Python Part 1

Python vs C++

- Python is a popular alternative/complement to C++
- Simpler syntax resulting in shorter and simpler code
- Interpreted language - slower than compiled C++
  - Available via interactive interpreter and *.py files
- Automatic variable declarations and memory management
- Very useful to be familiar with both

Installing and running python

- Python usually needs to be installed manually
  - Windows: can be installed using Microsoft Store
  - Linux: often already installed or available through package manager
  - Direct download: https://www.python.org/downloads/
- Generally you will want the latest version
  - Make sure you have at least version 3.6 for this class
- Run using python command
  - Often py (for Windows) or python3 are used
- Recommend Python extension in VSCode

Python versions

- Python versions are often not mutually compatible
  - Significant break between python 2 and python 3
  - Generally, python 3 is considered the default now, but python 2 is sometimes needed
- Check which version you are running with python --version
- Versions of individual packages are also often incompatible
  - It is often necessary to carefully set up your work environment to run complex code
- Virtual environments are often used to ensure versions are all correct
  - Beyond the scope of this class

Python interpreter

- Run python without a *.py file to enter the interactive interpreter
  - You will be prompted with >>>
- Python commands can be entered directly line-by-line
  - Very useful for simple procedures, but not great for complex code
- Quit interpreter using exit()

Basic python syntax

- No line terminators except when defining scope
- Scopes are defined by a line ending with colon (:)
  - All lines within scope must be indented (at least one space)
  - Nested scopes done with multiple levels of indentation
  - Be careful to ensure scoping is as expected
- Single line comments are denoted with #
- Block comments denoted as """ … """ or ''' … '''
- Arguments are defined with (…)

Packages

- Included functionality is rather limited, but extensive packages are available
- Many standard packages already come installed, e.g.,:
  - math: same mathematical functions as cmath in C++ (python cmath is for complex numbers)
  - os: interfaces to OS functionality (e.g., bash functionality from within python)
  - time: access to current time
- Include packages using import command
  - Can provide local name to save on typing

```
import math
import calendar as cal

math.sqrt(4)
cal.weekday(2023,2,28)
```

Installing packages

- There are many other available packages: https://pypi.org/
- For almost everything you want to do, there is a package with the functionality
  - If you are writing a complex algorithm, it likely already exists
- Install packages using pip (already available for 3.6 or newer):
  - pip install <package> - install a package
  - pip list - list installed packages with version information
  - pip install <package>=x.y.z - install version x.y.z of a package
  - pip install --upgrade <package> - update installed package to latest version
  - pip uninstall <package> - uninstall a package

Variables

- Variables do not need to be declared
  - Variable created and type assigned when value is first assigned
  - Types can change after they have been set
- Variable types can be cast using e.g., str(...), int(...) and float(...)
- Check the type of a variable with type(...)

```
x = 7
y = "Some text"
```

Input and output

- Print to screen using print() command
- Read in user input using input(<prompt>) command

```
text = input("Please enter text: ")
print("Your text is: " + text)
```

Mathematical operators

- Arithmetic
  - + : addition
  - - : subtraction
  - * : multiplication
  - / : division
  - % : modulus (remainder divide)
  - ** : exponentiation
  - // : floor division
- Logical
  - and : logical AND
  - or : logical OR
  - not : logical NOT
- Assignment
  - = : assign value
  - += : increase by value
  - -= : decrease by value
  - *= : multiply by value
  - /= : divide by value
  - %= : modulus by value
- Comparison
  - == : equal to
  - != : not equal to
  - > : greater than
  - < : less than
  - >= : greater than or equal to
  - <= : less than or equal to

Logical flow controls

```
if <condition 1>:
    <do something>
    if <another condition>:
        <do something>
elif <condition 2>:
    <do something>
else:
    <do something>
```

While loop

```
i = 1
while i < 10:
    print(i)
    i += 1
```

```
i = 1
while i < 10:
    if i == 4:
        # break out of loop
        break
    print(i)
    i += 1
```

```
i = 1
while i < 10:
    if i == 4:
        # skip to next iteration
        continue
    print(i)
    i += 1
```

For loop

```
# loop over a list and print each element
fruits = ["apple","banana","orange"]
for x in fruits:
    print(x)

# print each letter in a string
for x in "apple":
    print(x)

# print integers 0 to 9
for x in range(10):
    print(x)
```

Resources

- https://www.w3schools.com/ - Great online learning resource
- https://www.youtube.com/@codebreakthrough - Excellent tutorial videos
- https://wiki.python.org/moin/BeginnersGuide - Good documentation
- https://learn.microsoft.com/en-us/windows/python/ - For Windows users
- https://www.python.org/
- https://stackoverflow.com/ - Ask questions to experts