>>

Python (i)

- Python
- Python Basics
- More on Python

^ >>

❖ Python

Python is a very popular programming language

- easy to learn/use
- with a wide range of useful libraries

We assume that you know enough Python to manipulate DBs

- the primary goal is Database, not Python programming

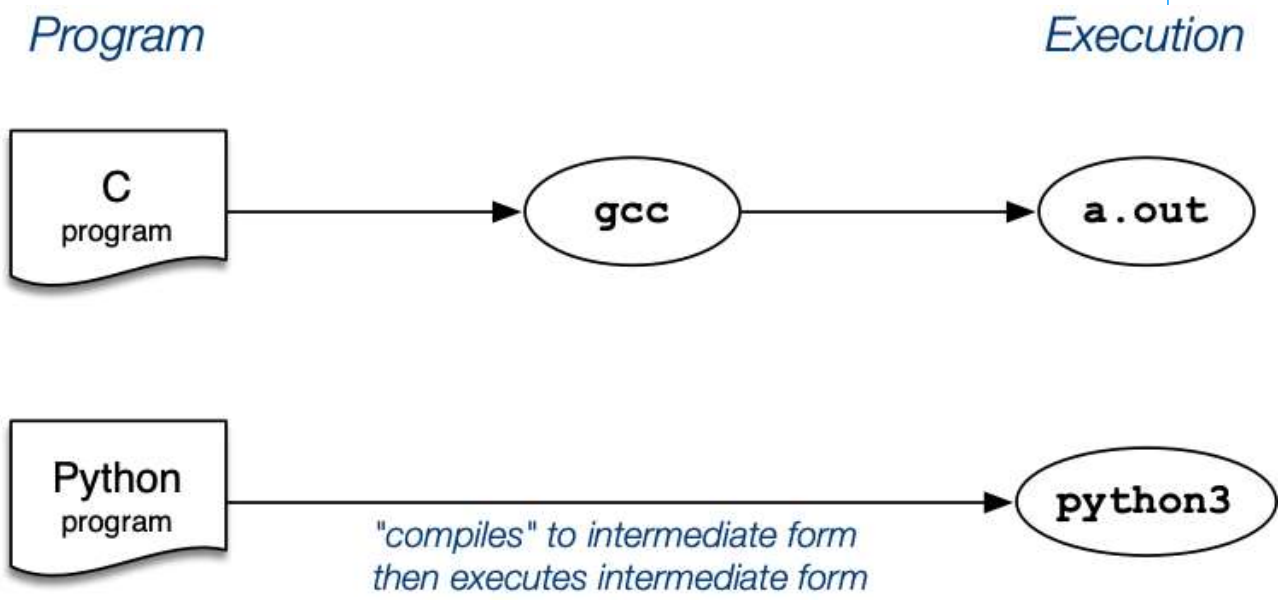
If you're not overly familiar with Python ...

- there will be many examples of Python code in this course
- there are many excellent tutorials online
- some of this content was "borrowed" from COMP1531 lectures

<< ^ >>

❖ Python (cont)

Unlike C, Python is an interpreted language



Such languages are often called "scripting languages"

❖ Python (cont)

Python has an interactive interface (like **psql**)

```
$ python3
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" ...
>> print("Hello, world")
Hello, world
>> quit()
$
```

Or you can run programs that are stored in files

```
$ echo 'print("Hello, world")' > hello.py
$ python3 hello.py
Hello, world
$
```

❖ Python Basics

Like C, Python programs consist of

- expressions, statements, control structures, function definitions, imports, ...

Unlike C, Python uses indentation to indicate code nesting, e.g.

Python

```
if Condition:
   Statements1
else:
   Statements2
Next Statement
```

C

```
if (Condition ) {
   Statements1
} else {
   Statements2
}
Next Statement
```

<< ^ >>

❖ Python Basics (cont)

Comments are introduced by #, to end of line

Data types and constants:

- booleans, e.g. `True, False`
- numbers, e.g. `1, 42, 3.14, -5`
- strings, e.g. `"a string", "string2", 'it\'s fun'`
- lists, e.g. `[1,4,9,16,25], ['a','b','c']`
- tuples, e.g. `(3,5), (1,'a',3.0)`
- dictionaries, e.g. `{'a': 5, 'b': 98, 'c': 99}`

Assignment is via =, "the usual" operators are available

❖ Python Basics (cont)

Example operators and expressions

```
name = "Giraffe"
age = 18
height = 2048.11 # mm

print(name + ", " + str(age) + ', ' + str(height))
print(f"{name}, {age}, {height}")
print(type(name))
print(type(age))

n = 16 // 3
print(f"3 ** 3 == {3 ** 3}")
print(f"16 / 3 == {16 / 3}")
print(f"16 // 3 == {n}")
```

❖ Python Basics (cont)

Defining functions

```
# recursive factorial

def fac(n):
    if n <= 1:
        return 1
    else:
        return n * fac(n-1)

print('5! =',fac(5))
```

❖ Python Basics (cont)

Defining functions (cont)

```
# iterative factorial

def faci(n):
    f = 1
    for i in range(1,n):
        f = f * i
    return f

print('6! =',faci(6))
```

A collection of related functions can be packaged into a module

❖ Python Basics (cont)

C programs can import library definitions, e.g.

```
#include <stdlib.h>
#include <stdio.h>
```

Python programs can import external modules (module = collection of definitions)

```
import sys
import psycopg2
import sound.effects.echo
from sound.effects import echo
```

Packages (e.g. **sound**) are collections of sub-packages and modules

❖ Python Basics (cont)

Example: **echo** in Python

```
import sys
                        # extracts a slice from argv
for arg in sys.argv[1:] :
   print(arg, end=" ")    # don't put '\n' after print
print("")
```

which is is used as (if placed in file **echo.py**)

```
$ python3 echo.py arg1  "arg 2"  arg3
arg1 arg 2 arg3
$
```

❖ Python Basics (cont)

Example: sequence generator ... 1 2 3 4 5 ...

```
#!/usr/bin/python3
import sys
if len(sys.argv) < 3:
   print("Usage: seqq lo hi")
   exit(1)
hi = int(sys.argv[2])
i = lo = int(sys.argv[1])
while i <= hi:
   print(i)
   i += 1    # no ++ operator
```

which can be used as which is is used as (if placed in executable file **seqq**)

```
$ ./seqq 2 4
2
3
4
$
```

<< ^

❖ More on Python

Lots of info available on **`python.org`**

- including docs.python.org/3/tutorial/introduction.html

And many others, e.g. www.learnpython.org

Or ask for "free python3 tutorials" on Google

Python has hundreds of modules/libraries on all kinds of topics

We focus on the **`psycopg2`** database connectivity module

Produced: 26 Oct 2020