Lecture 5 String in Python

IDS 400
Programming for Data Science in Business

Last class

- Loop structure (***while***, ***for*** statement)

Recap examples

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1	2	3	4	5
Row 2	2	4	6	8	10
Row 3	3	6	9	12	15
Row 4	4	8	12	16	20
Row 5	5	10	15	20	25

- Print the square where each element is the product of its row and column number (i*j).

Recap examples

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1	2	3	4	5
Row 2	2	4	6	8	10
Row 3	3	6	9	12	15
Row 4	4	8	12	16	20
Row 5	5	10	15	20	25

- Print the square where each element is the product of its row and column number (i*j).

```
# i controls rows, and j controls columns

# start from the first row
i = 1

# define the maximum height of the square
height = 5

# for each row, we iterate all columns using nested loops
while i <= height:
    j = 1
    line = ''
    while j <= height:
        # For each row, concatenate columns and save it in a string
        line = line + str(i*j) + '\t'
        j += 1  # move to next column
    print(line)
    i += 1  # move to next row
```

```
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
```

Recap examples

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1	2	3	4	5
Row 2	2	4	6	8	10
Row 3	3	6	9	12	15
Row 4	4	8	12	16	20
Row 5	5	10	15	20	25

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1				
Row 2	2	4			
Row 3	3	6	9		
Row 4	4	8	12	16	
Row 5	5	10	15	20	25

- Print the square where each element is the product of its row and column number (i*j).

```
# i controls rows, and j controls columns

# start from the first row
i = 1

# define the maximum height of the square
height = 5

# for each row, we iterate all columns using nested loops
while i <= height:
    j = 1
    line = ''
    while j <= height:
        # For each row, concatenate columns and save it in a string
        line = line + str(i*j) + '\t'
        j += 1  # move to next column
    print(line)
    i += 1  # move to next row
```

```
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
```

- Print the triangle where each element is the product of its row and column number (i*j).

Recap examples

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1	2	3	4	5
Row 2	2	4	6	8	10
Row 3	3	6	9	12	15
Row 4	4	8	12	16	20
Row 5	5	10	15	20	25

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1				
Row 2	2	4			
Row 3	3	6	9		
Row 4	4	8	12	16	
Row 5	5	10	15	20	25

- Print the square where each element is the product of its row and column number (i*j).

```
# i controls rows, and j controls columns

# start from the first row
i = 1

# define the maximum height of the square
height = 5

# for each row, we iterate all columns using nested loops
while i <= height:
    j = 1
    line = ''
    while j <= height:
        # For each row, concatenate columns and save it in a string
        line = line + str(i*j) + '\t'
        j += 1  # move to next column
    print(line)
    i += 1  # move to next row
```

```
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
```

- Print the triangle where each element is the product of its row and column number (i*j).

```
# i controls rows, and j controls columns

# start from the first row
i = 1

# define the maximum height of the square
height = 5

# for each row, we iterate all columns using nested loops
while i <= height:
    j = 1
    line = ''
    while j <= i:
        # For each row, concatenate columns and save it in a string
        line = line + str(i*j) + '\t'
        j += 1  # move to next column
    print(line)
    i += 1  # move to next row
```

```
1
2       4
3       6       9
4       8       12      16
5       10      15      20      25
```

For This Class

- String
- Quiz 1

The string data type

- Text is represented in programs by the string data type.
- A string is a sequence of characters enclosed within quotation “ or apostrophes ‘ .

```
str1 = "Hello"
str2 = 'spam'
print(str1, str2)
```

```
Hello spam
```

```
type(str1)
```

```
str
```

```
type(str2)
```

```
str
```

The string data type

- Getting a string as input

```
firstName= input("Please enter your name:")
```

```
Please enter your name:IDS400
```

```
print("Hello", firstName)
```

```
Hello IDS400
```

The string data type

- We can access the individual characters in a string through *indexing*.
- The positions in a string are numbered from the left, starting with 0.
- The general form is *<string_name>[expr]*, where the value of expr determines which character is selected from the string.

String	H	e	l	l	o		B	o	b	!
Index	0	1	2	3	4	5	6	7	8	9

```
greet = "Hello Bob!"
greet[0]
```

```
'H'
```

```
print(greet[0], greet[2], greet[4])
```

```
H l o
```

```
x = 11
print(greet[x - 2])
```

```
!
```

The string data type

- In a string of n characters, the last character is at position n-1 since we start counting with 0.
- We can index from the right side using negative indexes.
- Indexing returns a string containing a single character from a larger string.

String	H	e	l	l	o		B	o	b	!
Index(+)	0	1	2	3	4	5	6	7	8	9
Index(-)	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

The string data type

- We can also access a contiguous sequence of characters, called *substring*, through a process called **slicing**: `<string>[<start>:<end>]`
- **Start** and **end** should both be *integers*.
- The slice contains the substring *beginning at position start* and runs up to but doesn't include the position end.

The string data type

String	H	e	l	l	o		B	o	b	!
Index	0	1	2	3	4	5	6	7	8	9

```
greet[0:3]
```

```
'Hel'
```

```
greet[:5]
```

```
'Hello'
```

```
greet[:]
```

```
'Hello Bob!'
```

```
greet[5:9]
```

```
' Bob'
```

```
greet[5:]
```

```
' Bob!'
```

String is iterable

```
str1='I love Python'
for i in str1:
    print(i)
```

```
I

l
o
v
e

P
y
t
h
o
n
```

The string data type

Can we put two strings together into a longer string?

- **Concatenation** "glues" two strings together (+)
- **Repetition** builds up a string by multiple concatenations of a string with itself (*)

```
"Spam" + "And" + "Eggs"
```

```
'SpamAndEggs'
```

```
"spam" * 5
```

```
'spamspamspamspamspam'
```

```
3 * "spam"
```

```
'spamspamspam'
```

```
(3 * "spam") + ("eggs" * 5)
```

```
'spamspamspameggseggseggseggseggs'
```

The string data type

Operator	Meaning
+	Concatenation
*	Repetition
<string>[]	Indexing
<string>[:]	Slicing
len(<string>)	Length
for <var> in <string>	Iteration through characters

String methods

- **s.capitalize()** – Copy of s with only the first character capitalized.
- **s.title()** – Copy of s; first character of each word capitalized.
- **s.center(width)** – Center s in a field of given width.
- **s.count(sub)** – Count the number of occurrences of sub in s.
- **s.find(sub)** – Find the first position where sub occurs in s.
- **s.join(list)** – Concatenate list of strings into one large string using s as separator.
- **s.ljust(width)** –Like center, but s is left-justified.

String methods

- **s.capitalize()** – Copy of s with only the first character capitalized.

```
s = "i love Python"

s.capitalize()
'I love python'
```

- **s.title()** – Copy of s; first character of each word capitalized.

```
s.title()
'I Love Python'
```

- **s.center(width)** – Center s in a field of given width.

```
s.center(30)
'         i love Python          '
```

- **s.count(sub)** – Count the number of occurrences of sub in s.

```
s.count('o')
2
```

- **s.find(sub)** – Find the first position where sub occurs in s.

```
s.find('o')
3
```

- **s.join(list)** – Concatenate list of strings into one large string using s as separator.

```
Names = ("John", "Peter", "Vicky")

x = "#".join(Names)

print(x)
```

```
John#Peter#Vicky
```

- **s.ljust(width)** –Like center, but s is left-justified.

```
s.ljust(30)
'i love Python                 '
```

The string data type

- **s.lower()** – Copy of s in all lowercase letters.

```
s.lower()
```
```
'i love python'
```

- **s.lstrip()** – Copy of s with leading white space removed.

```
s1 = ' i love Python'
```
```
s1.lstrip()
```
```
'i love Python'
```

```
s.replace('o', 'x')
```
```
'i lxve Pythxn'
```

- **s.replace(oldsub, newsub)** – Replace occurrences of oldsub in s with newsub.
- **s.rfind(sub)** – Like find, but returns the right-most position.

```
s.rfind('o')
```
```
11
```

- **s.rjust(width)** – Like center, but s is right-justified.

```
s.rjust(30)
```
```
'                 i love Python'
```

- **s.rstrip()** – Copy of s with trailing whitespace removed.

```
s2 = 'i love Python   '
```
```
s2.rstrip()
```
```
'i love Python'
```

- **s.split(sep)** – Split s into a list of substrings using sep as delimiter string.

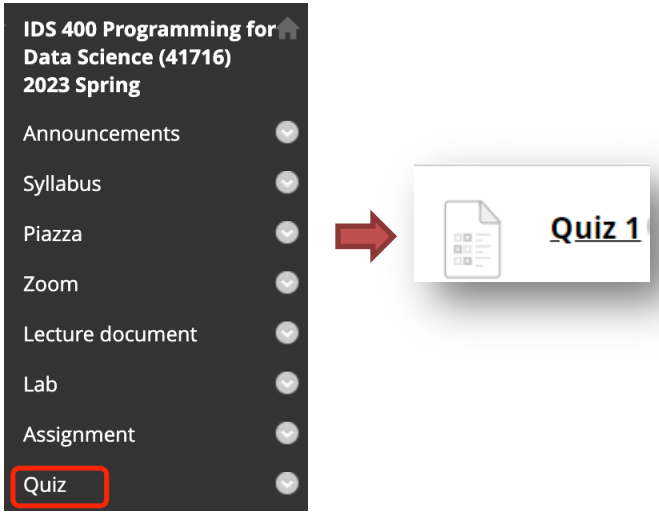
```
s.split(' ')
```
```
['i', 'love', 'Python']
```

- **s.upper()** – Copy of s; all characters converted to uppercase.

```
s.upper()
```
```
'I LOVE PYTHON'
```

Lab *String*

Quiz 1



- **25 multichoice questions**
- **60 minutes**
- You only have ONE attempt to do the quiz.
- This quiz is forced to complete. Once started, this test must be completed in one sitting. Do not leave/refresh/close the test before clicking "*Save and Submit*".
- There is a 60 minutes timer. Warnings appear when half the time, 5 minutes, 1 minute, and 30 seconds remain (The timer does not appear during this test). Test will save and submit automatically when time expires.
- You can change your previous answers before submission.
- Please leave your camera on and mute your microphone during the quiz. If you have a question, please send a private message at chat panel.
- Once you submit, your score will be available immediately.