CISS450: Artificial Intelligence
Lecture 5: Strings

Yihsiang Liow

Literals

- Examples:
  - `"Who Killed Marilyn Monroe?"`
  - `'"Is that The Mountain?" asked Bilbo'`
  - `"""Adam said, "Madam, I'm Adam." """`
  - `'''Adam said, "Madam, I'm Adam."'''`
- Enclose characters with
  - `' or "`
  - `''' or """`
- Try this:

```
s = ""What iss he, my preciouss?""
t = """He said, "That's it!""""
```

Literals

- ''' or """ version preserves multi-line format
- Try this:

```
print("""
  What has roots as nobody sees,
  Is taller than trees
    Up, up it goes,
    And yet never grows?
""")
```

Literals

- Special characters
  - `\n` = new line
  - `\t` = tab
  - `\'` = character '
  - `\"` = character "
  - Check documentation for others
- Example

```
print("one\ttwo\nthree\tfour")
print("\"Nonsense!,\" said Thorin ...")
print('"Nonsense!," said Thorin ...')
```

Characters

- In C/C++, you have both characters and strings
- In Python, there is no character type. A character is just a string of length one.

Bracket Operator

- Try the following:

```
s = '"Nonsense!," said Thorin ...'
print(s[0], s[4], s[-1], s[-5])
print(s[3:5])
print(s[:5])
print(s[3:])
print(s[3:-4:2])
print(s[::-2])
print(s[3:2])
print(s[100])
```

- Substrings are also called slices in Python. Can you slice out `"Thorin"`?

Bracket Operator

- Note that if you want to make a copy of a string you can do:

```
x = "Joe Doe"
y = "Joe Doe"
```

or

```
x = "John Doe"
y = x[:]
```

Bracket Operator

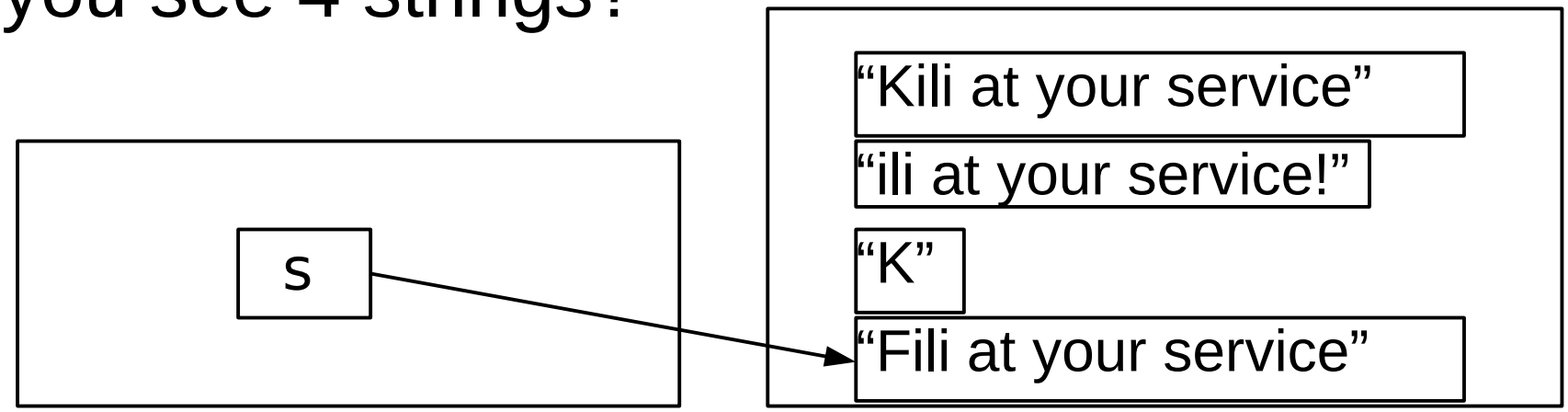
- Try this: We want to change Kili to Fili

```
s = '"Kili at your service!"'
s[1] = 'F'
```

- Won't work: ***Strings are immutable***
- Solution: Slice it up

```
s = s[:1] + 'F' + s[2:]
```

- Do you see 4 strings?



Length

- Try this:

```
print(len("How long is this string?"))
print(len(""))
print(len("\t"))
print(type(len("")))
```

Comparison Operators

- Operators: ==, !=, <, <=, >, >=
- Try this:

```
s0 = "abc"
t0 = "abc"
b = s0==t0
print("1.", b, id(s0), id(t0))
s1 = 100 * "1"
t1 = 100 * "1"
b = s1==t1
print("2.", b, id(s1), id(t1))
print("3.", s0==s1)
```

Comparison Operators

- != means “not equal”. Try:

```
print("gold" != "gold", "gold" != "silver")
print(!("gold"=="gold)",!("gold"=="silver"))
```

- < and > compares by lexicographical or dictionary order. Try this:

```
print("abc" < "abd")
print("a" < "abd")
print("f" < "abc")
```

- <= means < or ==
- >= means > or ==

Boolean

- Strings can be type converted to booleans using `bool()` function:
  - Empty string `""` is `False`
  - Non-empty strings are `True`
- The type coercion is automatic if a string is where a boolean value is expected. For instance later we'll talk about the if statement. The following is valid Python code:

```
if "Hello": print "World!"
```

Does it print `"World!"`?

and, or, not

- Try this:

```
s = "123"
t = "abc"
print("1.", s and t)
print("2.", "" and t)
print("3.", s and "")
print("4.", s or t)
print("5.", "" or t)
print("6.", s or "")
print("7.", not s)
print("8.", not "")
```

`and`, `or`, `not`

- So
  - `s and t` = `""` if either `s` or `t` is `""`
  - `s and t` = `t` if neither `s` nor `t` is `""`

  and
  - `s or t` = `s` if either `s` is not `""`
  - `s or t` = `t` if `s` is `""`

  and
  - `!s` is `True` if `s` is `""`
  - `!s` is `False` if `s` is not `""`

in

- Try this:

```
a = "Bill Gates"
print("Bill" in a, "William" in a)
```

find

- Try this:

```
s = "water, water everywhere"
t = "water"
print(s.find(t))
print(s.find("soda"))
print(s.find(t, 1))
print(s.find(t, 1, 5))
```

- Look for a method that will search from the last character instead of from the first

Useful Functions/Methods

- Try this:

```
s = "  good grief   "
print("1.", s.capitalize())
print("2.", s.lstrip())
print("3.", s.rstrip())
print("4.", s.strip())
print("5.", s.replace(" ", ""))
```

Useful Strings

- In the string module there are some useful strings already defined. If you want to use them, you need to import the string module. Try this:

```
import string
print(string.letters, string.lowercase)
s = input("Enter a single character:")
print("Letter entered:", s in string.letters)
print("Big letter entered:", s in string.lowercase)
print("Small letter entered:", s in string.uppercase)
print("Digit was entered:", s in string.digits)
print("Punctuation was entered:", s in string.punctuation)
```

Type Conversion

- Try this:

```
s = "123"
t = int(s)
print(s + s, t + t)
```

- Instead of `"123"` try the above with
  - `"1.23"`
  - `" "`
  - `"  1  "`
  - `"  1  1  "`
- Experiment with `float(s)` instead of `int(s)`

split and join

- Try this:

```
words = ["Cat", "in", "the", "hat"]
print("1.", "...".join(words))
print("2.", "".join(words))
title = " ".join(words)
print(title.split())
print(title.split("t"))
print(title.split("t", 1))
```

Python Documentation

- Do you know where to look in order to find out more about strings?

ASCII code

- You can get the ASCII code of a character using the ord function:

```
s = "abc"
for c in s:
    print(c, ord(c))
```

- You can get the ASCII character from an integer value in the range of 0..127:

```
x = chr(97)
print(x, type(x), ord(x))
for i in range(98, 110):
    x += chr(i)
    print(x)
```

Unicode strings

- Look at http://www.unicode.org/charts/
- Unicode strings are strings made up of unicode characters which includes ASCII characters but also a lot more (characters from other languages)
- Unicode strings have similar methods/operators as regular (ASCII) strings
- Try this:

```
x = chr(20860)
print(x, type(x), ord(x))
```

Unicode strings

- Also, try this and study it very carefully:

```
x = 'a'
y = u'a'
z = 'b'
s0 = x + z
s1 = y + z
print(s0, type(s0))
print(s1, type(s1))
print(s0[0], s0[0], type(s0[0]),type(s0[1]))
print(s1[0], s1[1], type(s1[0]), type(s1[1]))
print(s0[0], s0[1], ord(s0[0]), ord(s0[1]))
print(s1[0], s1[1], ord(s1[0]), ord(s1[1]))
```