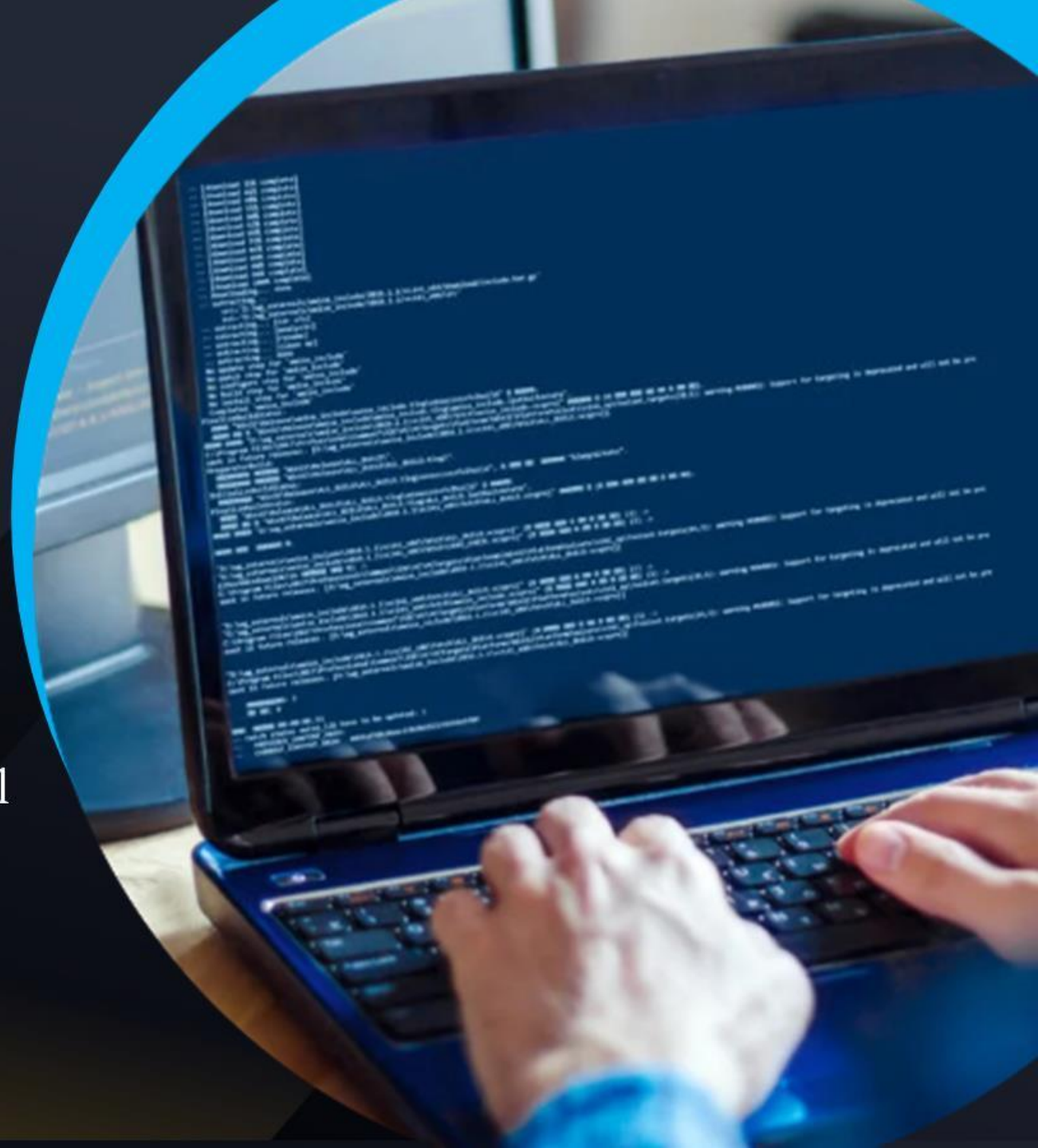


Programming Foundations in Python

Prepared by Dr. Hamada I. AbdulWakel

Lecture 4





مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

String Data Type

Lecture 4

Prepared by
Dr. Hamada I. AbdulWakel

www.midocean.ae

RECAP: LECTURE 3

- The relational operators and comparison expressions.
- The selection statements.
- The iteration statements.

Lecture

04

LECTURE MAIN POINTS

- String data type.
- String features.
- String with conditional and looping.
- String functions.

STRING DATA TYPE

- String declaration.
 - Using double quotes marks " "
 - `S1 = " this is a string "`
 - Using single quotes marks ' '
 - `S2 = ' this is a string '`
 - Using `str()` function
 - `S3 = str('this is a string')`

STRING DATA TYPE

Cont'd

- When the string itself contains single or double quotes.

`S1 = "He said that "I am from Egypt""` #syntax error

- Solutions:

- Using the scape character

`S1 = "He said that \"I am from Egypt\""`

- Using three-double/single quotes. Also, it used to write a string in multiple lines.

`S1 = """ He said that "I am from Egypt """`

`S2 = ''' He said that
I am from Egypt '''`

STRING DATA TYPE FEATURES

- Immutable: we can not change its content in the same location.
- Subscript operator [] is useful when you want to use the position/index to get the element in this position, as we done in strings to get the character in a specific position.

STRING DATA TYPE FEATURES

Cont'd

- Immutable description.

```
s1 = "Hello"
```

```
s1[2] = 'x' #get run-time error
```

```
print( id( s1 ) )
```

```
s1 += " world"
```

```
print( id( s1 ) ) #get a new memory location
```

Lecture

04

STRING OPERATORS

- The + operator means **concatenation**.

```
s1 = "Hello"
```

```
s2 = 'World'
```

```
print( s1 + s2 )
```

- The * operator means **repeatation**.

```
s1 = "Hello World"
```

```
print( s1 * 3 )
```

STRING FUNCTIONS

- **ord('character'):** to get the ASCII code for the given character.
`print( ord( 'A' ) )`
- **chr(number):** to get the character corresponds to the given number.
`print( chr( 95 ) )`
- **len(string):** to get the string length.
`s1 = "Hello"`
`print( len( s1 ) )`

STRING FUNCTIONS

Cont'd

- **max(string):** to get the maximum letter that has max ASCII code in the given string.

```
s1 = "ESU"  
print( max( s1 ) )
```

- **center(width, character):** to center align the string, using a specified character (space is default) as the fill character.

```
s3 = "ESU"  
print(s3.center(10))    #here, 10 is the total number of characters  
print(s3.center(10, "*"))
```

STRING FUNCTIONS

Cont'd

- **strip()**: to remove the start and end whitespaces in the string. There are two versions of this function **rstrip()** and **lstrip()**. You can specify which character(s) to remove, if not, any whitespaces will be removed.

```
s2 = "    Hello    "
print( s2.strip())
print(s2.rstrip())
print(len(s2.rstrip()))
s2 = "@@@Hello@@@"
print( s2.strip("@"))
```

STRING FUNCTIONS

Cont'd

- **isalpha()**: to check if the string contains **only** alphabetic.
`s1 = "Hello123"`
`print(s1.isalpha())`
- **isdigit()**: to check if the string contains **only** numbers.
`s1 = "123"`
`print(s1.isdigit())`
- **isalnum()**: to check if the string contains alphabetic and number.
`s1 = "Hello123"`
`print(s1.isalnum())`

STRING FUNCTIONS

Cont'd

- **islower()**: to check if the string is written in lower case.
`s1 = "ESU"`
`print(s1.islower())`
- **isupper()**: to check if the string is written in upper case.
`s1 = "ESU"`
`print(s1.isupper())`
- **isspace()**: to check if the string contains whitespaces.
`s1 = " "`
`print(s1.isspace())`

STRING FUNCTIONS

Cont'd

- **startswith(substring):** to check if the string begins with the given substring.

```
s1 = "ESU"  
print(s1.startswith("H"))
```
- **endswith(substring):** to check if the string ends with the given substring.

```
s1 = "ESU"  
print(s1.endswith("U"))
```

STRING FUNCTIONS

Cont'd

- **replace(old, new):** to replace each old substring by new substring.

```
s1 = "Midocean U."  
print(s1.replace("U.", "University"))
```

- **split(delimiter):** to split the string according to the delimiter.

```
s1 = "Midocean University"  
print(s1.split())  
s2 = "Midocean&University"  
print(s2.split('&'))
```

STRING FUNCTIONS

Cont'd

- **count(substring):** to count the number of occurrence of the given substring.

```
s1 = "Everyone Smart University"  
print(s1.count("Smart"))
```
- **find(substring):** to find the index of the substring in the string.

```
s1 = "Everyone Smart University"  
print(s1.find("Smart"))
```
- **rfind(substring):** to find the **last** index of the substring in the string.

```
s1 = "Everyone Smart University based on Smart technologies"  
print(s1.rfind("Smart"))
```

STRING FUNCTIONS

Cont'd

- **title()**: to capitalize the first letter of each word in a string.

```
s1 = "midocean university"  
print(s1.title())
```
- **capitalize()**: to capitalize the first letter of a string.

```
s1 = "midocean university"  
print(s1.capitalize())
```
- **swapcase()**: to change each letter case in the string.

```
s1 = "MiDocean University"  
print(s1.swapcase())
```

STRING FUNCTIONS

Cont'd

- **lower()**: to convert the string to lower case.

```
s1 = "Midocean University"  
print(s1.lower())
```
- **upper()**: to convert the string to upper case.

```
s1 = "Midocean University"  
print(s1.upper())
```
- **splitlines()**: to split a string into a list. The splitting is done at line breaks.

```
s1 = "Midocean\nUniversity"  
print(s1.splitlines())
```

STRING FUNCTIONS

Cont'd

- **rjust(width, character):** to right align the string, using a specified character (space is default) as the fill character.

```
s1 = "Everyone Smart University"  
print(s1.rjust(50, "@"))
```

- **ljust(width, character):** to left align the string.

```
s1 = "Everyone Smart University"  
print(s1.ljust(50, '$'))
```

STRING FUNCTIONS

Cont'd

- **expandtabs(number):** to set the tab size to the specified number of whitespaces.

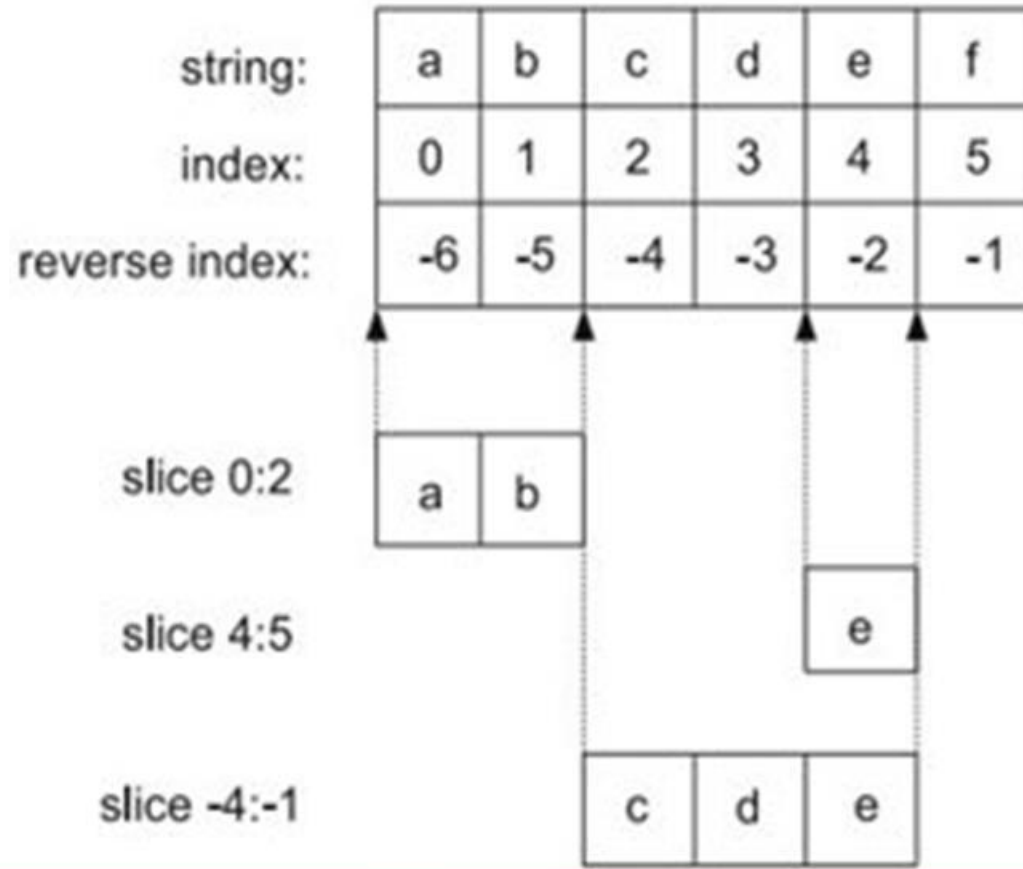
```
s1 = "Everyone\tSmart\tUniversity"  
print(s1.expandtabs(5))
```
- **isidentifier():** to returns True if the string is a valid identifier, otherwise False.

```
s1 = "2ESU"  
print(s1.isidentifier())
```

Lecture

04

STRING SLICING



STRING SLICING

Cont'd

- Python's subscript operator can be used to obtain a substring through a process called slicing.
 - Using [start : stop]
 - Note: **stop - 1**
 - Using [start : stop : step]
 - If Step is **positive**:
 - Step - 1: the number of elements to skip
 - Stop - 1: the stop index
 - If Step is **negative**:
 - Step + 1: the number of elements to skip
 - Stop + 1: the stop index

STRING SLICING

Cont'd

- Code.

```
S = "Hello ESU"  
print( S[ 1 ] )  
print( S[ -1 ] )  
index = 10  
print( S[ index ] )  
print( S[ : 3 ] )  
print( S[ 3 : ] )
```

```
#IndexError  
# from 0 to 3-1  
# to the end
```

STRING SLICING

Cont'd

- Code.

```
S = "Hello ESU"  
print( S[ 2 : 5 ] )  
print( S[ -5 : -1 ] )  
print( S[ 0 : 9 : 2 ] )  
print( S[ : : 2 ] )  
print( S[ : : -2 ] )
```

#here, -1 -1 = -2

STRING AND CONDITIONAL OPERATORS

- When using `in` operator, it returns `True` if target string is somewhere in search string, or `False` otherwise.

```
s1 = "I love Python"  
print( "I" in s1 )  
print( " " in s1 )  
print( "x" in s1 )
```

STRING AND LOOPING

- We can loop over a string as python deals with it as a **sequence**.
- A program that print each character of a given string from the end.
 - Code for help:

```
s = "Hello"  
length = len(s)  
while length > 0:  
    print( s[ length - 1 ] )  
    length -= 1
```

STRING AND LOOPING

Cont'd

- A program that accepts a string and calculates the number of digits and letters.
 - Code for help:

```
s = input("Input a string")
dig = lett = 0
for chr in s :
    if chr.isdigit():
        dig = dig + 1
    elif chr.isalpha():
        lett = lett + 1
    else: pass
print("Letters", lett)
print("Digits", dig)
```

STRING FORMATING

- String formatting expressions are a powerful tool for specifying the format of numbers in a string.
 - Code:

```
Age = 35  
print( "My age is" + Age)      #TypeError  
s1 = " %d this is a string with a number" %123  
print( s1 )
```

STRING FORMATING

Cont'd

- Code:

```
s2 = " %5d this is a string with a number" %123
```

```
s3 = " %-5d this is a string with a number" %123
```

```
s4 = " %.2f" %1.23895426
```

```
s5 = "a string with one integer %d followed by one floating point  
number %.2f" % (123 , 1.23)
```

STRING FORMATING

Cont'd

- String formatting using the `format()` function.

```
s1 = " {} this is a string with a number".format(123)
```

```
s2 = "a string with one integer {} followed by one floating point  
number {}".format(123 , 1.23)
```

```
s3 = "a string with one integer {} followed  
by one floating point number {:.2f}".format(123 , 1.2389)
```

STRING FORMATING

Cont'd

- String formatting using the `format()` function.

```
X = 123456789
```

```
s1 = " {:.d} this is my money".format(X)
```

```
print(s1)
```

```
A, B, C = 5, 6, 7
```

```
print("Hello {} {} {}".format(A, B, C))
```

```
print("Hello {1} {0} {2}".format(A, B, C))
```

```
print("Hello {1:d} {0:0.2f} {2:d}".format(A, B, C))
```

STRING FORMATING

Cont'd

- String formatting using `f` in the beginning of the string.

```
X = 2020
```

```
Name = "ESU"
```

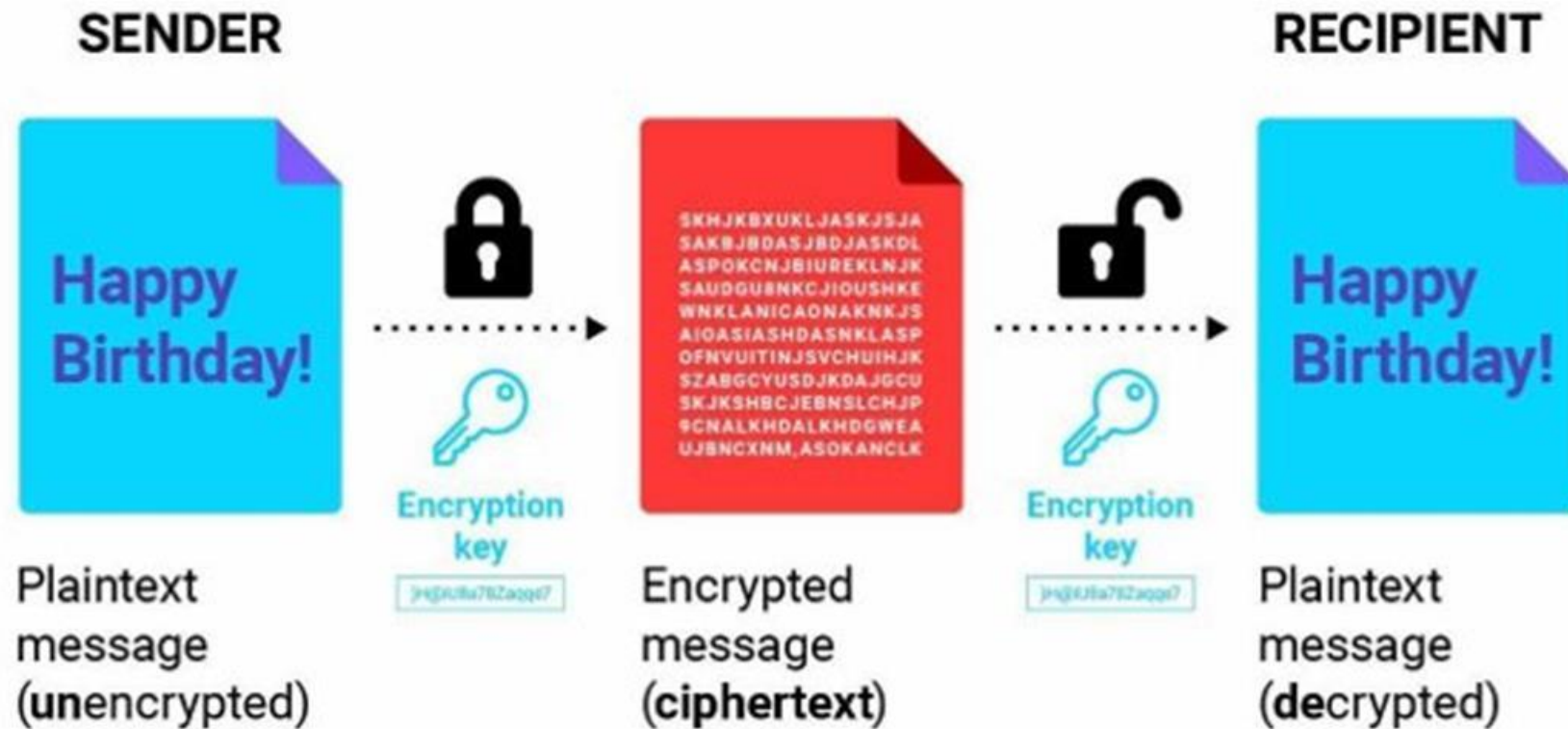
```
print( f" The {Name} university has been established since {X}")
```

CASE STUDY FOR STRING

- It is easy to observe data crossing a network.
- **Data encryption** can be used to protect information transmitted on networks.
- **Cipher text:** the encrypted message generated using one or more keys in the encryption step.
- **Decrypt cipher:** the message is back to its original plain text form in the decryption step.

CASE STUDY FOR STRING

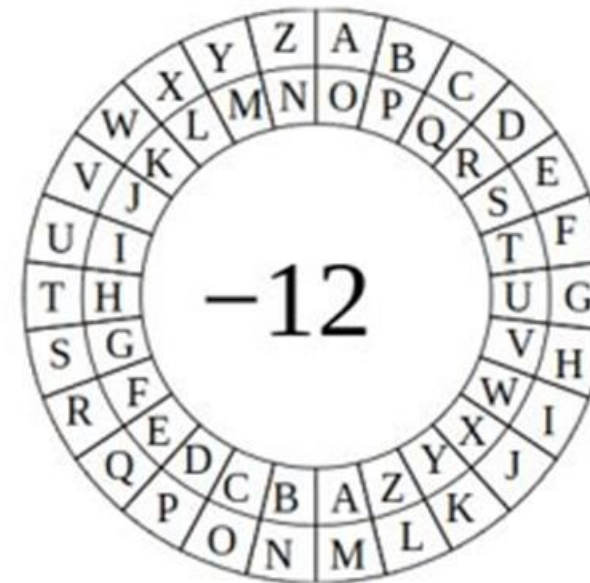
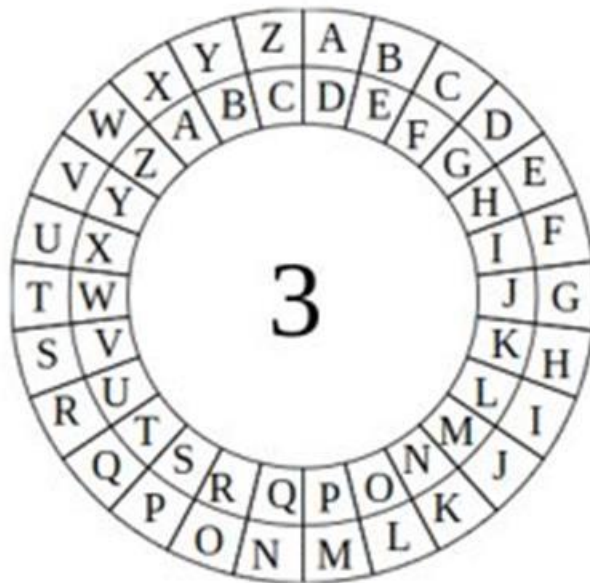
Cont'd



CASE STUDY FOR STRING

Cont'd

- **Caesar cipher** replaces each character in plain text with a character at a given distance away.



Lecture

04

CAESAR ENCRPTION CODE

```
text = "midocean"; k = 5; result = ""
for i in range(len(text)):
    char = text[i]
    oldValue = ord(char)
    cipherValue = oldValue + k
    if cipherValue > ord('z'):
        cipherValue %= 122
        cipherValue += 96
    code = chr(cipherValue)
    result += code
print(result)
```

Lecture

04

CAESAR DECRYPTION CODE

```
text = "rnithjfs";    k = 5;    result = ""
for i in range(len(text)):
    char = text[i]
    oldValue = ord(char)
    cipherValue = oldValue - k
    if cipherValue < ord('a'):
        cipherValue -= 96
        cipherValue += 122
    code = chr(cipherValue)
    result += code
print(result)
```

QUESTIONS FOR ONLINE DISCUSSION

- Write a program that asks the user for a string and displays a string that repeats the input string n times, where n is the length of the input string.
- Write a program to count vowels and consonants in a string.
- Write a program to remove duplicates in a string.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY



Thank You

