LECTURE 6

DOHYUNG KIM

WHAT IS DISCUSSED IN THE LAST CLASS

- Loops in python

TODAY, WE WILL LEARN ABOUT

- Strings

STRING LITERALS

- You may use quotes for string literals

```
str1 = "Welcome to the python class"
str2 = 'Welcome to the python class'

print(str1)
print(str2)
print(str1==str2)
```

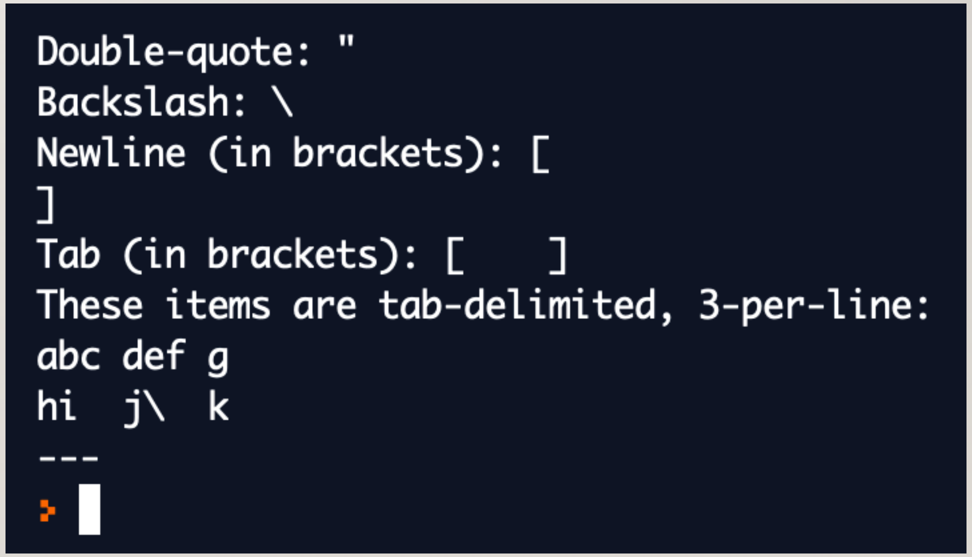
Single quotes and double quotes have the same meaning

- What about triple-quotes?

```
str3 = """Welcome to
the python class"""
str4 = "Welcome to\n the python class"
print(str3)
print()
print(str4)
print()
print(str3==str4)
```

ESCAPE SEQUENCES

```python
print("Double-quote: \"")
print("Backslash: \\")
print("Newline (in brackets): [\n]")
print("Tab (in brackets): [\t]")

print("These items are tab-delimited, 3-per-line:")
print("abc\tdef\tg\nhi\tj\\\tk\n---")
```

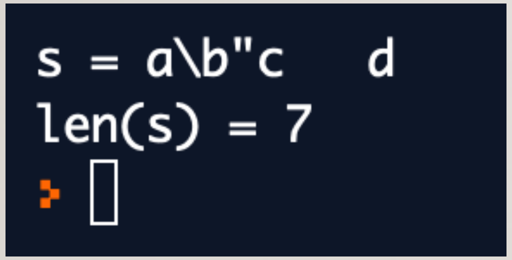


```
Double-quote: "
Backslash: \
Newline (in brackets): [
]
Tab (in brackets): [    ]
These items are tab-delimited, 3-per-line:
abc def g
hi  j\  k
---
> 
```

ESCAPE SEQUENCES

- An escape sequence produces a single character

```
s = "a\\b\"c\td"
print("s =", s)
print("len(s) =", len(s))
```



```
s = a\b"c	d
len(s) = 7
>
```

EXAMPLES OF STRING CONSTANTS

- Defined in string module

```python
import string
print(string.ascii_letters)
print(string.ascii_lowercase)
print("-----------")
print(string.ascii_uppercase)
print(string.digits)
print("-----------")
print(string.punctuation)
print(string.printable)
print("-----------")
print(string.whitespace)
```

EXAMPLES OF STRING OPERATORS

- **+** : concatenation
- ***** : repetition

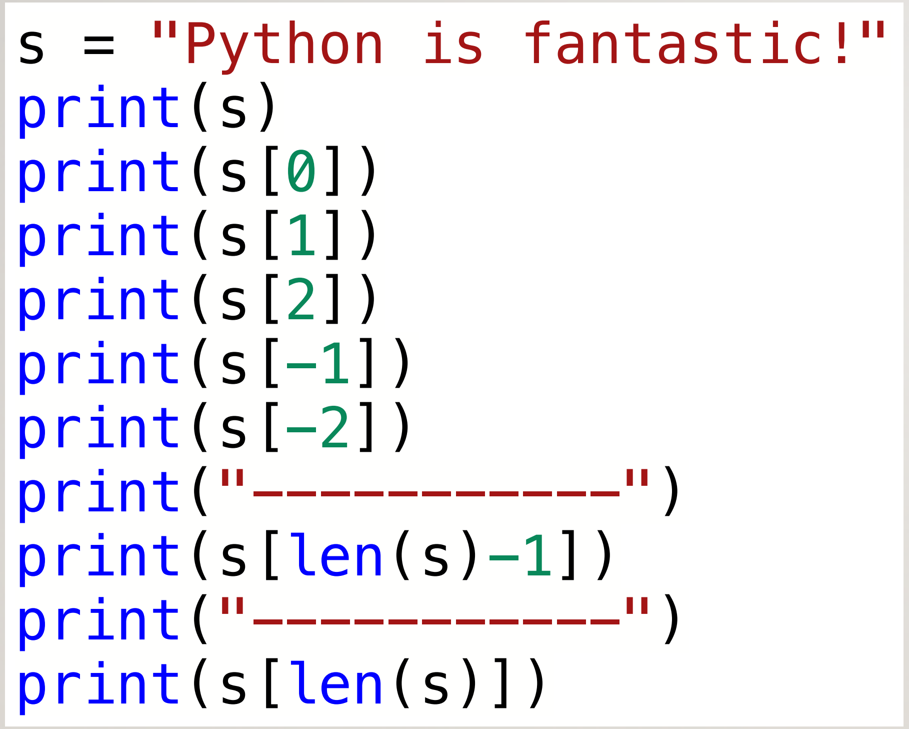
```
print("abc" + "def")
print("abc" * 3)
print("abc" + 3)
```

- **in** : checking occurrence

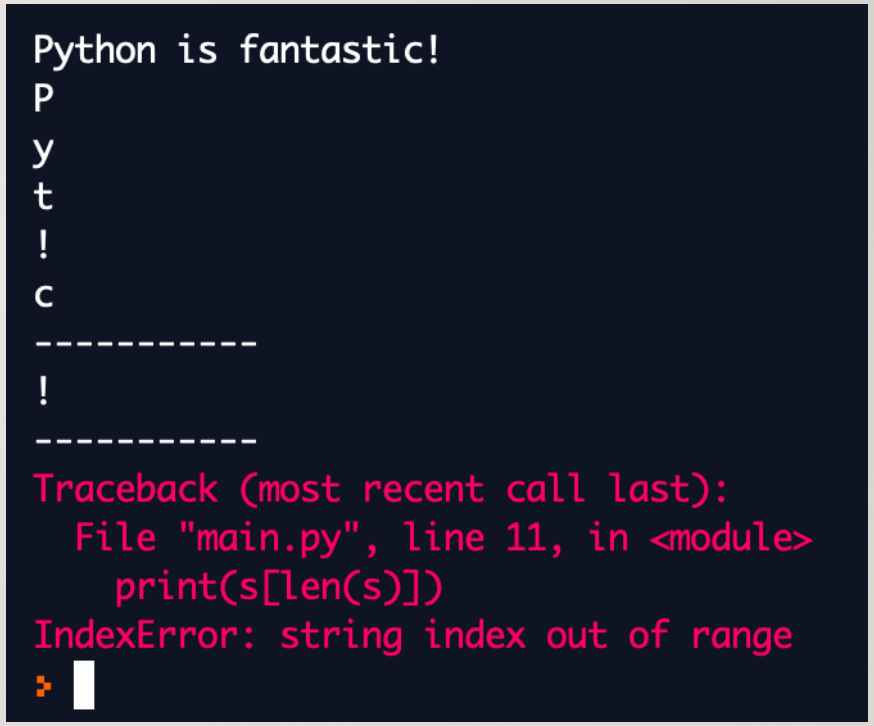
```
print("mart" in "smart")
print("mile" in "smiles!")
print("Yes" in "yes!")
print("" in "No way!")
```

EXAMPLES OF STRING OPERATORS

- String indexing



```python
s = "Python is fantastic!"
print(s)
print(s[0])
print(s[1])
print(s[2])
print(s[-1])
print(s[-2])
print("-----------")
print(s[len(s)-1])
print("-----------")
print(s[len(s)])
```

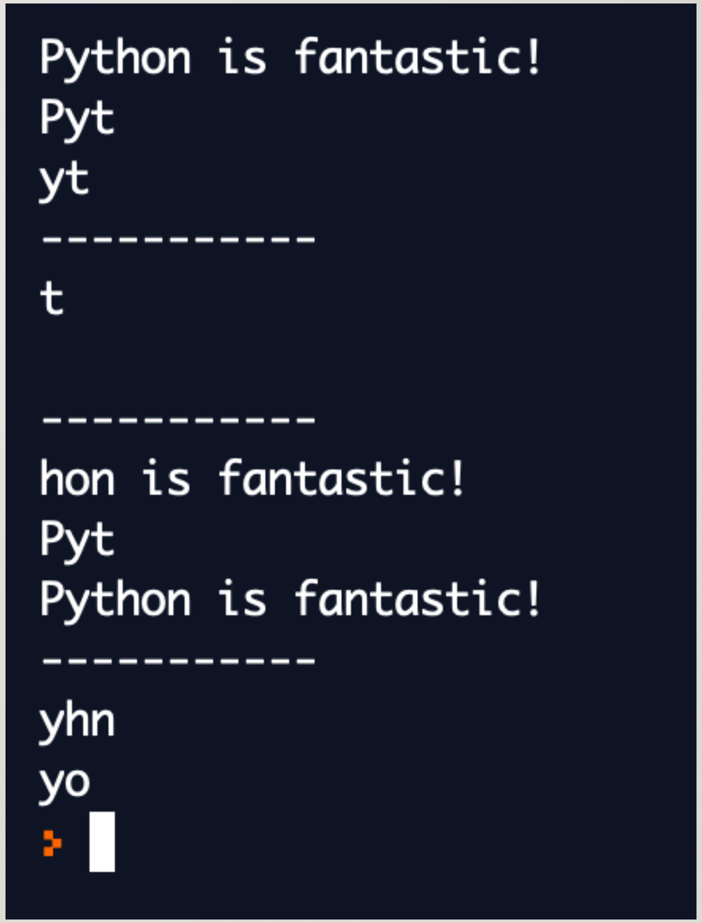


```
Python is fantastic!
P
y
t
!
c
-----------
!
-----------
Traceback (most recent call last):
  File "main.py", line 11, in <module>
    print(s[len(s)])
IndexError: string index out of range
> 
```

EXAMPLES OF STRING OPERATORS

- String slicing

```python
s = "Python is fantastic!"
print(s)
print(s[0:3])
print(s[1:3])
print("-----------")
print(s[2:3])
print(s[3:3])
print("-----------")
print(s[3:])
print(s[:3])
print(s[:])
print("-----------")
print(s[1:7:2])
print(s[1:7:3])
```



```
Python is fantastic!
Pyt
yt
-----------
t

-----------
hon is fantastic!
Pyt
Python is fantastic!
-----------
yhn
yo
>
```

PRACTICE

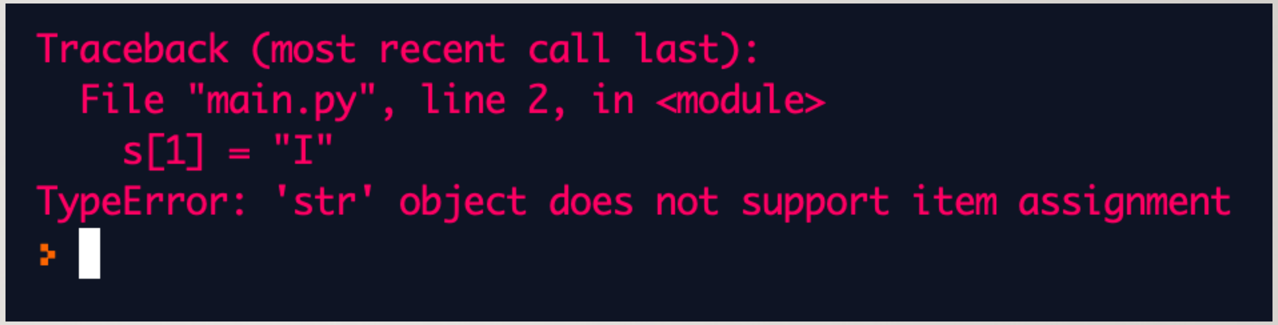
- Reverse the input string

```python
def reverseString1(str):
  res = ""
  for i in range(len(str)-1, -1, -1):
    res += str[i]
  print(res)

def reverseString2(str):
  print(str[::-1])

s = input("Input the string:")
reverseString1(s)
reverseString2(s)
```

WAIT A SECOND!

- Strings are **immutable** sequences

```python
s = "python"
s[1] = "I"
print(s)
```



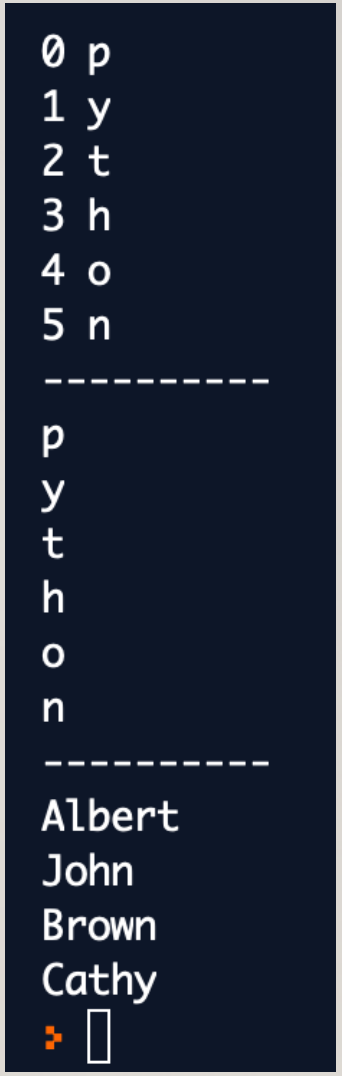
```
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    s[1] = "I"
TypeError: 'str' object does not support item assignment
>
```

- If you want to change some characters, you must create a new string

```python
s = "python"
s = s[:1] + "I" + s[2:]
print(s)
```

LOOPING OVER STRINGS

```python
s = "python"
for i in range(len(s)):
  print(i, s[i])

print("----------")

for c in s:
  print(c)

print("----------")

names = "Albert,John,Brown,Cathy"
for name in names.split(","):
  print(name)
```



```
0 p
1 y
2 t
3 h
4 o
5 n
----------
p
y
t
h
o
n
----------
Albert
John
Brown
Cathy
```

LOOPING OVER STRINGS

```
str = """\
KNU = Kangwon National University
KNU = Kyungpook National University
SNU = Seoul National University
PNU = Pusan National University
CNU = Chungnam National University
"""
for line in str.splitlines():
  if (line.startswith("KNU")):
      print(line)
```

PRACTICE

- Palindrome
  - a word, phrase, or sequence that reads the same backward as forward, e.g., *madam*

- Write a function to check whether a word is palindrome or not

BUILT-IN FUNCTION FOR STRINGS

- str() and len()

```
name = input("Enter your name: ")
print("Hi, " + name + ". Your name has " + str(len(name)) + " letters!")
```

- chr() and ord() : conversion between unicode number and character

```
print(ord("A"))
print(chr(65))
print(chr(ord(“A")+1))
```

BUILT-IN FUNCTION FOR STRINGS

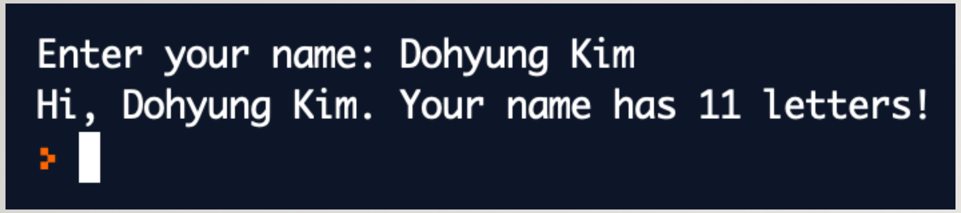
- eval()

```
def func():
  print("Code inside a function")

s = "(3**2 + 4**2)**0.5"
print(eval(s))

s = "func()"
print(eval(s))
```

PRACTICE

- Wait! How can we count Alphabets only?

```python
name = input("Enter your name: ")
print("Hi, " + name + ". Your name has " + str(len(name)) + " letters!")
```



```
Enter your name: Dohyung Kim
Hi, Dohyung Kim. Your name has 11 letters!
```

- Try to solve after discussing string methods

STRING METHODS

- Character types: isalnum(), isalpha(), isdigit(), islower(), isspace(), isupper()

```
def p(test):
  print("True     " if test else "False    ", end="")

def printRow(s):
  print(" " + s + "  ", end="")
  p(s.isalnum())
  p(s.isalpha())
  p(s.isdigit())
  p(s.islower())
  p(s.isspace())
  p(s.isupper())
  print()

def printTable():
  print("  s   isalnum  isalpha  isdigit  islower  isspace  isupper")
  for s in "ABCD,ABcd,abcd,ab12,1234,    ,AB?!".split(","):
    printRow(s)
```

STRING METHODS

- String edits: lower(), upper(), replace(), strip()

```python
print("This is nice. Yes!".lower())
print("So is this? Sure!!".upper())
print("   Strip removes leading or trailing whitespace only   ".strip())
print("   Strip removes leading or trailing whitespace only   ".lstrip())
print("   Strip removes leading or trailing whitespace only   ".rstrip())
print("This is nice.  Really nice.".replace("nice", "sweet"))
print("This is nice.  Really nice.".replace("nice", "sweet", 1))

print("----------------")
s = "This is so so fun!"
t = s.replace("so ", "")
print("original:", s)
print("replaced:", t)
```

STRING METHODS

- Substring search: count(), startswith(), endswith(), find(), index()

```
print("Hickory Dickory Dock".count("kor"))
print("HicKory DickoRy Dock".count("kor"))
print("--------")
print("Kangwon National University".startswith("Kan"))
print("Kangwon National University".startswith("Seo"))
print("--------")
print("Kangwon National University".endswith("ty"))
print("Kangwon National University".endswith("city"))
print("--------")
print("Kangwon National University".find("National"))
print("Kangwon National University".find("national"))
print("--------")
print("Kangwon National University".index("National"))
print("Kangwon National University".index("rational"))
```

PRACTICE

- Revisit the problem that only counters alphabet characters

```
def countAlphabet(x):
  res = 0
  for c in x:
    if(c.isalpha()):
      res += 1
  return res

def lenWithoutBlank(x):
  return len(x) - x.count(" ")

name = input("Enter your name: ")
print("Hi, " + name + ". Your name has " + str(len(name)) + " letters!")
print("Hi, " + name + ". Your name has " + str(countAlphabet(name)) + " letters!")
print("Hi, " + name + ". Your name has " + str(lenWithoutBlank(name)) + " letters!")
```

STRING FORMATTING

- A string with %s

```
breed = "Superman"
print("Did you see %s?" % breed)
```

- A integer with %d

```
age = 20
print("I am %d years old" % age)
```

- A float number with %f

```
grade = 72.5
print("The average score is %d in this semester" % grade)
print("The average score is %f in this semester" % grade)
```

STRING FORMATTING

- A string with %s

```
breed = "Superman"
print("Did you see %s?" % breed)
```

- A integer with %d

```
age = 20
print("I am %d years old" % age)
```

- A float number with %f, %.[precision]f

```
grade = 72.557
print("The average score is %d in this semester" % grade)
print("The average score is %f in this semester" % grade)
print("The average score is %.2f in this semester" % grade)
```

STRING FORMATTING

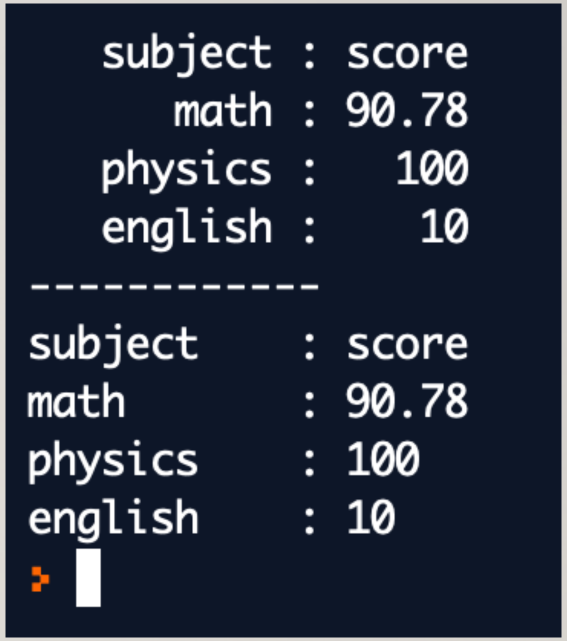
- Multiple values

```
name = "Albert"
math = 100
physics = 100
english = 99.9
print("""\
The %s's scores for math, physic, and English are
%d, %d, and %2f, respectively."""
% (name, math, physics, english))
```

STRING FORMATTING

- Alignment with %[width]

```python
math = 90.78
physics = 100
english = 10
print("%10s : %5s" % ("subject", "score"))
print("%10s : %5.2f" % ("math", math))
print("%10s : %5d" % ("physics", physics))
print("%10s : %5d" % ("english", english))

print("------------")

print("%-10s : %-5s" % ("subject", "score"))
print("%-10s : %-5.2f" % ("math", math))
print("%-10s : %-5d" % ("physics", physics))
print("%-10s : %-5d" % ("english", english))
```



```
   subject : score
      math : 90.78
   physics :   100
   english :    10
------------
subject    : score
math       : 90.78
physics    : 100
english    : 10
```

QUESTION?