CSCU9YE - Artificial Intelligence

Lecture 2: Introduction to Python

Prof. Gabriela Ochoa, University of Stirling

Resources

- Books
  - Lutz, M. Learning Python. 5th Edition (2013) http://learning-python.com
  - Downey, A. B. Think Python, 2nd Edition. (2015) http://greenteapress.com/wp/think-python/
- Online tutorials
  - https://www.w3schools.com/python/
  - https://www.python.org/about/gettingstarted/

Why Python?

Advantages

- Friendly & easy to learn
- Both procedural & object oriented
- Open source & portable
- Code that is as understandable as plain English. Very compact
- Work well with others (glue)
- Powerful for Data Science and Machine Learning (libraries)

Disadvantage

- It is an interpreted language rather than compiled– hence might take up more CPU time.

- Released by its designer, Guido Van Rossum (Dutch researcher), in 1991
- The language got its name, not from those dangerous reptiles, but from a BBC comedy series from the70’A "Monty Python's Flying Circus".

Interpreters vs. Compilers

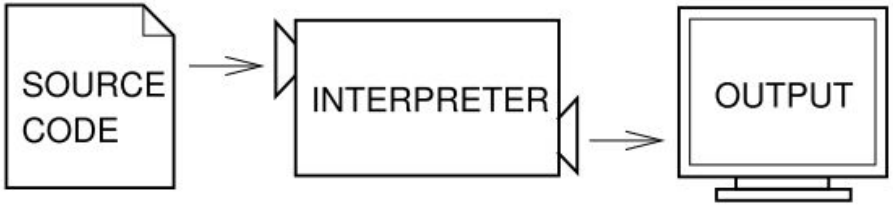


Figure 1.1: An interpreter processes the program a little at a time, alternately reading lines and performing computations.

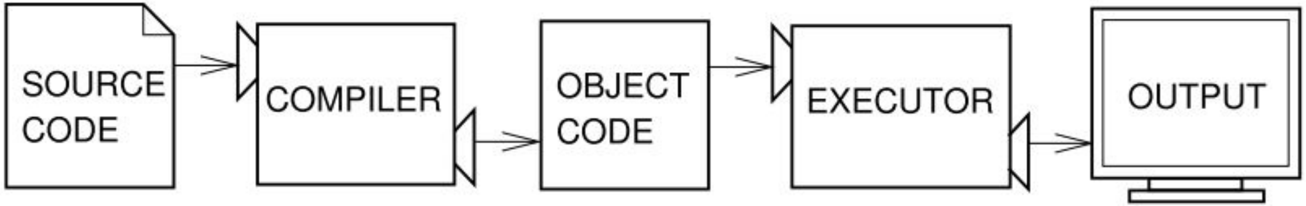


Figure 1.2: A compiler translates source code into object code, which is run by a hardware executor.

(Downey, 2015)

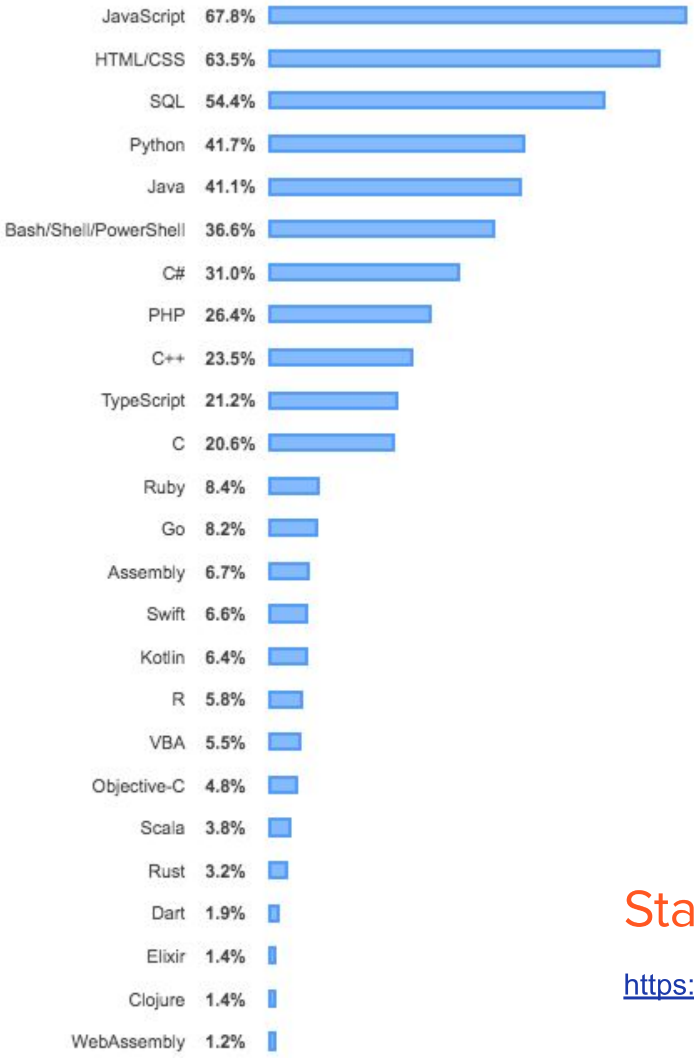
Python is an interpreted language

Two ways of using the Python interpreter

1. **Interactive mode**, you type Python programs and the interpreter displays the results

   ```
   >>>  1 + 1
   2
   ```

2. **Script mode,** store the code in a file and use the interpreter to execute its content
   - By convention, Python scripts have names that end with `.py`
   - To execute the script, you have to tell the interpreter the name of the file: `python myscript.py`



Most popular technologies

Programming, Scripting, and Markup Languages

- For the 7th year in a row, JavaScript is the most commonly used programming language,
- Python has risen in the ranks again.
- This year, Python just edged out Java in overall ranking, much like it surpassed C# last year and PHP the year before.
- Python is the fastest-growing major programming language today.

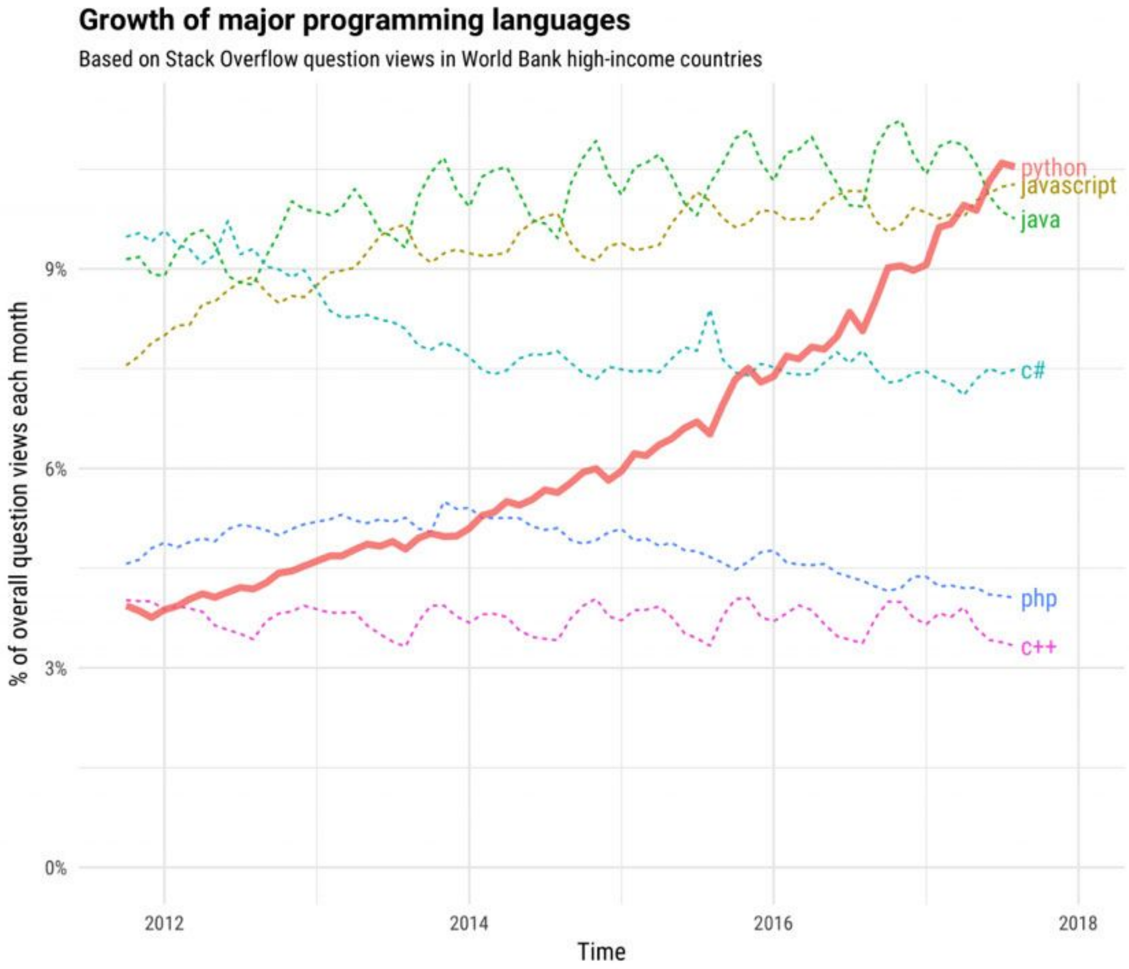
Stack Overflow’s annual Developer Survey

https://insights.stackoverflow.com/survey/2019#top-paying-technologies

Top programming languages 2019

Top Programming languages as per GitHut 2.0

https://codeburst.io/10-top-programming-languages-in-2019-for-developers-a2921798d652



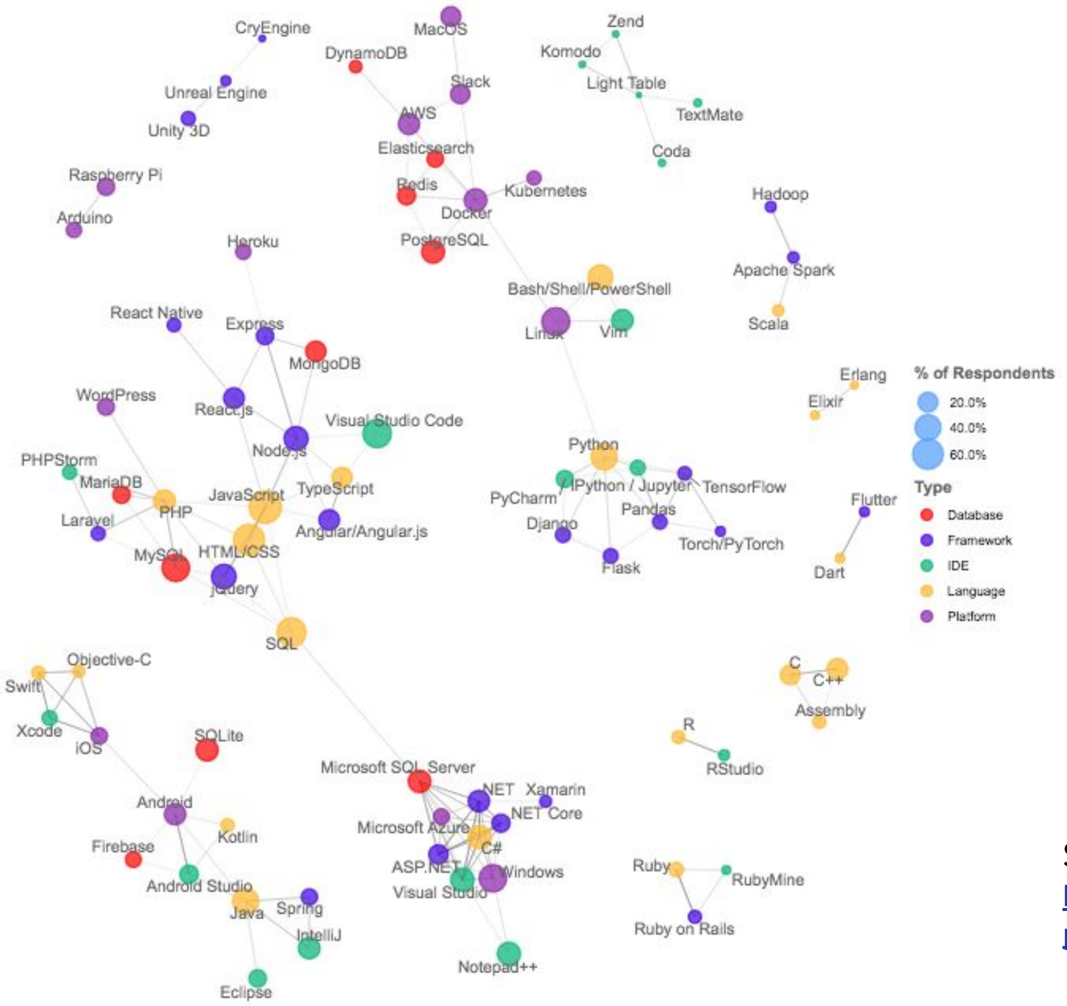
Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries

Python

- One of most popular and promising programming languages in the last few years
- Extraordinary growth
- The de-facto choice for machine learning and deep learning.
- StackOverflow’s recent study that focused on high-income countries (those defined wealthy by World Bank), Python was found to be more popular than JavaScript.

https://stackoverflow.blog/2017/09/06/incredible-growth-python/



The network shows which technologies are most highly correlated with each other.

- Left: web-development
- Down middle: Microsoft
- Lower left: mobile
- Middle : Python
- Top middle: operation technologies

Source: https://insights.stackoverflow.com/survey/2019#top-paying-technologies

Our first python program

C

```c
#include "stdio.h"
int main() {
   printf("Hello World\n");
}
```

Python

Notice: no ;

```python
print "Hello World"  # Python 2

print("Hello World") # Python 3
```

Java

```java
public class Hi {
  public static void main (String [] args) {
    System.out.println("Hello World");
  }
}
```

Python is dynamically typed

C

```c
#include "stdio.h"
int main() {
   int x = 3;
   int y = 4;
   printf("%s"\n,x+y);
}
```

Python

```python
x = 3
y = 4
print(x+y)
```

Notice: no explicit type declaration
But there are still types behind the scenes.

Python is dynamically typed

C

```
#include "stdio.h"
int main() {
   int x = 3;
   x= 4.5;
}
```

What happens if we try to do this?

Python

```
x = 3
x = 4.5
```

No error in Python

Dynamic Typing!!

Variables

- No need to declare
- Need to assign (initialise, use of uninitialised variable raises exception)
- Not typed

```
if friendly: greeting = "hello world"
else: greeting = 12**2
print greeting
```

- ***Everything*** is a "variable":
    - Even functions, classes, modules

Common Types in Python

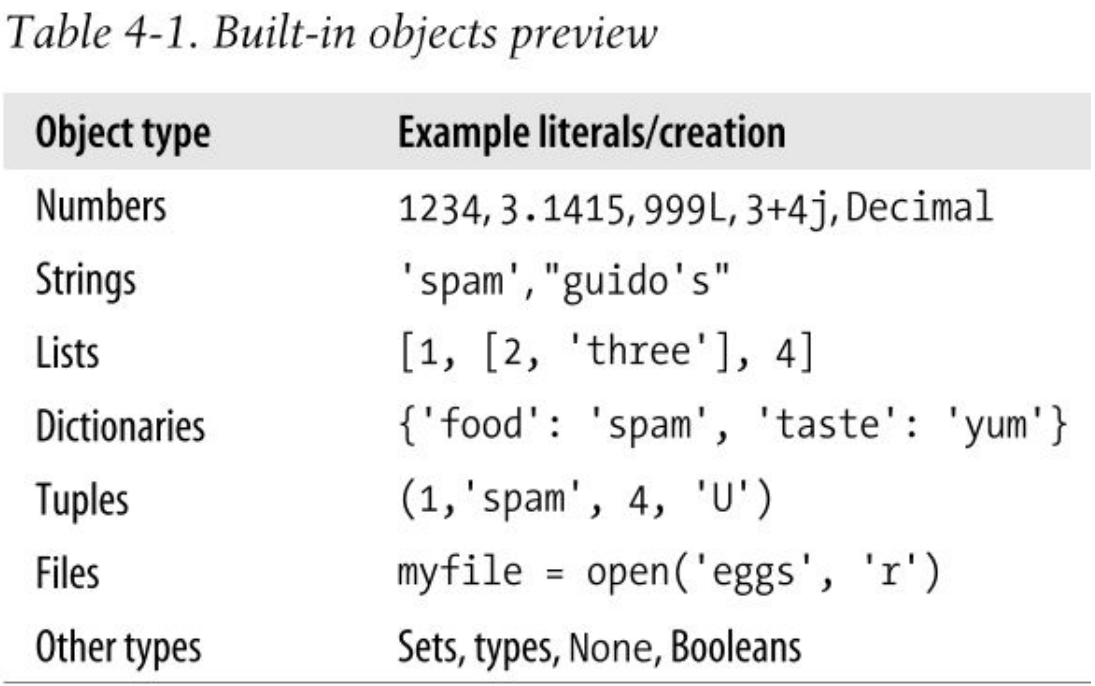
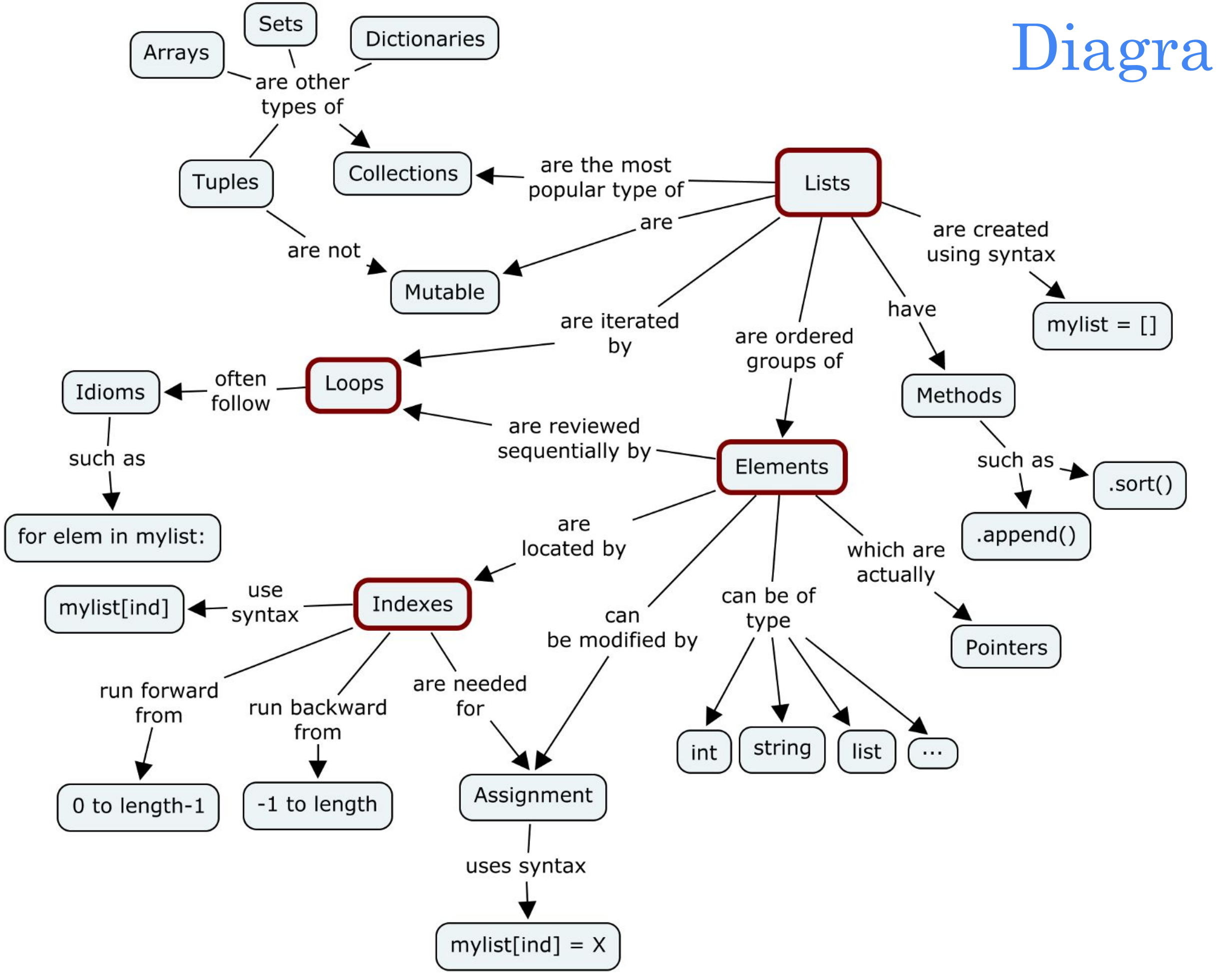


Table 4-1. Built-in objects preview

Object type	Example literals/creation
Numbers	`1234`, `3.1415`, `999L`, `3+4j`, `Decimal`
Strings	`'spam'`, `"guido's"`
Lists	`[1, [2, 'three'], 4]`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`
Tuples	`(1,'spam', 4, 'U')`
Files	`myfile = open('eggs', 'r')`
Other types	Sets, types, `None`, Booleans

Diagram of Python



https://medium.com/@meghamohan/mutable-and-immutable-side-of-python-c2145cf72747

For Loops

C

```c
#include "stdio.h"
int main() {
   int i = 0;
   for(i=0; i < 10; i++) {
      printf("%d\n",i);
   }
}
```

Python

```python
for i in range(0,10):
   print i
```

```
range(start, stop[, step])
```

Returns values between start and stop, increasing by the value of step (defaults to 1).

What is the output of this loop ?

```python
for i in range(0,10,2):
   print i
```

While Loops

Syntax

```
while condition:
    statements
```

Example

```
i = 2
while i < 12:
    print(i)
```

```
2
5
8
11
```

Conditional

Syntax

```
if condition:
    statements
[elif condition:
    statements] ...
else:
    statements
```

Example: "dog years algorithm"

```
age = input("Age of the dog: ")
print
if age < 0:
   print("This cannot be true!”)
elif age == 1:
   print("about 14 human years”)
elif age == 2:
   print ("about 22 human years”)
else:
   human = 22 + (age -2)*5
   print ("Human years: ", human)
```

Grouping Indentation

C

```c
for (i = 0; i < 20; i++) {
   if (i%3 == 0) {
      printf("%d\n", i);
      if (i%5 == 0)
        printf("Bingo!\n");
    }
  printf("---\n");
}
```

Python

```python
for i in range(20):
    if i%3 == 0:
        print i
        if i%5 == 0:
            print "Bingo!"
    print "---"
```

```
0
Bingo!
---
---
---
3
---
---
---
6
---
---
---
9
---
---
---
12
---
---
---
15
Bingo!
---
---
---
18
---
---
```

Functions, Procedures

```
def name(arg1, arg2, ...):
    """documentation"""    # optional doc string
    statements


return                   # from procedure
return expression             # from function
```

Example Function

```
def gcd(a, b):
    "greatest common divisor"
    while a != 0:
        a, b = b%a, a    # parallel assignment
    return b
```

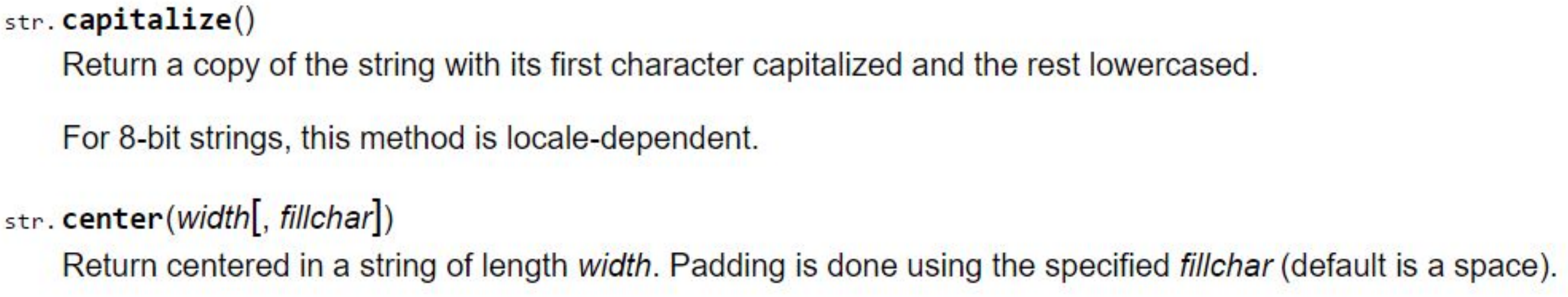
```
>>> gcd.__doc__
'greatest common divisor'
>>> gcd(12, 20)
4
```

String objects

https://www.w3schools.com/python/python_ref_string.asp

More examples

Example of String Methods



str.**capitalize**()

Return a copy of the string with its first character capitalized and the rest lowercased.

For 8-bit strings, this method is locale-dependent.

str.**center**(*width*[, *fillchar*])

Return centered in a string of length *width*. Padding is done using the specified *fillchar* (default is a space).

```
>>> x = "Hello"
>>> x.lower()
'hello'
```

<var_name>.<method_name>(params)

Strings

"hello"+"world"	"helloworld"	# concatenation
"hello"*3	"hellohellohello"	# repetition
"hello"[0]	"h"	# indexing
"hello"[-1]	"o"	# (from end)
"hello"[1:4]	"ell"	# slicing
len("hello")	5	# size
"hello" < "jello"	True	# comparison
"e" in "hello”	True	# search

Lists

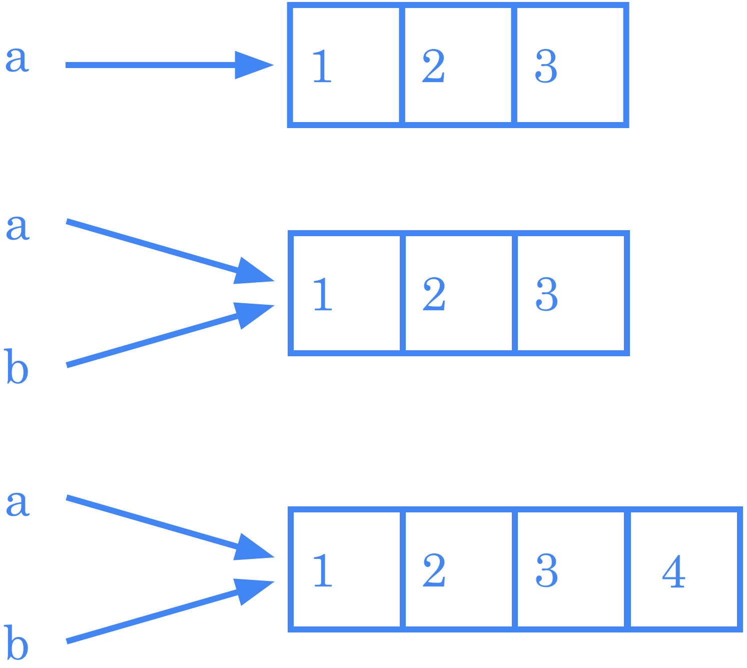
- Flexible arrays
  - a = [99, 56, 67, 45]
- Same operators as for strings
    - a+b, a*3, a[0], a[-1], a[1:], len(a)
- Item and slice assignment
    - a[0] = 98 # [98, 56, 67, 45]
    - a[1:3] = [57, 68] # [98, 57, 68, 45]
    - del a[-1] # [98, 57, 68]

More List Operations

```
>>> a = range(5)        # [0,1,2,3,4]
>>> a.append(5)         # [0,1,2,3,4,5]
>>> a.pop()             # [0,1,2,3,4]
5
>>> a.insert(0, 42)     # [42,0,1,2,3,4]
>>> a.pop(0)            # [0,1,2,3,4]
42
>>> a.reverse()          # [4,3,2,1,0]
>>> a.sort()             # [0,1,2,3,4]
```

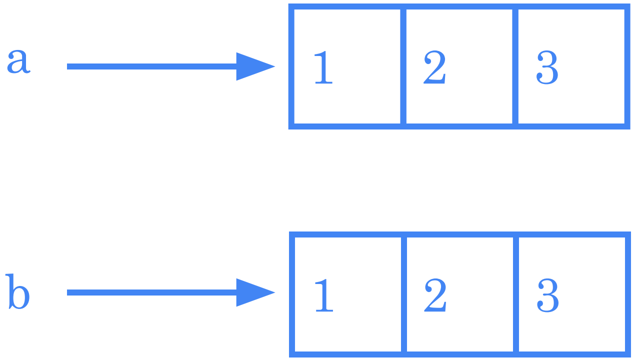
Changing a Shared List

```
a = [1, 2, 3]

b = a

a.append(4)
```



Copying a List

```
a = [1, 2, 3]

b = a [:]
```



File Objects

- f = open(*filename*[, *mode*[, *buffersize*])
    - mode can be "r", "w", "a" (like C stdio); default "r"
    - append "b" for text translation mode
    - append "+" for read/write open
    - buffersize: 0=unbuffered; 1=line-buffered; buffered
- methods:
    - read([*nbytes*]), readline(), readlines()
    - write(*string*), writelines(*list*)
    - seek(*pos*[, *how*]), tell()
    - flush(), close()
    - fileno()

Classes

```
class name:
    "documentation"
    statements
```

-or-

```
class name(base1, base2, ...):
    ...
```

Most, *statements* are method definitions:

```
    def name(self, arg1, arg2, ...):
        ...
```

May also be *class variable* assignments

Example Class

```
class Stack:
    "A well-known data structure…"
    def __init__(self):            # constructor
        self.items = []
    def push(self, x):
        self.items.append(x)
    def pop(self):
        x = self.items[-1]
        del self.items[-1]
        return x
    def empty(self):
        return len(self.items) == 0    # Boolean result
```

Using Classes

- To create an instance, simply call the class object:

```
x = Stack()    # no 'new' operator!
```

- To use methods of the instance, call using dot notation:

```
x.empty()              # > 1
x.push(1)              # [1]
x.empty()              # > 0
x.push("hello")        # [1, "hello"]
x.pop()                # > "hello" # [1]
```

- To inspect instance variables, use dot notation:

```
x.items        # -> [1]
```

Summary & What’s next?

Python

- Interpreted language, both procedural and OO modes
- Emphasizes code readability (using whitespace indentation to delimit code blocks rather than curly brackets)
- Simple syntax, fewer lines of code
- Open source, portable, mixes well with other languages
- Widely used for AI and Machine Learning

We did not cover some important features of Python such as tuples, dictionaries, exceptions, modules and packages and list comprehension.