

# LECTURE 11

## STRING METHODS

## MATH AND RANDOM MODULES

MCS 260 Fall 2021

David Dumas

# REMINDERS

- Project 1 due today at 6pm central
- Homework 4 open, worksheet 4 solutions available
- Project 2 due Fri 8 Oct

# METHODS

Every value in Python is actually an **object**: data packaged together with functions that operate on the data.

Functions that are part of an object are called **methods**.

They are called with a special *dot* syntax:

```
>>> L=[1,2,3]
>>> L.append(4) # method of list
>>> s="sharks with lasers"
>>> s.split() # method of str
['sharks', 'with', 'lasers']
```

## We've seen a lot of methods so far:

### list

---

<code>.append(x)</code>	add x to end
<code>.insert(i,x)</code>	add x at index i
<code>.remove(x)</code>	remove first instance of x
<code>.pop()</code>	remove and return last element
<code>.index(x)</code>	get index of first x in the list

### dict

---

<code>.keys()</code>	return iterable of all keys
<code>.values()</code>	return iterable of all values
<code>.items()</code>	return iterable of key-value pairs

### str

---

<code>.split()</code>	split along whitespace, return list
<code>.lower()</code>	convert letters to lowercase

Note: **whitespace** means consecutive characters that all produce spaces or newlines (tab, space, "`\n`", ...)

Here are some additional str methods that are useful:

**str**

---

<code>.strip()</code>	remove leading and trailing whitespace
<code>.index(x)</code>	search for substring x and return index
<code>.upper()</code>	convert to uppercase
<code>.isdigit()</code>	Boolean: all characters digits?
<code>.isalpha()</code>	Boolean: all characters letters?
<code>.isspace()</code>	Boolean: all characters whitespace?
<code>.splitlines()</code>	Split along newline characters (returns list)
<code>.replace(old,new)</code>	Replace each occurrence of old with new.

# EXAMPLE

Let's write a program that takes a passage of text and computes the number of words and the number of distinct words.

# .JOIN()

The `.join()` method of a string `s` takes an iterable as parameter, and concatenates each element of the iterable with `s` between them.

```
>>> s = "+"
>>> L = ["me", "laser-sharks", "Shakespeare"]
>>> s.join(L)
'me+laser-sharks+Shakespeare'
>>> "".join(L)
'melaser-sharksShakespeare'
```

Using `"".join(iterable)` might be the most common case.

`.strip()`, `.split()`, `.join()`, `.splitlines()`, `.replace()` are probably the most-used methods for basic string processing.



# MORE ON .STRIP() AND .SPLIT()

The `.strip()` method takes an optional parameter, `chars` which must be a string. It removes any characters from `chars` from the start and end.

```
>>> s = "mathematics"  
>>> s.strip("sam")  
'thematic'
```

The `.split()` method takes an optional parameter, `sep`, which if given is the delimiter (instead of whitespace).

```
>>> s = "spiders and ticks and mites"  
>>> s.split(" and ")  
['spiders', 'ticks', 'mites']
```

# THE MATH MODULE

The statement

```
import math
```

loads the **math module**, after which functions and constants in that module can be used in your code, e.g.

```
math.sqrt(2)           # square root of 2  
math.exp(-1)          # 1/e  
math.sin(0.5*math.pi) # 1.0
```

The math module includes all common trig functions, logarithms and exponentials, and the constants pi and e. The [the math module docs](#) have a full list.

# THE RANDOM MODULE

The random module, imported with

```
import random
```

includes functions to generate (pseudo)random numbers, e.g.

```
>>> random.random()      # Random float in [0,1)
0.482824082899013
>>> random.randint(1,6) # Random int between 1 and 6 inclusive
5
>>> random.choice( ["Yes", "No", "Maybe"] ) # random item of seq
'No'
>>> L = ["a", "b", "c", "d"]
>>> random.shuffle(L)    # IN-PLACE shuffle of a list
>>> L
['a', 'd', 'c', 'b']
```

# PSEUDORANDOM NUMBERS

Python uses an equation to generate pseudorandom numbers, starting from some initial data, the **seed**. By default, the seed is computed from the current time.

So the numbers returned are not random at all!

The PRNG can be manually seeded using `random.seed(value)`.

```
>>> random.seed(42)
>>> random.random()
0.6394267984578837
>>> random.seed(42)
>>> random.random()
0.6394267984578837
```

# REFERENCES

- In *Downey*:
  - Section 8.8 discusses a few string methods
  - Section 10.9 discusses `split()`
  - Section 3.2 discusses the `math` module
  - Section 13.2 discusses the `random` module

# REVISION HISTORY

- 2021-09-16 Initial publication