	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Analysis of Discrete-Time Signals Using Z-Transform	
Experiment No: 17	**Date:**	**Enrollment No: 92400133055**

GITHUB

Aim: Analysis of Discrete-Time Signals Using Z-Transform
IDE: Visual Studio Code

Install Library pip install sympy
#Example 1: import sympy as sp
Define symbols
n, z, a = sp.symbols('n z a')
 # Define the signal x[n] = a^n *u[n]
x_n = a**n
# Compute the Z-transform
X_z = sp.summation(x_n * z**(-n), (n, 0, sp.oo))
# Print the result
print("Z-transform of x[n] = a^n u[n]:")
sp.pprint(X_z, use_unicode=True)
Output:

```
Z-transform of x[n] = a^n u[n]:
⎧    1              │a│
⎪ ────────     for  │─│ < 1
⎪    a              │z│
⎪ - ─ + 1
⎪    z
⎪
⎨  ∞
⎪ ___
⎪ ╲
⎪  ╲    n  -n
⎪  ╱   a ⋅z      otherwise
⎪ ╱
⎪ ‾‾‾
⎩n = 0
```

#Example 2:
Define symbols n, z, a =
sp.symbols('n z a') # Define the
signal x[n] = a^n * u[n] x_n = 2**n
# Compute the Z-transform
X_z = sp.summation(x_n * z**(-n), (n, 0, sp.oo))
# Print the result

print("Z-transform of x[n] = a^n u[n]:")

sp.pprint(X_z, use_unicode=True) Output:

```
Z-transform of x[n] = a^n u[n]:
⎧     1              1
⎪  ───────      for ─── < 1/2
⎪       2           │z│
⎪  1 - ─
⎪       z
⎪
⎪ ∞
⎨ ___
⎪ ╲
⎪  ╲    n  -n
⎪  ╱   2 ⋅z        otherwise
⎪ ╱
⎪ ‾‾‾
⎪n = 0
⎩
```

#Example 3: import sympy as sp
Define symbols n, z =
sp.symbols('n z') # Define the
unit step signal u[n] u_n = 1 #
Compute the Z-transform
U_z = sp.summation(u_n * z**(-n), (n, 0, sp.oo))
# Print the result print("Z-transform of the unit
step signal u[n]:") sp.pprint(U_z,
use_unicode=True) Output:

```
Z-transform of the unit step signal u[n]:
⎧   1           1
⎪ ─────    for ─── < 1
⎪     1        │z│
⎪ 1 - ─
⎪     z
⎪
⎪ ∞
⎨ ___
⎪ ╲
⎪  ╲   -n
⎪  ╱  z      otherwise
⎪ ╱
⎪ ‾‾‾
⎪n = 0
⎩
```

	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Analysis of Discrete-Time Signals Using Z-Transform	
Experiment No: 17	**Date:**	**Enrollment No: 92400133055**

#Example 4: import
sympy as sp #
Define symbols
n, z, alpha = sp.symbols('n z alpha') # Define
the signal x[n] = exp(alpha * n) * u[n] x_n =
sp.exp(alpha * n) # Compute the Z-transform
X_z = sp.summation(x_n * z**(-n), (n, 0, sp.oo))
# Print the result
print("Z-transform of x[n] = exp(alpha * n) u[n]:")
sp.pprint(X_z, use_unicode=True) Output:

```
Z-transform of x[n] = exp(alpha * n) u[n]:
  ∞
 ___
 ╲
  ╲    -n  α·n
  ╱   z  ·e
 ╱
 ‾‾‾
n = 0
```

#Example 5: import
sympy as sp # Define
symbols n, z =
sp.symbols('n z')
# Define the finite sequence x[n] = {1, 2, 3} x_n
= [1, 2, 3]
# Compute the Z-transform manually
X_z = sum(x_n[i] * z**(-i) for i in range(len(x_n)))
# Print the result
print("Z-transform of the finite sequence {1, 2, 3}:")
sp.pprint(X_z, use_unicode=True) Output:

```
Z-transform of x[n] = exp(alpha * n) u[n]:
  ∞
 ___
 ╲
  ╲    -n  α·n
  ╱   z  ·e
 ╱
 ‾‾‾
n = 0
```

	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Analysis of Discrete-Time Signals Using Z-Transform	
Experiment No: 17	**Date:**	**Enrollment No: 92400133055**

#Example 6 import
sympy as sp #
Define symbols
n, z, omega = sp.symbols('n z omega')
# Define the sinusoidal sequence x[n] = sin(omega * n) * u[n]
x_n = sp.sin(omega * n) # Compute the Z-transform
X_z = sp.summation(x_n * z**(-n), (n, 0, sp.oo))
# Print the result
print("Z-transform of x[n] = sin(omega * n) u[n]:")
sp.pprint(X_z, use_unicode=True) Output:

```
Z-transform of x[n] = sin(omega * n) u[n]:
  ∞
 ___
 ╲
  ╲    -n
  ╱   z  ·sin(n·ω)
 ╱
 ‾‾‾
n = 0
```

Post Lab Exercise:

- Using Python, compute the Z-transform of the sequence $x[n] = 3^n u[n]$.

Code:

```
import sympy as sp n, z, a =
sp.symbols('n z a') x_n =
a**n
X_z = sp.summation(x_n * z**(-n), (n, 0,
sp.oo)) print("Z-transform of x[n] = a^n
u[n]:") sp.pprint(X_z, use_unicode=True) X_z_3
= X_z.subs(a, 3) print("Z-transform of x[n] =
3^n u[n]:") sp.pprint(X_z_3, use_unicode=True)
```

Output:

```
Z-transform of x[n] = a^n u[n]:
⎧    1              │a│
⎪  ─────      for   │─│ < 1
⎪    a              │z│
⎪  - ─ + 1
⎪    z
⎪
⎪   ∞
⎪  ___
⎨  ╲
⎪   ╲    n  -n
⎪   ╱   a ·z      otherwise
⎪  ╱
⎪  ‾‾‾
⎪ n = 0
⎩
Z-transform of x[n] = 3^n u[n]:
⎧    1              1
⎪  ─────      for  ─── < 1/3
⎪      3           │z│
⎪  1 - ─
⎪      z
⎪
⎪   ∞
⎪  ___
⎨  ╲
⎪   ╲    n  -n
⎪   ╱   3 ·z       otherwise
⎪  ╱
⎪  ‾‾‾
⎪ n = 0
⎩
```

- Using Python, compute the Z-transform of the sequence $x[n] = \cos(wn)u[n]$. Code:

```
import sympy as sp import
math
n, z, w = sp.symbols('n z w') x_n
= sp.cos(w*n)
X_z = sp.summation(x_n * z**(-n), (n, 0, sp.oo)) print("Z-
transform of x[n] = cos(wn)u[n]:") sp.pprint(X_z,
use_unicode=True)
```

Output:

```
Z-transform of x[n] = cos(wn)u[n]:
  ∞
 ___
 ╲
  ╲    -n
  ╱   z   ⋅cos(n⋅w)
 ╱
 ‾‾‾
n = 0
```