Introduction to programming with Python 3

What is programming?

- "Coding"
  - Writing a series of instructions to accomplish a goal
  - This is actually the *second* part
- Planning
  - Deciding *what* needs to be done to accomplish the goal, and *how* to do it
  - This should be the *first* part of the process

"Programming" a meal

- Plan:
  - What to eat?
  - What ingredients needed?
  - How to prepare?
  - How to serve?
- Do the work ("coding"):
  - Get out ingredients
  - Combine, cook as necessary
  - Put on plate and serve

A basic pattern for most programs

- Simple *program*, maybe a *script*:
  1) Get *inputs* for the user
  2) *Process* the inputs – manipulate them to calculate a desired result
  3) Produce *output* for the user
- The "user"
  - Maybe a person running the program
  - Maybe the "real world" – sensors, motors, etc.

More elaborate program patterns

- Do something in a "loop":

```
while not_done:
    get input
    process input to generate results
    produce output
    decide if done or not
```

- "Event-driven" program:

```
wait until something happens
    process the event
    generate an output
repeat (forever?)
```

Programs and Scripts

- Program – the set of instructions that accomplish a goal
  - Written in a programming language such as Python (or Java, Javascript, C, Assembly, etc....)
- Script – a particular style of program
  - Usually interacts with *other* programs to govern their usage
  - Often more accessible than a "pure" program

Interpreted versus Compiled

- Python is *interpreted*
  - Each program statement translated into machine instructions, then executed immediately

  * *Easier to find and fix errors*

- Some languages are *compiled*
  - C, Fortran, Java, etc.
  - All program statements translated into machine instructions first, saved in disk file for later use

  * *Better long-term performance*

Interactive programming

- Perform one command, see the result
- Good for exploratory work, figuring out how to do something
  - but tedious for repeated tasks
- Depends on an interpreted language and an interactive environment
  - *a.k.a.* IDE, "Integrated Development Environment"

Programming and Python

What is Python?

- A programming language
  - not a snake
  - not even a comedy troupe
- A **scripting** language
  - Scripts issue commands to othe
- A “Swiss Army Knife”
  - Command-line interpreter for ad-hoc stuff
  - Long-term applications
- Versions: Python2 versus Python3
  - Both in use, because some modules not updated

What is Python 3?

- A *programming language*
  - A specific way of writing the instructions to do a task
  - "Syntax" – a language's rules for how to express something

    Partly characterizes that language
- Version 3 of Python is mostly compatible with version 2
  - Syntax is mostly the same, but some operations have changed

What can I do with Python?

- Casual calculator
- Scientific programming
- Graphical / GUI programming
- Games programming (!)
- **Scripting**
  - Programs that control other programs or the operating system
  - Very goal-oriented
  - Usually text-oriented, not graphical

Python's interpreted environments

- "**`python`**" – most basic usage
- "**`ipython`**" – better help for typing
- "**`jupyter qtconsole`**"
  - separate window
- "**`jupyter notebook`**"
  - runs in a browser
- Jupyter environments include support for scientific computing

Getting Python for yourself

- Basic package – from the origin:
  - https://www.python.org/downloads/
- Jupyter and ipython:
  - https://www.anaconda.com/download/
- Other releases, for specialized needs:
  - IronPython – Python 2 only
  - ActiveState – free, commercial editions
  - Enthought Canopy – bundle for data science

others…

An IDE

- Many IDEs are available
  - Free ones
  - Commercial (paid) ones
- Geany ("genie")
  - Cross-platform: Linux, MacOS, Windows
  - Supports Python and other languages
  - Free
  - Windows: browse to http://download.geany.org/geany-1.33_setup.exe

Starting out

- Start python – from a command line:
  `jupyter qtconsole`
- Try some processing statements:
  - »**`3 + 7`**
  - »**`"Hello" + "world"`**
    compare to **`"Hello " + "world"`** with a space
  - »**`x = 99 * 48`**
    Where's the answer???
    (and what is *x*?)

Remark

- **DON'T INDENT** statements (yet)
  - Indentation has an important meaning in Python (unlike other programming languages)
- Everything we are doing for now occurs at the outermost level of indentation
- Later language features will *require* indentation – wait until then

Output

- Control the output with print():

```
»print(x)
```

Ah, there's the answer!

```
»print(5-9)

»print(5, "-", 9)

»print("Hello " + "world")
```

Python in a Script File

- Script file (or "program" file):
  - Text file
  - Suffix ***must be*** ".py" (not ".txt")
  - Python statements are individual lines in the the file
  - The "print()" statement provides output
- Using a script file:
  - Enter
    `python3 myscript.py`
    on command line

try it: save these lines in a file

```
#!/usr/bin/env python3
# convert meters to feet,
# the hard way.

m = 17    # pick a number, any number...

cm_per_m = 100
in_per_cm = 1 / 2.54
feet_per_in = 1 / 12

ft = m * cm_per_m * in_per_cm * feet_per_in
print(m, 'meters equals', ft, 'feet.')
```

Input; Assignment

- *Assignment* – give a named *variable* a new value
  - »`x = 99 * 48`
  - »`mystring = "Hello world"`
  - »`y = max(3, 7, -99)`
- The **`input()`** function assigns user input string to a variable
  - »`invar = input()`

More input

- **`input()`** takes an optional argument for a *prompt*
  - »`value = input("What value? ")`
- Always returns a string
  - Convert strings to numbers with:
    - int()
    - float()
  - »`xval = int( input("Number? ") )`

Number types: int( ) versus float( )

- What kind of numbers are:
  - 14
  - 5
  - float(5)
- What kind of number is:
  - 14 / 5
- What kind of number is:
  - 14 // 5
- What is:
  - int(14 / 5)

Put it together

- A simple program:

input portion

```
»a = float( input("a? ") )
»b = float( input("b? ") )
```

processing

```
»x = a / b
```

output portion

```
»print("Answer is", x)
```

Is that all there is?

- Most of this course is about different ways to *process* the inputs
- Also, how best to provide user-friendly *outputs*
- Less focus: *inputs*
  - Multiple inputs
  - Where to get inputs from
    - a user with a keyboard?
    - a disk file?

(and the last shall be first….)

- Multiple inputs – example:

```
Your name: Springbean
Height (feet and inches)? 6 8
Weight (pounds)? 125
```

- This shows *two* numbers being provided to a *single* **`input()`** statement (with a prompt)
- Input is a string of two digits and a space character between them

Extract the pieces

- Input looks like
  "6 8"

```
Height (feet and inches)? 6 8
Weight (pounds)? 125
```

- Split into two parts:
  - **feets, inchs = "6 8".split()**
    feets gets "6"
    inchs gets "8"
- Convert each part to a number
  - **feet = int(feets)**
  - **inches = int(inchs)**

Try it

- Three-part name
  - **name = input('full name? ')**
  - **f, m, l = name.split()**
    f gets "Robert"
    m gets "Abdon"
    l gets "Montante"
- date
  - **datestr = input('Date (m/d/y)? ')**
  - **month, date, year =**
    datestr.split('/') # split on '/'
  - **month = int(month) # redefine "month"**

Summary

- Program patterns
- Interactive calculations
- Script files
- Output – **`print()`**
- Variables
- Input – **`input()`**
- Type conversion – **`float()`**, **`int()`**
- Splitting up a string – **`.split()`**