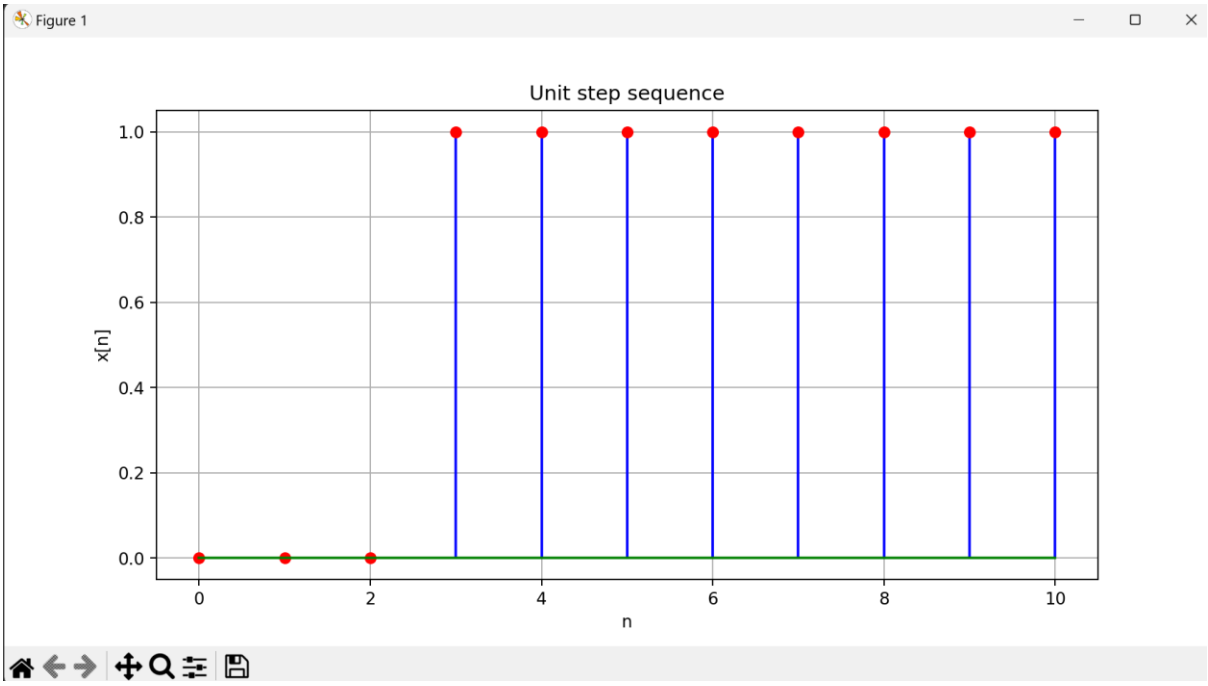
Unit step sequence – generate

```
# 새로운 값으로 변수 재설정
start = 0
end = 10
impulse = 3

# 배열을 다시 만들고, 리스트 표현식으로 Unit step sequence 생성
dt_seq = np.arange(start, end+1)
unit_sample_list = [1 if (i+start) >= impulse else 0 for i in range(end+1)]
unit_sample_array = np.array(unit_sample_list)

# Unit step sequence 값 출력
print(f"Unit sample sequence: {unit_sample_array}")

# np.zeros를 사용해 같은 작업을 수행
unit_sample_array = np.zeros(end+1, dtype=int)
unit_sample_array[impulse:] = 1

# Unit step sequence 값 다시 출력
print(f"Unit sample sequence: {unit_sample_array}")
```

```
Unit sample sequence: [0 0 0 1 1 1 1 1 1 1 1]
Unit sample sequence: [0 0 0 1 1 1 1 1 1 1 1]
```

Unit step sequence – plot

```
51  # 그래프의 크기 설정
52  plt.figure(figsize= (10, 5))
53  # stem plot을 만들고, 그래프의 각 요소에 색상 설정을 위한 변수 추가
54  markers, stemlines, baseline = plt.stem(dt_seq, unit_sample_array); plt.grid()
55
56  # 각 그래프 요소에 색상을 적용
57  markers.set_color("red")
58  stemlines.set_color("blue")
59  baseline.set_color("green")
60
61  # 그래프 제목과 축 라벨 설정
62  plt.title("Unit step sequence")
63  plt.xlabel("n"); plt.ylabel("x[n]")
64
65  # 그래프 출력
66  plt.show()
```



Periodic sequence – generate

```
68  # 정현파 DT 신호
69  n = np.linspace(0, 4*np.pi, 64) # 0에서 4π 사이를 64개의 구간으로 나누기
70  amp = 3 # 진폭 값을 3으로 설정
71  omega = 3*np.pi/8 # 각주파수를 3π/8로 설정
72  phase = np.pi/2 # 위상 값을 π/2로 설정
73  xn = amp * np.cos(omega*n + phase) # 정의된 변수들을 바탕으로 코사인 함수 생성
74
75  N = 2*np.pi / omega # 신호 주기 계산
76  print(f"Period: {N}, if k == 1") # 주기 출력
```

```
Period: 5.333333333333333, if k == 1
```

Periodic sequence – plot

```
78  # 그래프 그리기
79  plt.figure(figsize= (10, 5)) # 그래프 사이즈 설정
80  plt.stem(n, xn, basefmt="blue"); plt.grid() # 이산 신호를 stem plot으로 표현하고, 파란색 베이스라인 추가
81  plt.title("Periodic sequence") # 그래프 제목 설정
82  plt.xlabel("n"); plt.ylabel("x[n]") # x축과 y축 라벨 설정
83
84  # 그래프 출력
85  plt.show()
86
```