CDS 230
Modeling and Simulation I

Module 4

Strings

Dr. Hamdi Kavak
http://www.hamdikavak.com
hkavak@gmu.edu

GEORGE MASON UNIVERSITY

Center for Social Complexity

Strings: what and why

- A variable type to store text.
  - Text means *letters* (a..z,A..Z) and *other symbols (0-9,*&…).*
- Made of a sequence of characters.
  - Characters are represented using the Unicode Standard*
- Can hold valuable information like DNA sequence
  - **A** = adenine **C** = cytosine **G** = guanine **T** = thymine
- ***Natural Language Processing*** deals with of all kind texts.
  - Language understanding, language translation, document summarization, named entity extraction, sentiment analysis…

*http://www.unicode.org/versions/Unicode12.1.0/

Creating strings

- Creation is straightforward … just put characters within quotes

```
'A string'
```

```
'A string'
```

- You can store a string in a variable

```
str1 = 'Hello'
str2 = 'world'
```

- You can use either single quote only or double quote only when creating a string. Do not mix the two.

```
str3 = 'hello"
```

Don’t do this

Getting user input

- Use `input()` function and assign to a variable

```
username = input("Your name:")
print(username)
```

```
Your name:
```

Type your name and hit enter. Whatever you write here will be a string assigned to the variable `username`

- You can convert input string (or any string) to a number by passing the string to the following functions
  - `int()` will convert a string to an integer number
  - `float()` will convert a string to a floating-point number
- What if the passed text is not a proper number?

Some string operations

- + operator concatenates two strings
  - The result is still a string.

```
str1 = 'Hello'
str2 = 'world!'
greeting = str1 + str2
print (greeting)
```

  - What if you concatenate a string with an integer?

```
str1 = 'Hello'
num = 11
result = str1 + num
print (result)
```

- * operator creates multiple copies

```
print('a'*3)
```

```
aaa
```

Can you guess the output

```
num = 16
char1 = '~'
print(char1*num)
num = int(num / 2)
print(char1*num)
num = int(num / 2)
print(char1*num)
num = int(num / 2)
print(char1*num)
num = int(num / 2)
print(char1*num)
```

Printing strings using `print()` function

```
dept = 'CDS'
school = 'GMU'
semester = 'spring'
year = 2020
```

1. Passing multiple parameters

```
print(dept,school,semester,year)
```

```
CDS GMU spring 2020
```

Extra space added between parameters

2. Concatenation

```
print(dept + school + semester + str(year))
```

```
CDSGMUspring2020
```

Number is converted to a string to concatenate with + operator

3. f-string

Don't forget 'f' here

```
print(f'{dept} dept at {school} welcomes {semester} {year} semester')
```

```
CDS dept at GMU welcomes spring 2020 semester
```

Just write variable names within curly braces. Python will take care of the rest.

Indexing

```
name = 'Jacob'
```

```
name[]
```

This is where you pass an index

- String variables use index numbers (within square brackets) to access individual characters.
- Index is a non-negative integer starting at zero.
  - 0 => **J** 1 => **a** 2 => **c** 3 => **o** 4 => **b**
- You can pass an integer variable as an index
- But you can't index beyond `length-1`.

```
name[2]
```

```
'c'
```

```
i = 3
name[i]
```

```
'o'
```

```
name[5]
```

```
---------------------------------------------
-----------------------------
IndexError
Traceback (most recent call last)
<ipython-input-21-94fccd0ebc6d> in <module>
----> 1 name[5]

IndexError: string index out of range
```

Length

- The `len()` function tells us how many characters are in the string, including whitespaces.

```
uni = 'George Mason University'
```

```
len(uni)
```

```
23
```

Slicing

- Used for extracting substrings from a string.

```
uni = 'George Mason University'
```

- Again, we use brackets, but the syntax is a bit different

```
uni[0:6]
```

```
'George'
```

start index 0 (inclusive)

colon

end index 6 (exclusive)

```
uni[:6]
```

```
'George'
```

These two statements are identical

More slicing

```
uni = 'George Mason University'
```

- Starting at a character and slicing till the end

```
uni[5:]
```

```
'e Mason University'
```

- Starting from the end. Note that it starts at -1.

```
uni[-1]
```

```
'y'
```

- Slicing last n characters.

```
uni[-5:]
```

```
'rsity'
```

Count

```
cst = 'Howdy World'
n = cst.count('H')
print(n)
```

```
1
```

The `count()` function will count the number of occurrences of a given substring.

```
n = cst.count('Wo')
print(n)
```

```
1
```

The `count()` function can take a substring with more than one character.

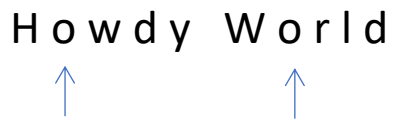
Some string functions

```
print(cst.lower())
howdy world

print(cst.upper())
HOWDY WORLD

print(cst.swapcase())
hOWDY wORLD
```

There are functions to convert the text to

- All lower case,
- All upper case, or
- Swap the cases.

Find



The first letter ‘o’ is at position 1.
The second ‘o’ is at position 7.

```
cst = 'Howdy World'

print(cst.find('o'))
1

print( cst.find('o',2))
7
```

The `find()` function will find the location of the first occurrence of a given substring.

We can also indicate where the search should start. In this case, the search starts at position 2, so it will not see the ‘o’ in position 1.
It finds the next ‘o’ which is at position 7.

Special characters

- What if you want to use quote as part of the string?
- Or represent a new line?
- In Python, certain sequence of characters have special meaning like \t representing tab.

Escape Sequence	Meaning
`\newline`	Backslash and newline ignored
`\\`	Backslash (\)
`\'`	Single quote (')
`\"`	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*

Source: https://docs.python.org/3/reference/lexical_analysis.html

Type check

- Python variable types are not explicit
- If you are not sure about the type of a variable, pass the variable to `type()` function.

```
name = 'Jacob'
print(type(name))

name = 111
print(type(name))
```

```
<class 'str'>
<class 'int'>
```

More strings?

- Check the course website for the following content
  - in operator
  - Comparisons
  - String functions

http://hamdikavak.com/course-modsim-1/

Homework 2 will be available on Blackboard at 11:45 am this Wednesday (Sep 15)