Python Foundations Continued

Why Python?

- very common scripting language
  - quick to write and test
  - slower to run compared to compiled languages
- easy to learn!
  - English-like syntax
- hard to learn!
  - English-like syntax
  - overloaded operations by default
- convenient
  - precoded packages and libraries with added functionality
- very flexible
  - multiple ways to do the same thing

Topics covered by Chapter 2

low-level language
high-level language
interpreted language
compiled language
source code
machine code
Python
integrated development environment (IDE)
Anaconda
Spyder
IPython console
shell
program (script)
command (statement)
object
type
scalar object
non-scalar object
literal
floating point
bool
None
operator
expression
value
shell prompt
variable
binding
assignment

Topics covered by Chapter 2

reserved word
comment (in code)
straight-line program
branching program
conditional
indentation (in Python)
nested statement
compound expression
constant time
computational complexity
conditional expression
strings
overloaded operator
repetition operator
type checking
indexing
slicing
type conversion (casting)
formatted string expression
input
Unicode
iteration (looping)
pseudocode
while loop
hand simulation
break
for loop
tuple
range
in operator
PEP 8 style guide

Group Activity

Group the terms by these categories:

1. About programming languages
2. Related to Calculations
3. Related to Control
4. Misc.

low-level language
high-level language
interpreted language
compiled language
source code
machine code
Python
integrated development environment (IDE)
Anaconda

Spyder
IPython console
shell
program (script)
command (statement)
object
type
scalar object
non-scalar object
literal

floating point
bool
None
operator
expression
value
shell prompt
variable
binding
assignment

reserved word
comment (in code)
straight-line program
branching program
conditional
indentation (in Python)
nested statement
compound expression
constant time
computational complexity
conditional expression

strings
overloaded operator
repetition operator
type checking
indexing
slicing
type conversion (casting)
formatted string expression
input

Unicode
iteration (looping)
pseudocode
while loop
hand simulation
break
for loop
tuple
range
in operator
PEP 8 style guide

Why do we care about Calculation and Control?

Algorithms in Python

- contain **calculations** and **control statements**
- ^^^ doing ^^^flow of execution, checking, skipping, finishing
- Can you identify the calculations and control statements?

An algorithm is like a recipe from a cookbook:

1. Put custard mixture over heat.
2. Stir.
3. Dip spoon in custard.
4. Remove spoon and run finger across back of spoon.
5. If clear path is left, remove custard from heat and let cool.
6. Otherwise repeat.

First Spec Lab

Number comparisons

number-comparison-starter

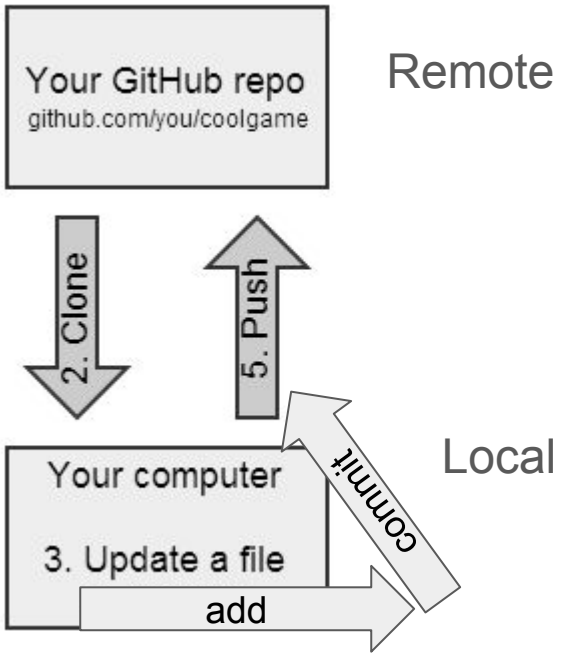
Technical Goals

- **Clone** a **repo** and update a **remote repo** using **git**
- **locally** load the **virtual environment**
- receive **Gatorgrade** report
- construct an **algorithm** using python fundamentals
- write **markdown** with correct formatting
- read and correct **linting** errors

Dealing with Repos on GitHub (GH)

A repository (or **repo**) that is on GH is a set of files that is stored in **online**.

- **git** is a **local** program (not online) that allows you to copy and modify the **remote** files (online)
- copying is done with **clone** or **pull**
- modifying is done with a sequence of three things: **add**, **commit**, **push**



git commands in terminal

first time only

```
git clone git@github.com:allegheny-college-cmpsc-101-fall-2024/course-materials.git
```

….edit files in Sublime Text….. save normally

```
git status

git add .    ← don't forget the `.`

git status

git commit -m "summarize your actions in a short, identifiable way."
```

no connection to remote

```
git push origin main
```

remote connection