String formatting

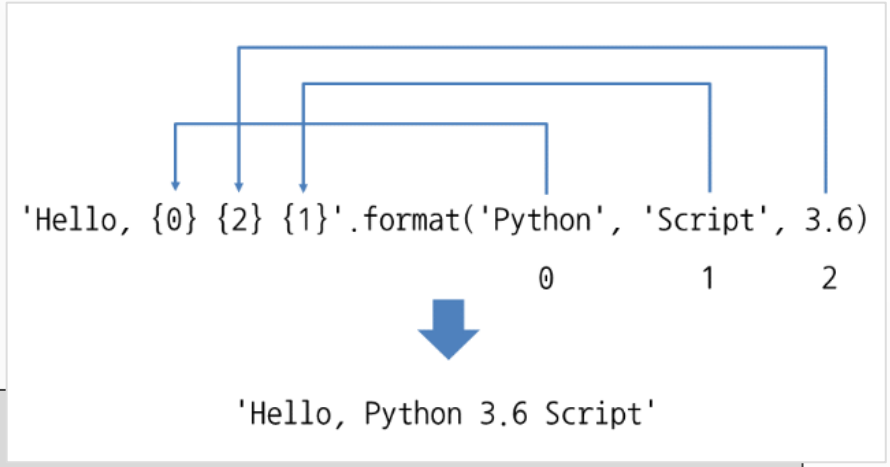
```
>>> Name = input(‘Your name’)
Min-ho Lee
>>> City =input('Where are you staying?’)
Nulsultan

'Hello, my name is Min-Ho Lee, and I am staying in Nulsultan’
```

String formatting

```
print('Hello, my name is Min-Ho Lee, and I am staying in Nulsultan')

Name = 'Min-Ho Lee'  ;  City = 'Nulsultan‘

# arrange (append) individual strings
print('Hello, my name is ' + Name  + ', and I am staying in ' + City )

# format function
print('Hello, my name is {}, and I am staying in {}'.format(Name, City ))

# format specifier with % symbol
print('Hello, my name is %s, and I am staying in %s' % (Name, City ))

# f-string
Print(f'Hello, my name is {Name}, and I am staying in {City}')
```

String formatting

- Formatting strings
  - **The format() method allows you to format selected parts of a string.**
  - **You can add parameters inside the curly brackets to specify how to convert the value**




```
>>> '{0} + {1} = {2}'.format(1, 2, 1+2)
1 + 2 = 3
```

- {number} is replaced with corresponding positional argument

```
>>> '{} + {} = {}'.format(5, 4, 5+4)
5 + 4 = 9
```

- If arguments used in sequence, numbers can be skipped

```
>>> 'Hello, {0} {2} {1}'.format('Python', 'Script', 3.6)
'Hello, Python 3.6 Script'
```

String formatting

- Formatting strings
  - Index Numbers, Named Indexes

```
>>> '{0} {0} {1} {1}'.format('Python', 'Script')
'Python Python Script Script’
>>> 'Hello, {0} {1}'.format('Python',  3.6)
'Hello, Python 3.6 Script’

>>> 'Hello, {language} {version}'.format(language='Python', version=3.6)
'Hello, Python 3.6‘
>>> 'Hello, {language} {version}'.format(version=3.6, language='Python')
'Hello, Python 3.6‘
```

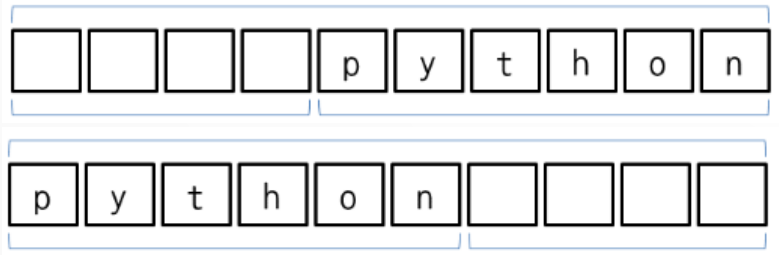
```
>>> language = 'Python'
>>> version = 3.6
>>> f'Hello, {language} {version}'
'Hello, Python 3.6'
```

Format specifier

- ‘%s’ % ‘string’ | ‘%d’ % ‘decimal integer’ | ‘%f’ % ‘fixed point’

```
>>> 'I am %s.' % 'james'
'I am james.'

>>> name = 'maria'
>>> 'I am %s.' % name
'I am maria.‘

>>> 'I am %d years old.' % 20
'I am 20 years old.'

>>> '%f' % 2.3
'2.300000'

>>> '%.2f' % 2.3
'2.30'

>>> '%.3f' % 2.3
'2.300'
```

```
>>> '%10s' % 'python'
'    python'
>>> '%-10s' % 'python'
'python    '
```




```
>>> '%10d' % 150
'       150'
>>> '%10d' % 15000
'     15000'

>>> '%10.2f' % 2.3
'      2.30'
>>> '%10.2f' % 2000.3
'   2000.30'
```

Format specifier

- ‘%s %d’ % (‘string’, ‘decimal integer’)

```
>>> 'Today is %d %s.' % (3, 'April')
'Today is 3 April.'

>>> 'Today is %d%s.' % (3, 'April')
'Today is 3April.'
```

https://docs.python.org/3/library/string.html

String-type variable

- Individual variables have string value.
- Basically any keystroke but there are some exceptions and special rules. More later.

```
>>> str1 = ‘ABC’
>>> str2 = ‘abc’
>>> str3 = ‘A B C’
>>> Digits = ‘123456789’
>>> Punctuation = ‘!:?,;’
>>> Special = ‘@#$%^&*()_’
```

Backslash sequence

- In **Python** strings, the backslash "\" is a special character, also called the "escape" character.

\	Backslash and newline ignored
\\	Print backslash
\’	Print Single quote
\"	Print Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
\ooo	Character with octal value *ooo*
\xhh	Character with hex value *hh*

https://www.quackit.com/python/reference/python_3_escape_sequences.cfm

Caution in string data type

- '\' indicates escaping

```
>>> print('First line. Second line.')
First line. Second line.
>>> print('First line. \nSecond line.')
First line.
Second line.
```

- Raw strings with r prefix -> no interpretation of "special characters"

```
>>> print('C:\computer\name\minho')


>>> print(r'C:\computer\name\minho')
```

Caution in string data type

- '\' indicates escaping

```
>>> print('First line. Second line.')
First line. Second line.
>>> print('First line. \nSecond line.')
First line.
Second line.
```

- Raw strings with r prefix -> no interpretation of "special characters"

```
>>> print('C:\computer\name\minho')
C:\computer
ame\minho
>>> print(r'C:\computer\name\minho')
```

Caution in string data type

- '\' indicates escaping

```
>>> print('First line. Second line.')
First line. Second line.
>>> print('First line. \nSecond line.')
First line.
Second line.
```

- Raw strings with r prefix -> no interpretation of "special characters"

```
>>> print('C:\computer\name\minho')
C:\computer
ame\minho
>>> print(r'C:\computer\name\minho')
C:\computer\name\minho
```

New line

- How to print quote, double quote, and backslash ?

Isn’t it true? *Expected output*

```
>>> print(‘isn’t it true?’)
SyntaxError: invalid syntax
>>> print(‘isn\’t it true?’)
```

“We are learning python” *Expected output*

```
>>> print(‘\”We are learning python\”’)
```

Single line

- Single statement per line
- Lines can be extended by ‘\’

```
>>> 1 + 2 + 3
6
>>> 1 + 2 + 3 \
+ 4 + 5 \
+6 + 7
28
```

- Parentheses can be also used for breaking expressions

```
>>> (1 + 2 + 3
+ 4)
10
```

Semicolon ‘;’

- It is used to separate commands on a single line.
- However, it is not considered “Pythonic.”.

```
>>> a = 3
>>> b = 4
>>> c = 5
```

```
>>> a = 3; b = 4; c = 5
```