CloudSkills.io

Week 2: Scripting Like A Developer

Michael Levan

@TheNJDevOpsGuy

linkedin.com/in/michaellevan

Course Outline

- **Lecture**: Core developer theory for DevOps and how to code like a developer
- **Projects**:
  - Setting up a dev environment
  - Reusable and clean Python code
  - Reusable and clean PowerShell code
  - Linting in PowerShell and Python
  - Testing in PowerShell and Python
  - Documenting code

Billing Management

- AWS Billing
- Azure Billing

Setting up Azure CLI and AWS CLI

- Demo in terminal

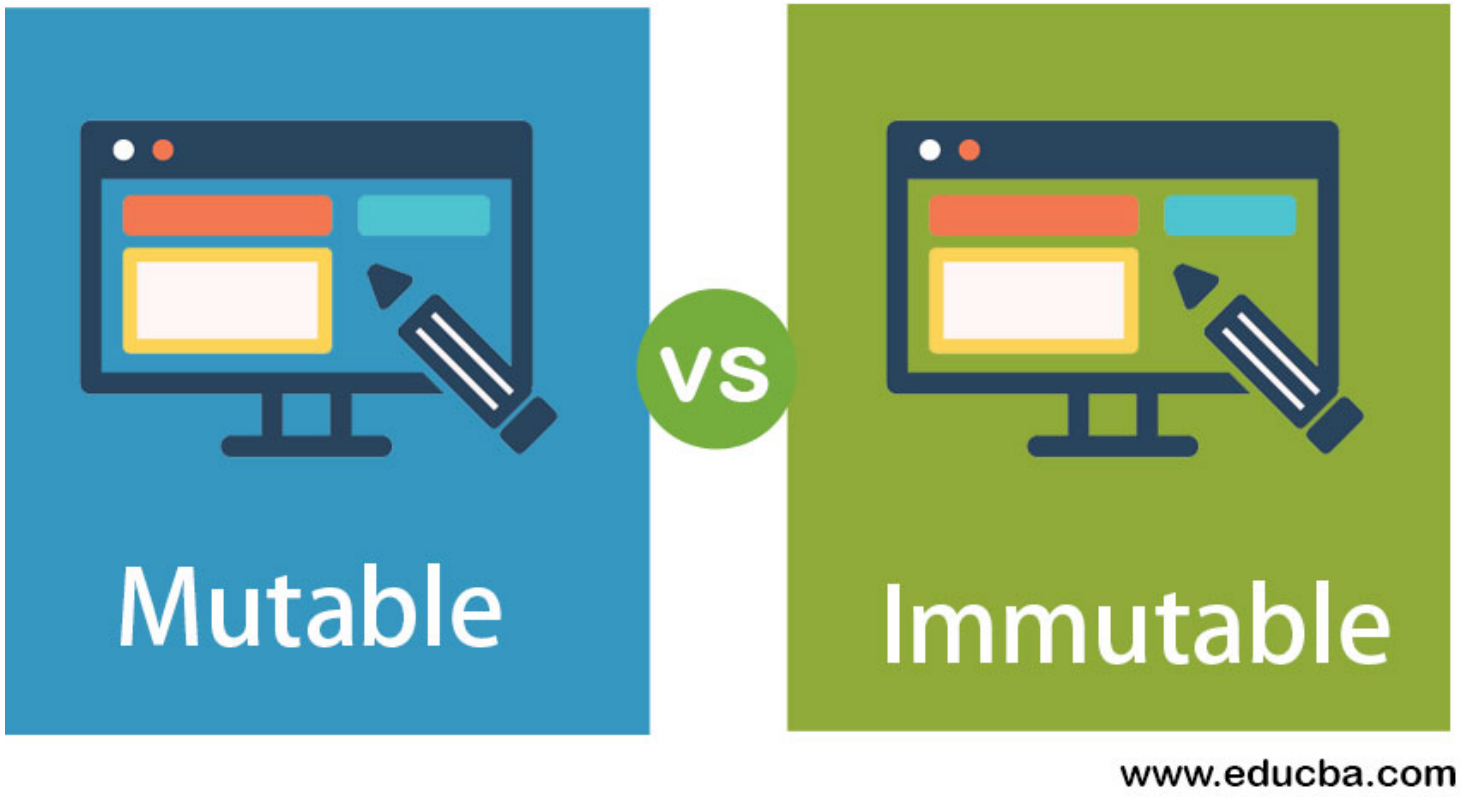
Developer Theory

- Immutable
- Mutable
- Declarative
- Imperative
- Procedural programming
- Object Oriented (OOP)
- Functional programming
- Idempotence
- Regression testing

Developer Theory (cont)

- Unit testing
- Mock testing

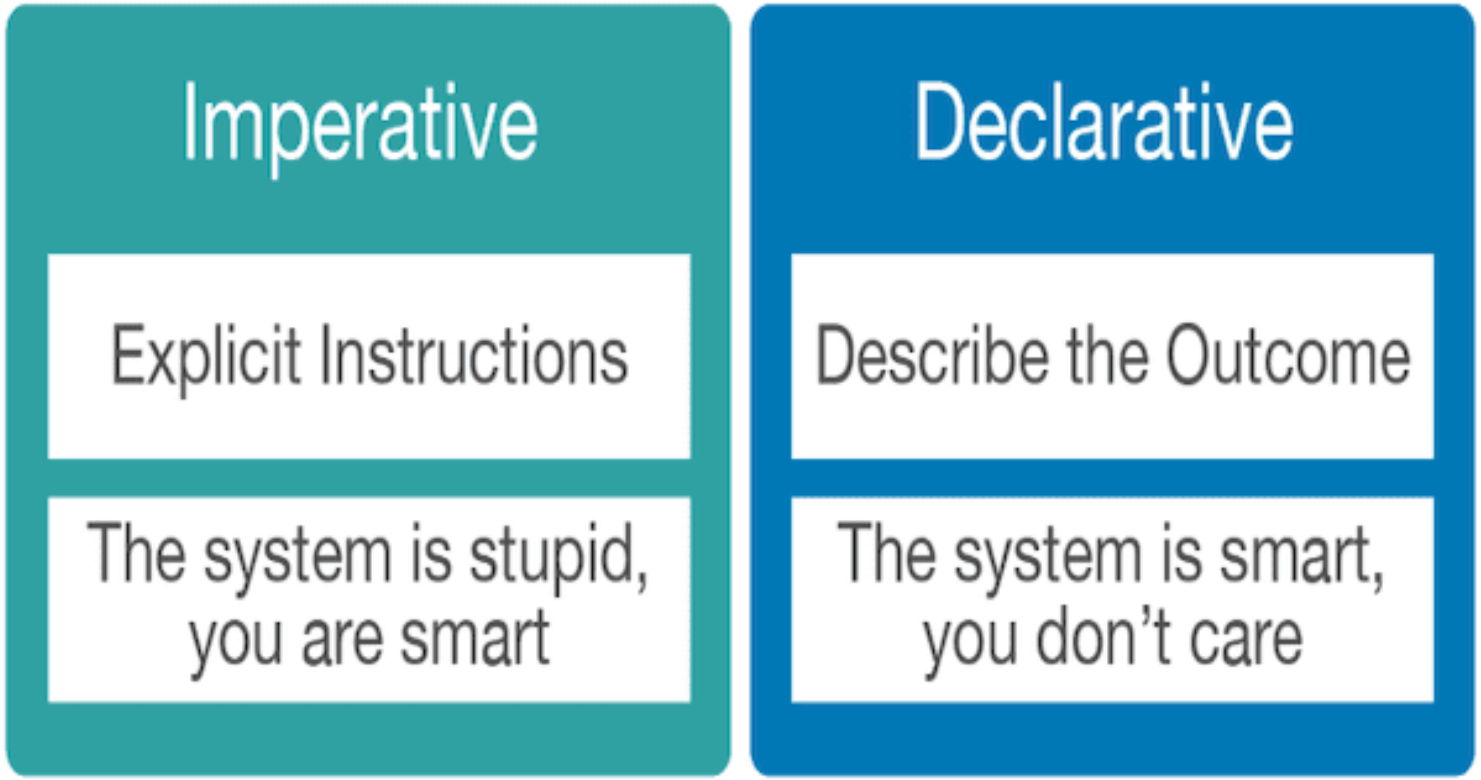
Immutable and Mutable



https://cdn.educba.com/academy/wp-content/uploads/2019/12/Mutable-vs-Immutable.jpg

- Immutable == You can start of change the code
- Mutable == Change the code all you want and the environment stays the same

Declarative and Imperative



https://www.copebit.ch/wp-content/uploads/Screenshot-2020-02-28-at-12.56.56-2-600x400-1.png

- Declarative == Tell me what to do, not how to do it (Terraform, CloudFormation, Go)
- Imperative == Tell me how to do it (Java, C#, C++)
- Python is an example of a language that's both declarative and imperative

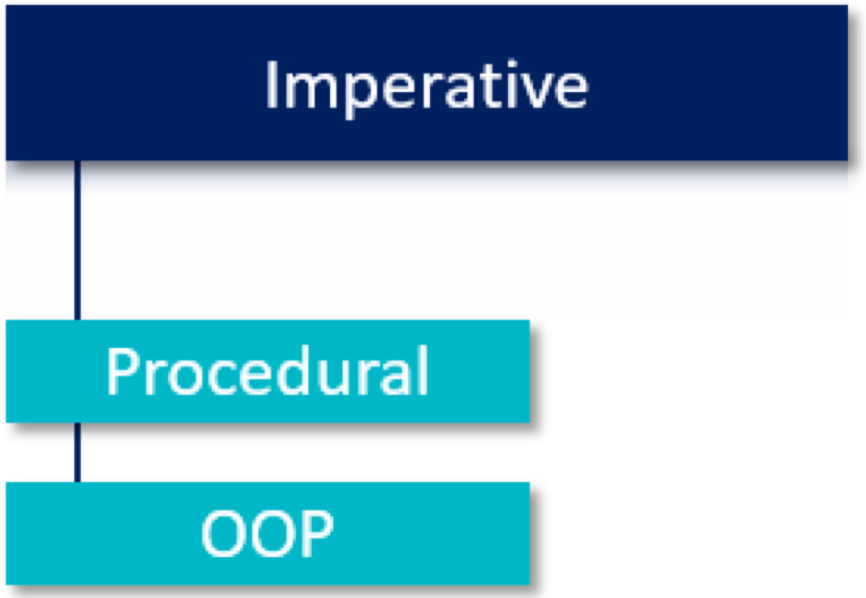
Imperative and Declarative

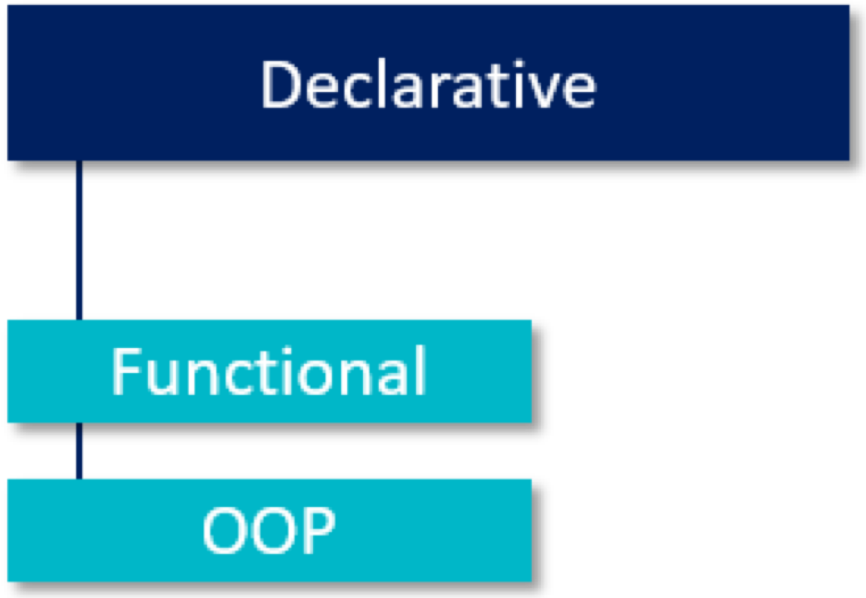
```
if (this.classList.contains('red')) {
  this.classList.remove('red');
  this.classList.add('blue');
} else {
  this.classList.remove('blue');
  this.classList.add('red');
}
```

```
this.state = { color: 'red' }

handleChange = () => {
  const color = this.state.color === 'red' ? 'blue' : 'red';
  this.setState({ color });
}
```

OOP, Functional, and Procedural

- Object Oriented Programming == Defining classes, objects of those classes, and methods
- Functional Programming == Like a mathematical function
- Procedural == A set of steps





Programming Styles

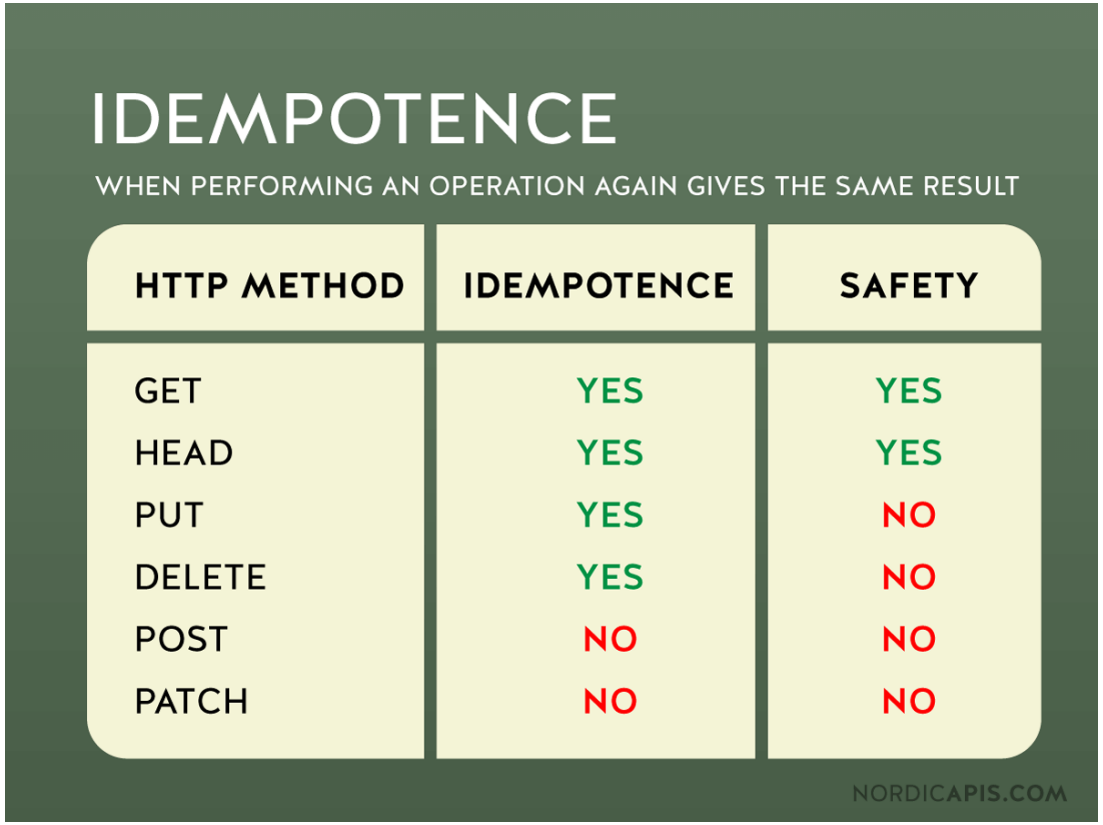
```
dog = ["Bulldog", "Husky", "Pitbull"]

def dogs(words):
    for i in range(len(words)):
        word = words[i]
        print(word)

dogs(dog)
```

```
class Dog:
    def __init__(self, breed, age):
        self.breed = breed
        self.age = age

dogsbreed = Dog("bulldog", 1)

print(dogsbreed.breed)
print(dogsbreed.age)
```

```
def add(x, y):
    print(x + y)

add(2, 4)4
```

Idempotence

- Make the same call without the result changing
- Making multiple, identical requests has the same effect as a single request



https://www.itprotoday.com/sites/itprotoday.com/files/styles/article_featured_retina/public/ThinkstockPhotos-490920800_0.jpg?itok=kQbb2T29

Testing Code

- Unit Tests
- Mock Tests
- Regression Tests