Introduction

Intermediate Application Development

Otago Polytechnic
Dunedin, New Zealand
Kaiako: Tom Clark

Administration

- Communication will take place on Teams and email
- We will use Git and Github extensively. You'll need to have a GitHub account and to set up Git on your computer.
- Course materials
  - Course notes and worksheets:
    https://github.com/tclark/op-intermediate-app-dev
  - Practical submissions:
    `https://classroom.github.com/a/ZOav9L3E`
  - Project one submissions:
    `https://classroom.github.com/a/rfDZDOUC`
  - Project two submissions:
    `https://classroom.github.com/a/aFVjykKU`

Python

We're going to do our programming in Python, so you'll need to have Python installed on your computer

- Use version ≥ 3.6
- Beware of Python 2
- Windows: Anaconda
- Use Homebrew or MacPorts to install Python3
- Linux: Install Python3 from you package manager.

Python

There are three ways we'll use Python.

- Interactive shell
- Invoke the interpreter
- JupyterLab
- While we're here: pip and Pipenv

The Python Language

- Developed 30 years ago by Guido van Rossum
- Interpreted, dynamically typed
- Core philosophy
  - Beautiful is better than ugly
  - Explicit is better than implicit
  - Simple is better than complex
  - Complex is better than complicated
  - Redability counts

Python Style

Python has clear style guidelines.

- PEP 8 `https://www.python.org/dev/peps/pep-0008/`
- Key points:
  - Code blocks are designated by indentation. Indents should be 4 spaces.
  - Class names are `CamelCased`
  - Variable names are `lower`, or `snake_cased`.

Programming Activity

1. Install Python, Git if necessary
2. Clone the course materials repo.
3. Click the GitHub classroom link for the practicals to set up your repo.
4. Clone/initialise that repo on your machine.
5. Add a `README` and a `.gitignore` (Python) to your repo.
6. Add, commit, and push to your repo.
7. Create a new branch, `01-practical` in your repo.
8. Add a subdirectory, `01-practical` and copy `01-practical.ipynb` from the class materials into it.
9. Open a shell, cd to this directory, and run `jupyter lab` to open the notebook. Complete the first question.
10. Add, commit, and push your new files.
11. On the GitHub page for your repo, create a pull request for this commit. Identify *tclark* as the reviewer.

OOP Review: Access

Access modifiers - Public
Class members are public by default.

```
class Cat:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def speak(self):
          return f'Meow, my name is {self.name}'

 if __name__ == '__main__':
     persian = Cat('Tom', 'persian')
     persian.name = 'Jerry'
     print(persian.speak())
```

OOP Review: Access

Access modifiers - Private/Protected
"Private" members can be identified with a leading underscore.

```
class Cat:
    def __init__(self, name, breed):
        self._name = name
        self._breed = breed

    def speak(self):
          return f'Meow, my name is {self._name}'
```

We can do this with fields and methods. The interpreter doesn't enforce any access restrictions. The notation merely tells programmers to treat the members as private.

OOP Review: Encapsulation

We can easily add setters and getters with the `@property` decorator.

```
class Cat:
    def __init__(self, name, breed):
        self._name = name
        self._breed = breed

    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, name):
        self._name = name

    if __name__ == '__main__':
        persian = Cat('Tom', 'persian')
        persian.name = 'Jerry'
        print(persian.name)
```

OOP Review: Inheritance

It is possible to extend a base class with a child class.

```
class Employee:
    def __init__(self, first_name, last_name, salary):
        self.first_name = first_name
        self.last_name = last_name
        self.salary = salary

    def __str__(self):
        return f'{self.first_name} {self.last_name}'

class SoftwareDeveloper(Employee):
    def __init__(self, first_name, last_name,
                 salary, prog_lang):
        super().__init__(first_name, last_name, salary)
        self.prog_lang = prog_lang
```

OOP Review: Polymorphism

```
class Country:
    def capital(self):
        raise NotImplementedError

class NewZealand(Country):
    def capital(self):
        return 'Wellington is the capital of New Zealand.'

class Brazil(Country):
    def capital(self):
        return 'Brasilia is the capital of Brazil.'

class Canada(Country):
    pass
```

OOP Review: Polymorphism

```
nzl = NewZealand()
bra = Brazil()
can = Canada()
for country in (nzl, bra, can):
    print(country.capital())
```

OOP Review: Polymorphism

Duck Typing

```
class NewZealand:
    def capital(self):
        return 'Wellington is the capital of New Zealand.'

class Brazil:
    def capital(self):
        return 'Brasilia is the capital of Brazil.'

class Canada:
   pass

nzl = NewZealand()
bra = Brazil()
can = Canada()
for country in (nzl, bra, can):
    print(country.capital())
```