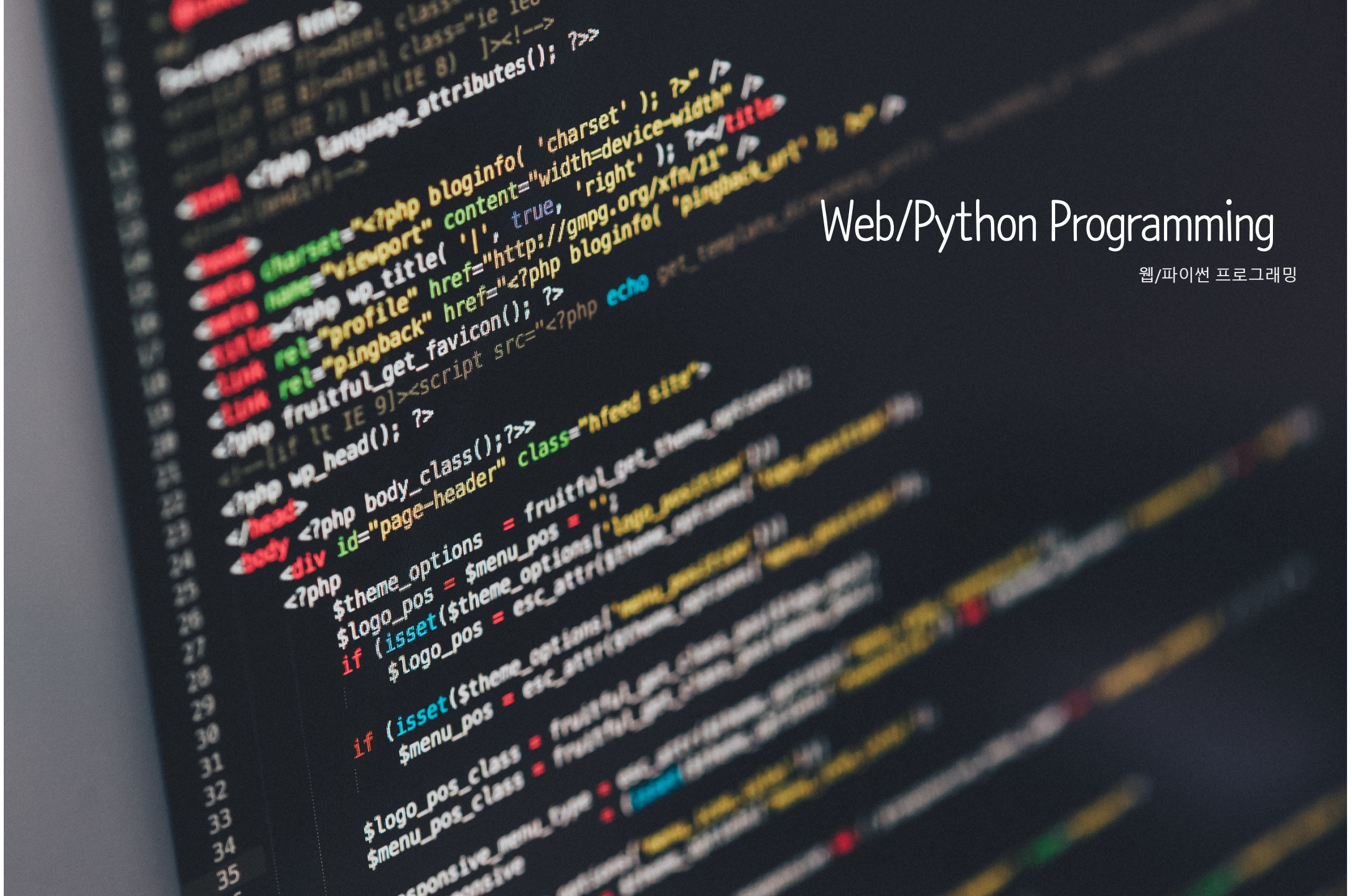

Web/Python Programming

웹/파이썬 프로그래밍

Today

- Working with Text
- Slicing strings
- Special characters in strings
- `print( )`
- `input( )`
- Covers Chapter 4 in your textbook

String

- In Python, text is represented as a string.
- String is a type. (`str`)
- String is a sequence of characters.
- Characters include letters, digits, and symbols.
- Characters include Latin alphabet, 한글, chemical symbols, musical symbols, and much more.

How to define a string?

- Single quotes
- Double quotes

- The opening and closing quotes must match.

```
>>> 'Aristotle'
'Aristotle'
>>> "Issac Newton"
'Issac Newton'
>>> 'Charles Darwin"
SyntaxError: EOL while scanning string literal
>>>
```

Empty string

- ‘’
- “”
- It contains no character.
- It’s not a blank. It’s an empty string.

Cf. How long can a string be?

- Limited only by computer memory.

Operations on strings

- Python built-in functions for string
  - len( ) : returns the length of a string
  - + : concatenates two strings
  - * : repeats and concatenates strings
  - int( ) : converts a string of numbers to integer type
  - float( ) : converts a string of numbers to floating-point type

len( ) and +

- len() returns the length of the string
- + concatenates two strings

```
>>> len('Albert')
6
>>> len('Einstein')
8
>>> len('Albert Einstein')
15
>>> a = 'Albert' + 'Einstein'
>>> a
'AlbertEinstein'
>>> type(a)
<class 'str'>
>>> len(a)
14
```

```
>>> len('')
0
>>> len("")
0
>>> 'Albert' + ''
'Albert'
>>> '' + 'Albert'
'Albert'
```

Type error when using +

- Python can add numbers using +
- Python can concatenate strings using +

```
>>> 'Albert' + 3
Traceback (most recent call last):
  File "<pyshell#51>", line 1, in <module>
    'Albert' + 3
TypeError: Can't convert 'int' object to str implicitly
```

```
>>> 9.0 + 'Albert'
Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    9.0 + 'Albert'
TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

```
>>> '9.0' + 'Albert'
'9.0Albert'
>>>
>>> 'Four score and ' + str(7) + ' years ago'
'Four score and 7 years ago'
```

int( ) and float( ) for strings

- Typecast

```
>>> 0
0
>>> '0'
'0'
>>> int('0')
0
>>> int("11")
11
>>> int('-324')
-324
>>> float('-324')
-324.0
>>> float("56.34")
56.34
```

```
>>> int('a')
Traceback (most recent call last):
  File "<pyshell#70>", line 1, in <module>
    int('a')
ValueError: invalid literal for int() with base 10: 'a'
>>>
>>> float('hello')
Traceback (most recent call last):
  File "<pyshell#72>", line 1, in <module>
    float('hello')
ValueError: could not convert string to float: 'hello'
```

* for strings

- Repeat and concatenate
- Multiplied with a number less than or equal to zero yields the empty string

```
>>> 'AT' * 5
'ATATATATAT'
>>> 4 * '-'
'----'
>>>
>>> 'GC' * 0
''
>>> 'AT' * -3
''
```

Assign a string to a variable

- Strings are values, so you can assign a string to a variable

```
>>> sequence = 'ATTGTCC'
>>> len(sequence)
7
>>> new_sequence = sequence + 'GGCCTCC'
>>> new_sequence
'ATTGTCCGGCCTCC'
>>> len(new_sequence)
14
>>> new_sequence * 2
'ATTGTCCGGCCTCCATTGTCCGGCCTCC'
>>>
```

Special characters in strings

- Single quote / double quote inside a string

```
>>> 'that's not going to work'
SyntaxError: invalid syntax
>>>
>>> "that's better"
"that's better"
>>>
>>> 'She said, "That is better."'
'She said, "That is better."'
```

- Both?

```
>>> 'She said, "That' + "'" + 's hard to read."'
'She said, "That₩'s hard to read."'
```

Escape sequence

- Backslash is called an escape character.
- \ + a single quote : escape sequence.
- “escaping from Python’s usual syntax rules for a moment”

- An escape sequence is one character (not two!)

```
>>> len('₩'')
1
>>> len('it₩'s')
4
```

Escape Sequence	Description
\'	Single quote
\"	Double quote
\\	Backslash
\t	Tab
\n	Newline
\r	Carriage return

Table 4—Escape Sequences

When do we use escape sequence?

- To create a multiline string
- To print information

```
>>> a = 'one
SyntaxError: EOL while scanning string literal
>>> a = '''one
two
three'''
>>> a
'one₩ntwo₩nthree'
>>> print(a)
one
two
three
>>> len(a)
13
```

```
>>> b = "one₩ntwo₩nthree"
>>> b
'one₩ntwo₩nthree'
>>> print(b)
one
two
three
```

Escape Sequence	Description
\'	Single quote
\"	Double quote
\\	Backslash
\t	Tab
\n	Newline
\r	Carriage return

Table 4—Escape Sequences

print( )

```
>>> a = 'one'
>>> a
'one'
>>> print(a)
one
>>> b = 'one₩ntwo₩nthree'
>>> b
'one₩ntwo₩nthree'
>>> print(b)
one
two
three
>>> c = 'one₩ttwo₩nthree₩tfour'
>>> c
'one₩ttwo₩nthree₩tfour'
>>> print(c)
one     two
three   four
```

print( )

- It takes a comma-separated list of values to print and prints the values with a single space between them and a newline after the last value

```
>>> print(1,2,3)
1 2 3
>>>
```

- With no arguments, it ends the current line, advancing to the next one

```
>>> print()

>>>
```

- It can print values of any type, and it can even print values of different types in the same function call

```
>>> print(1, 'two', 'three' , 4.0)
1 two three 4.0
>>>
```

```
>>> radius = 5
>>> print("The diameter is", radius * 2, "cm.")
The diameter is 10 cm.
```

Default setting of print()

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='₩n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
>>> print('a','b','c')
a b c
>>> print('a','b','c',sep='-')
a-b-c
>>> print('a','b','c',sep='!!')
a!!b!!c
>>> print('a','b','c',sep='-',end='r')
a-b-cr
```

Example

```
def convert_to_celsius(fahrenheit):
    """ (number) -> float
    Return the number of Celsius degrees equivalent to fahrenheit degrees.
    >>> convert_to_celsius(75)
    23.88888888888889
    """
    return (fahrenheit - 32.0) * 5.0 / 9.0
print('80, 78.8, and 10.4 degrees Fahrenheit are equal to ', end='')
print(convert_to_celsius(80), end=', \n')
print(convert_to_celsius(78.8), end=', and ')
print(convert_to_celsius(10.4), end=' Celsius.\n')
```

80, 78.8, and 10.4 degrees Fahrenheit are equal to 26.666666666666668, 26.0, and -12.0 Celsius.

input( )

```
>>> species = input()
Homo sapiens
>>> species
'Homo sapiens'
>>> population = input()
6973738433
>>> population
'6973738433'
>>> type(population)
<class 'str'>
```

```
>>> population = input()
6973738433
>>> population
'6973738433'
>>> population = int(population)
>>> population
6973738433
>>> population = population + 1
>>> population
6973738434
```

```
>>> species = input("Please enter a species: ")
Please enter a species: Python curtus
>>> print(species)
Python curtus
```

```
>>> population = int(input())
6973738433
>>> population = population + 1
6973738434
```

Slicing string

```
>>> a = “You only live once.”
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Y	o	u		o	n	l	y		l	i	v	e		o	n	c	e	.

```
>>> a[2]
‘u’
>>> b = a[4]+a[5]+a[6]+a[7]
>>> b
‘only’
>>> c = a[9:12]
>>> c
‘liv’
```

```
>>> d = a[10:]
>>> d
‘ive once.’
>>> e = a[:7]
>>> e
‘You onl’
```

Slicing string

```
>>> a = “You only live once.”
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Y	o	u		o	n	l	y		l	i	v	e		o	n	c	e	.
-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> a[-1]
‘.’
>>> a[-0]
‘Y’
>>> b = a[12:-2]
>>> b
‘e onc’
>>> c = a[12:-13]
>>> c
‘’
```

```
>>> date = ‘20170331Rainy’
>>> year = date[:4]
>>> month = date[4:6]
>>> day = date[6:8]
>>> weather = date[8:]
```

Summary

- Python uses type `str` to represent text as sequences of characters.
- Strings are created by placing pairs of single or double quotes around the text. Multiline strings can be created using matching pairs of triple quotes.
- Special characters like newline and tab are represented using escape sequences that begin with a backslash.
- Values can be printed using built-in function `print`, and input can be provided by the user using built-in function `input` .
- Exercises in Ch.4.7 in your textbook