Ally Financial Training Questions

•••

Jacob Cantrell

DevOps

Can you explain your understanding of DevOps and how it differs from traditional software development methodologies?

DevOps

- software development approach
- emphasizes collaboration and communication b/w development & operations
- automate many processes involved b/w the two

DevOps differs from traditional software development methodologies

- Development & operations teams working more closely
- relying heavily on automation,
- more reliant on automation & prioritizing CI/CD.

How do you use tools such as Git, Jenkins, and Ansible to automate the software development and deployment process?

	Git	Jenkins	Ansible
Description	Distributed version control system	Open-source automation server	Automated deployment
How?	Version control & collaboration	Changes are pushed to Git repo	Deployment of new app versions
What?	Source code	Building, testing, & deployment	Provisioning, config, & deployment

Can you walk me through a recent project you managed using DevOps practices and the challenges you faced?

In a recent DevOps project, a team used Git for version control & collaboration, Jenkins for CI/CD, and Ansible for infrastructure as code.

One challenge faced was ensuring the CI/CD pipeline was fast, reliable, and consistent. To overcome this, automated testing & code quality checks at every stage of the pipeline can be implemented. Automatic rollback mechanism in case of failures, which helped minimize downtime and reduce risk of human error, can also be set up.

Another challenge was ensuring the security of infrastructure and applications. To overcome this, security measures such as encryption, network security, and continuous monitoring & logging of infrastructure and applications can be implemented.

How do you use containerization and orchestration technologies, such as Docker & Kubernetes, to improve scalability and reliability?

Docker & Kubernetes improve scalability and reliability by:

1. Easy Deployment - Containers can be easily deployed on any system, making it easy to scale application horizontally by adding more containers.
2. Improved Resource Utilization - Containers share the host system's resources, allowing multiple containers to run on a single host.
3. High Availability - Kubernetes provides automatic failover & self-healing to ensure apps continue to run even if a host or container fails
4. Environment Consistency - Containers provide a consistent environment, making it easy to move apps from development to production
5. Automated Scaling - Kubernetes can automatically scale apps based on demand

Can you explain how you use monitoring and logging tools, such as Prometheus & ELK, to improve visibility & troubleshoot issues in production environment?

Prometheus & ELK improve visibility & troubleshoot issues by:

1. Real-time Monitoring - Prometheus provides real-time monitoring & alerting making it easy to detect & respond to issues in a timely manner
2. Root Cause Analysis - Prometheus provides numerous performance metrics that can identify the root cause of an issue
3. Improved Collaboration - ELK provides centralized location for log data, making it easy for multiple teams to collaborate on troubleshooting
4. Automated Alerting - Prometheus provides automated alerting, making it easy to set up alerts for specific conditions

How do you handle & manage infrastructure as code using tools such as Terraform or CloudFormation?

Terraform & CloudFormation help handle & manage IaC in the following ways:

1. Version Control - they use version control systems to store & manage configuration files that define the infrastructure
2. Repeatable & Automated Deployments - provide a declarative syntax that allows developers to describe desired state of their infrastructure
3. Infrastructure Testing - allow devs to write test to validate the infrastructure before it is deployed to production
4. Infrastructure Management - provides centralized location for managing infrastructure, making it easy to view & manage the entire infrastructure
5. Cost Optimization - provide way to manage infrastructure costs by automating provisioning & scaling of resources

Can you discuss your experience with CI/CD pipelines?

CI/CD pipelines:

- Improve speed, quality, & reliability
- Reduce risk of human error
- Make it easier to track changes
- Provide source code management

Personally:

- Used SonarQube to automatically test the building of source code
- Used Jenkins to automate the building, testing, and deployment to GitHub
- Used Git & GitHub for version control & source code management

How do you use tools, such as JIRA or Trello, to manage & track work in a DevOps environment?

Both manage & track in a DevOps environment:

- Create Agile boards for projects
- Add tasks/issues and assign them
- Organize them w/ labels
- View reports on sprints
- Can be integrated with other tools (i.e. GitHub) to automate workflows & streamline processes

Personally used JIRA to (on Capstone Project w/ JUMP):

- Used Kanban board to view tasks & manage tasks
- Updated progress as tasks were completed or started
- Assigned tasks/issues to members of the team

Can you explain how you use A/B testing and canary releases to minimize risk and improve the release process?

	A/B	Canary
Goal	See which variation is best	Validate new software before official release
Target	Small subset of users	Small subset of users
Prevents?	Risk of releasing a faulty product	Resolves issues before release to broader user base

Python

What is Python & what are its features?

Python:

- dynamic , open-source, interpreted programming language
- Supports OOP & procedure-oriented programming

Features:

- Free & open-sourced
- Easy to learn
- High-level language
- Dynamically-typed
- OOP
- Large standard library
- Cross-platform language

Explain the difference b/w Python 2 & Python 3

Python 2	Python 3
Stores strings in ASCII (byte code)	Stores Strings in Unicode
Function params can have default values	Params w/ defaults must come after those w/o them
‘/’ can performs integer division	‘/’ always performs floating point division
‘Except’ can catch multiple exceptions	Each exception must have its own ‘except’ block
xrange()	range()

What are the built-in data types in Python?

int	Whole number
float	Any number with decimals
list	Sequence of objects (mutable)
tuple	Sequence of objects (immutable)

dict	Keys mapped to values
bool	True or False (or None)
str	Sequence of characters enclosed by “” or ‘’
set	Unique list of objects

What is the difference between a tuple & a list?

Tuple is **immutable** - values cannot be changed

List is **mutable** - values can be changed

Both are sequences of objects, not defined by one type in particular

How do you create a dictionary in Python?

With curly brackets { & }

Keys (i.e. int 1 or str ‘key1’ etc.) have to be mapped to a value (i.e. int 1 or str ‘value1’ etc.)

What is a generator in Python?

A generator is:

- Function that allows creation of an overall more efficient iterator
- Uses ‘yield’ instead of ‘return’
- Can be paused & resumed at any time
- Can generate sequence of values on the fly

What are the different types of loops and when would you see each one?

FOR	WHILE
Iterates over sequence	Iterates as long as a condition is met
Used when known number of iterations	Used when unknown number of iterations

Whats is a decorator in Python?

A decorator is:

- Function that takes another function
- Extends behavior of latter function w/o explicitly modifying it
- Wraps input function in another function and adds additional functionality

How do you handle exceptions in Python?

Exceptions are handles by:

- Using **try** and **except** blocks
- In the try block is where the code will go that <u>might</u> raise an exception
- In the except block, if the specified ExceptionType is caught, then the block is executed
- i.e.

```
try:
    #code
except ExceptionType:
    # code to handle exception
```

What is the difference between a module & a package?

Module	Package
Single file containing Python code	Collection of Modules in a directory hierarchy
A file ending in ‘.py’ (i.e. mymodule.py)	A directory containing a ‘__init__.py’ file

What is PEP 8 & why is it important?

- Document that provides guidelines & best practices for writing Python code

- Important b/c:
  - Makes code more readable & understandable
  - Ensures consistency
  - Makes code easier to integrate
  - Easier to identify & correct potential issues

How do you debug Python code?

- Print statements
- Using pdb
  - Interactive debugger that needs to be imported and called
- Using an IDE
- External libraries
  - pudb
  - ipdb
- Unit testing

How would you reverse a string in Python?

Slicing - s[::-1]

What are advantages of using NumPy over regular Python lists?

- Faster computation
- Less memory usage
- Broadcasting
    - can perform operations on arrays of different types & sizes
- Vectorization
    - can perform operations on entire arrays at once
- Comprehensive mathematical functions
    - Linear algebra
    - Fourier transforms
    - Random number generator

What are the different types of inheritance in Python?

- **Single** - derives properties from single parent
- **Multiple** - can be derived from more than one base class
- **Multi-Level** - single inheritance with another single inheritance (i.e. child inherits father and grandfather)
- **Hierarchical** - when more than one class is created from a single base class
- **Hybrid** - multiple forms of inheritance at once

How do you implement a linked-list in Python?

Using classes, for example:

- A class for the LinkedList itself
  - Contains init & head (Node) attribute
  - Functions to add Node
- A class for a Node in the LinkedList
  - Contains init, and data (value) & next (Node) attributes

What is the difference b/w a deep & shallow copy in Python?

In **shallow**, a copy is made (not stored) and only the reference address is copied, and only the original is stored

In **deep**, both the copy and the original are stored

What are benefits of using a virtual environment in Python?

- Isolation
  - Isolate environment from other installments & projects on system
- Dependency management
  - Can install packages specific to project
- Reproducibility
  - Others can reproduce same environment w/ same packages & dependencies
- Ease of deployment
  - Can create virtual environment on deployment server to easily transition project
- Security
  - Reduce risk of security vulnerabilities

Explain difference between *args and **kwargs

***args** is a non-keyword associated arbitrary list or arguments

- i.e. [1, ‘Python’, 3, 123.456]

****kwargs** is keyword-associated arbitrary dictionary of arguments, where arguments are associated with a keyword

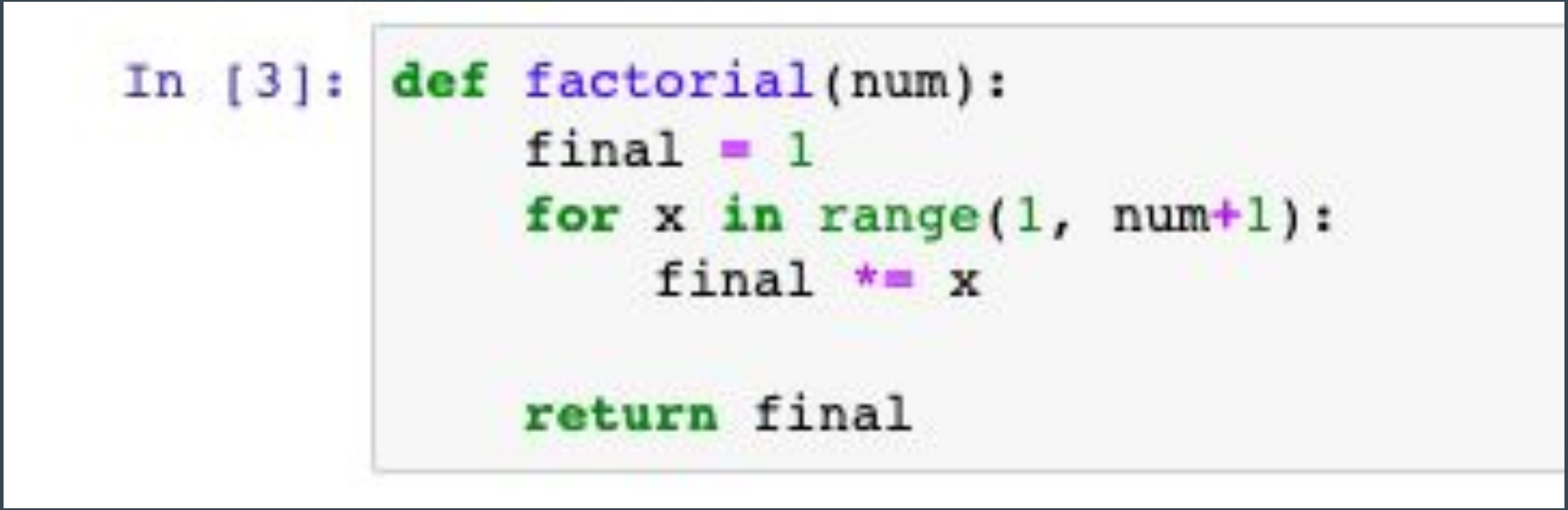
- i.e. { ‘nums’: [1,2,3], ‘fruit’: [‘pear’, ‘apple’, ‘banana’] }

How to check if a string is a palindrome in Python?

Without spaces, if the string is the same as the reverse string

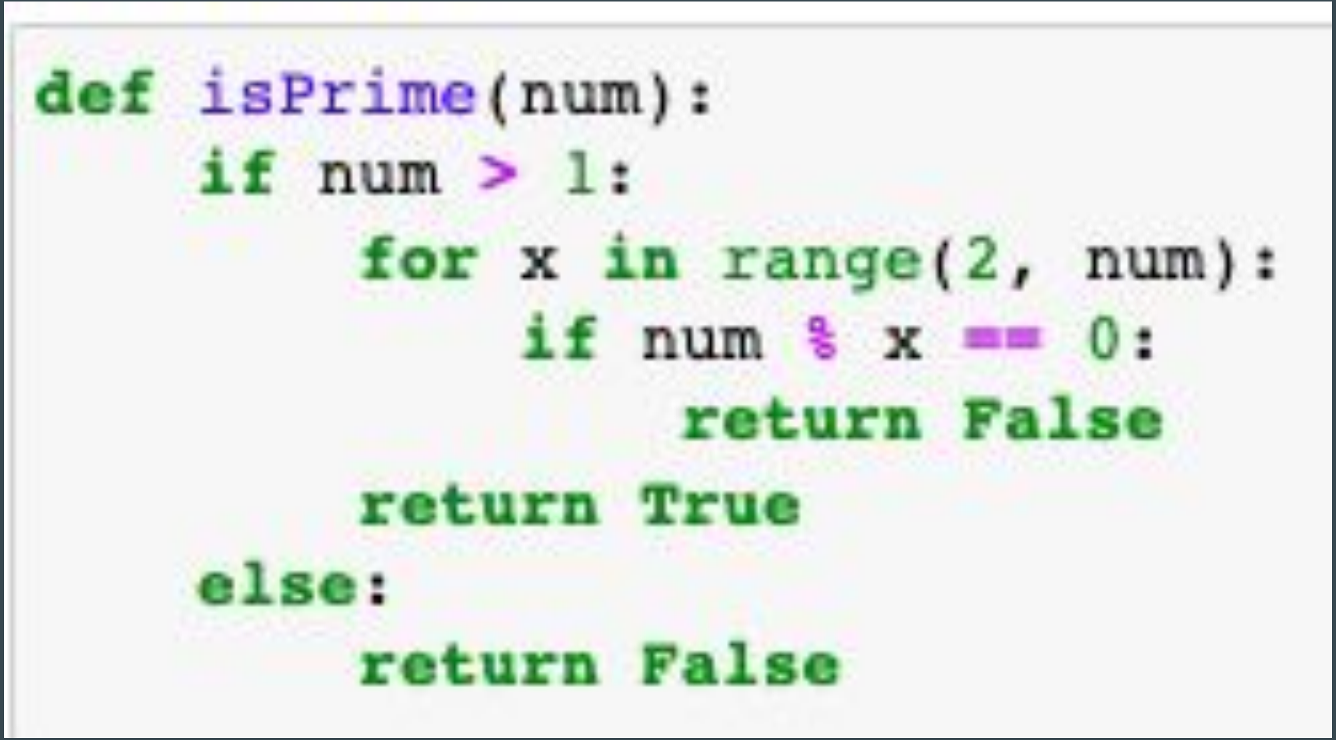
Python Scripts

Python Function - Factorial of a Number



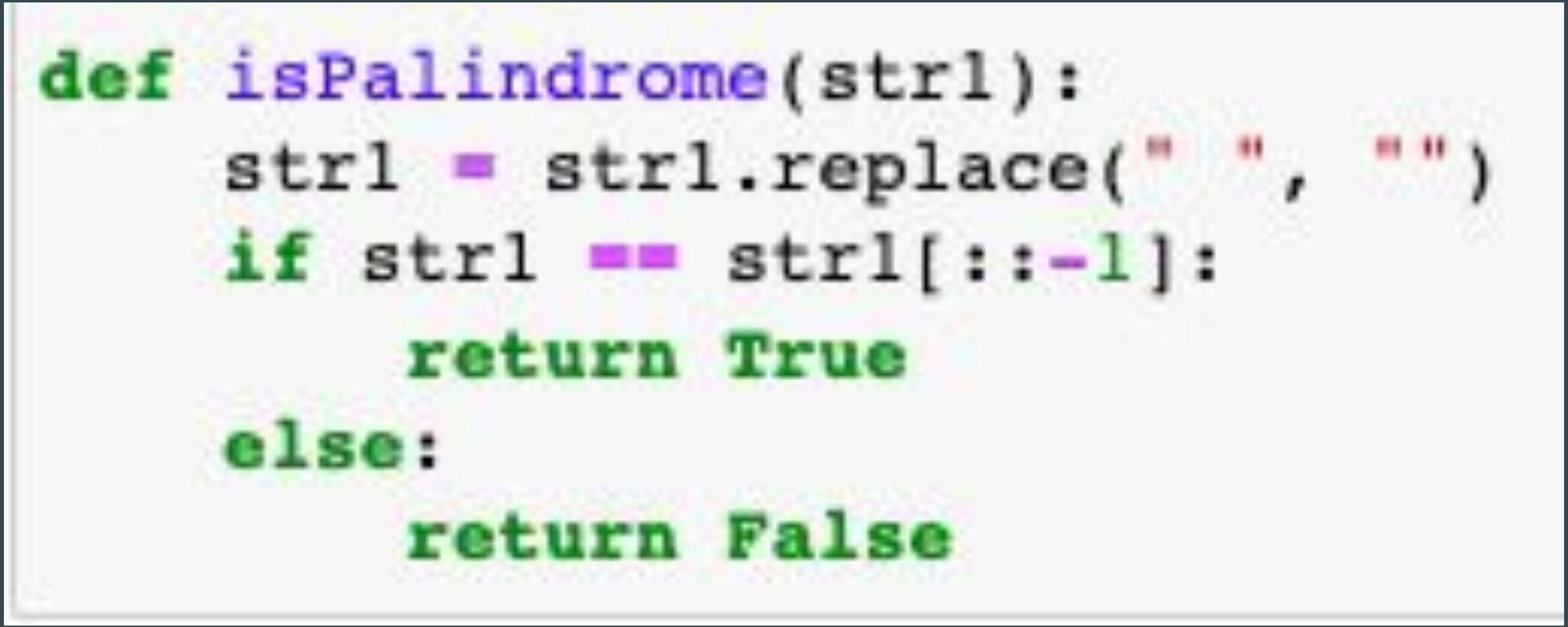
```
In [3]: def factorial(num):
            final = 1
            for x in range(1, num+1):
                final *= x

            return final
```

Python Function - Check if Prime



```python
def isPrime(num):
    if num > 1:
        for x in range(2, num):
            if num % x == 0:
                return False
        return True
    else:
        return False
```

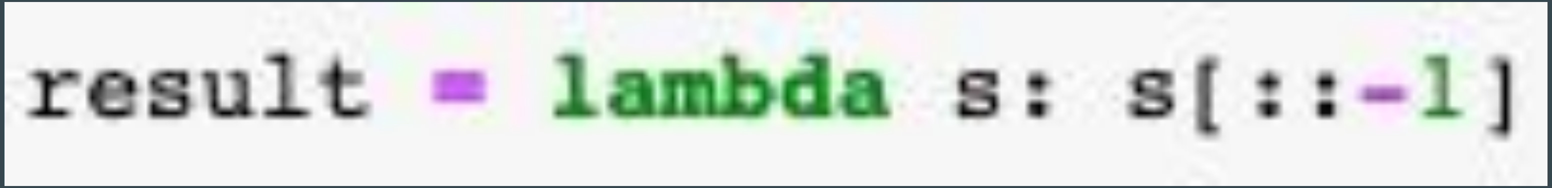
Python Function - String is Palindrome



```python
def isPalindrome(str1):
    str1 = str1.replace(" ", "")
    if str1 == str1[::-1]:
        return True
    else:
        return False
```

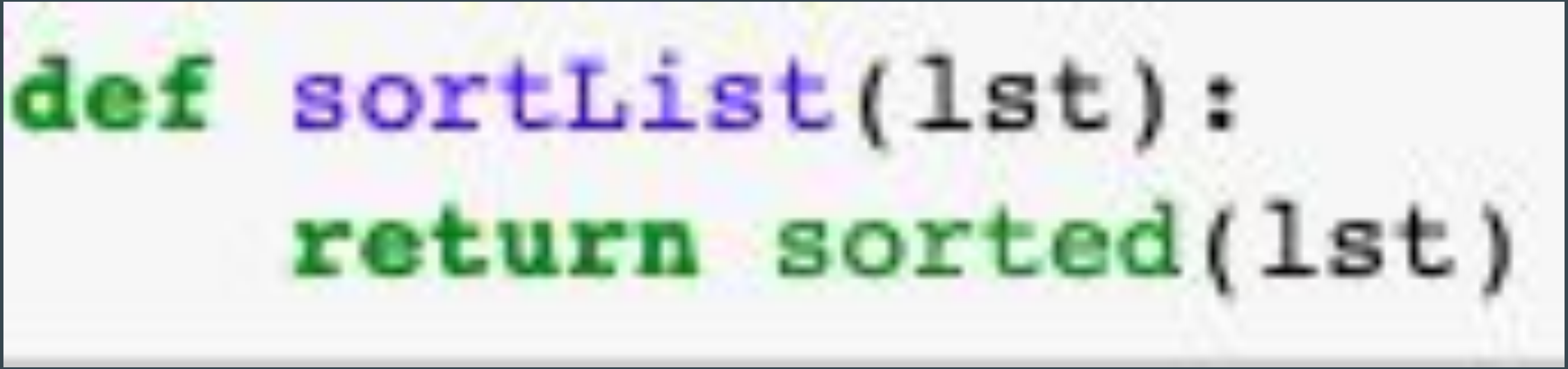
Python Function - Two Strings Are Anagrams

Python String - Reverse String



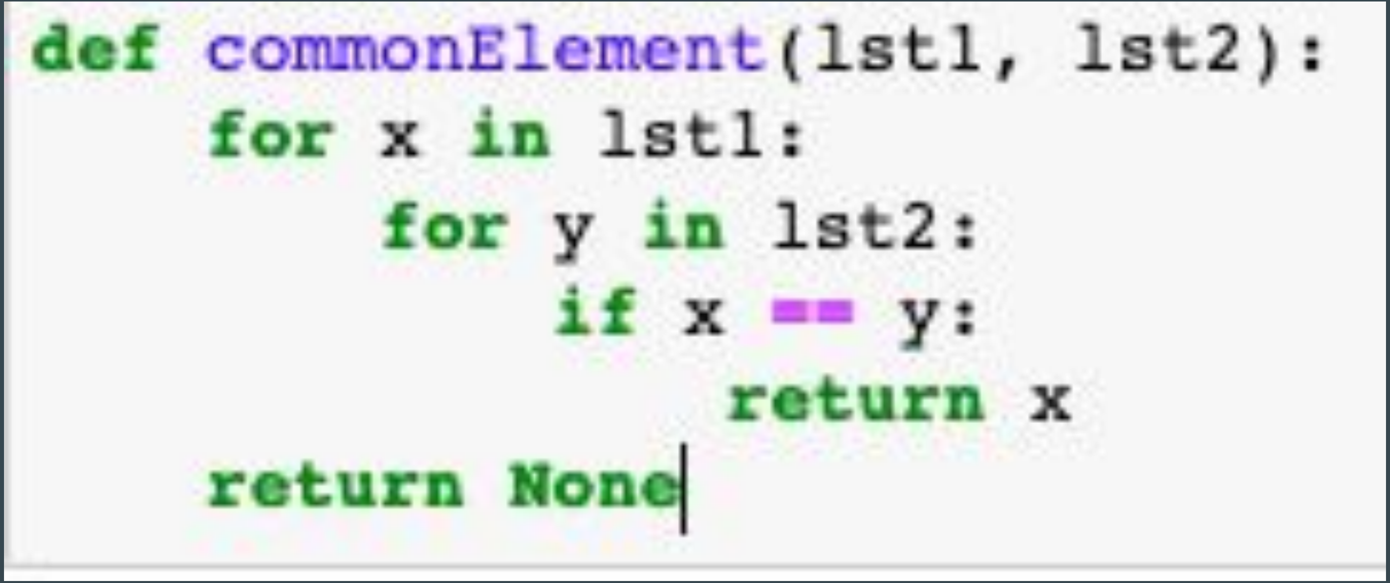
```
result = lambda s: s[::-1]
```

Python Functions - Sort List



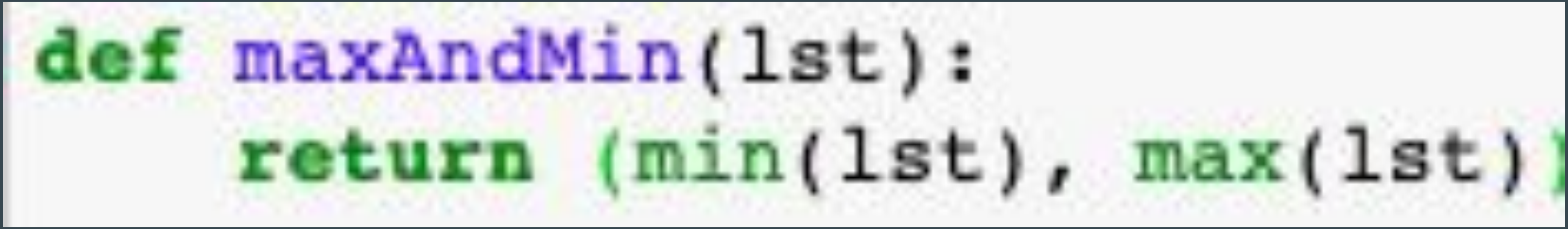
```
def sortList(lst):
    return sorted(lst)
```

Python Function - Common Element b/w Lists



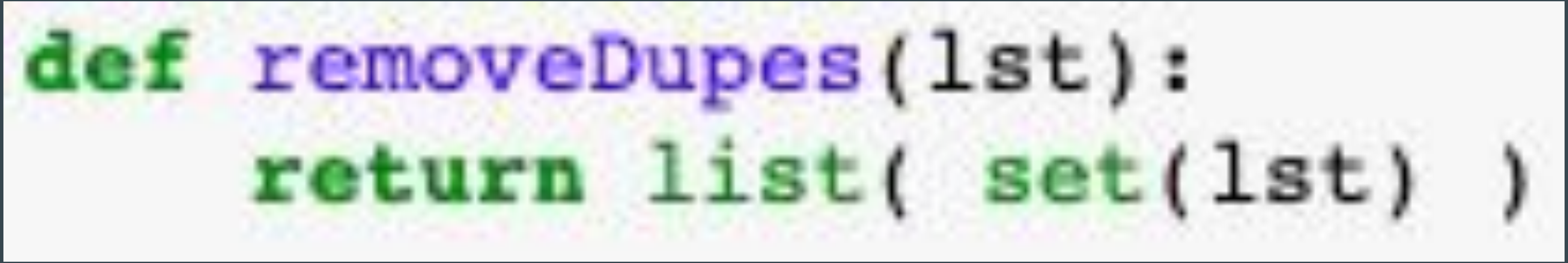
```
def commonElement(lst1, lst2):
    for x in lst1:
        for y in lst2:
            if x == y:
                return x
    return None
```

Python Functions - Max & Min in List



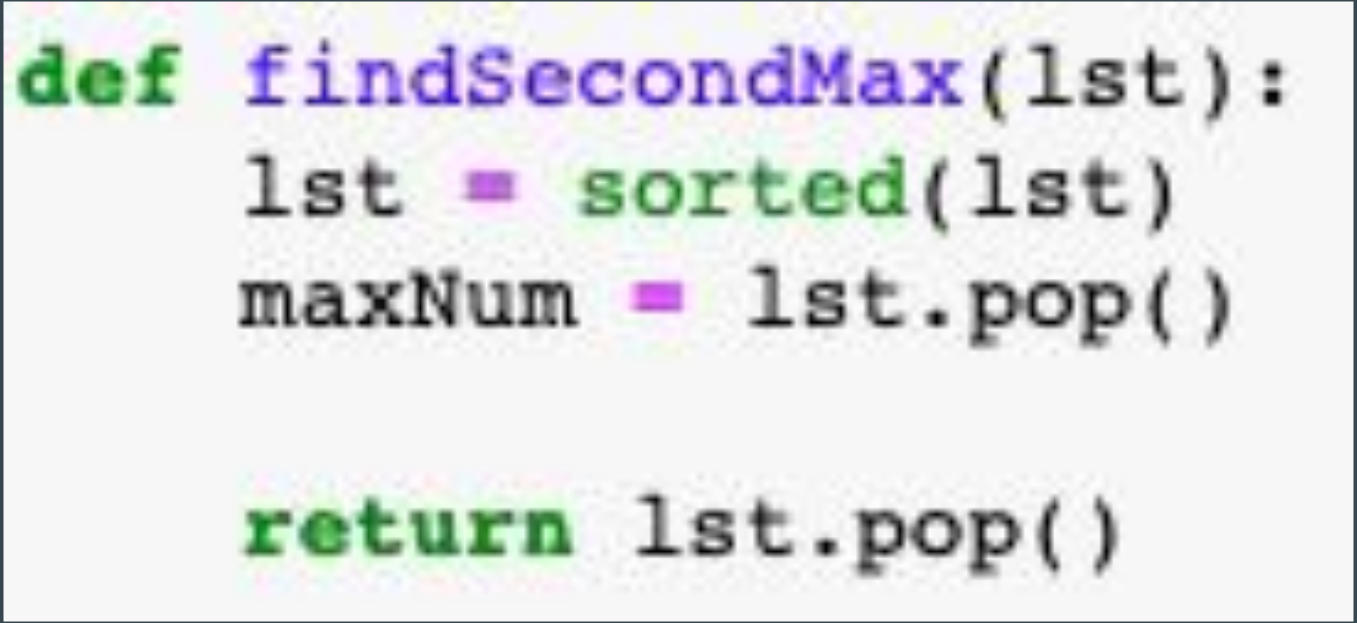
```
def maxAndMin(lst):
    return (min(lst), max(lst)
```

Python Function - Remove Duplicates From List



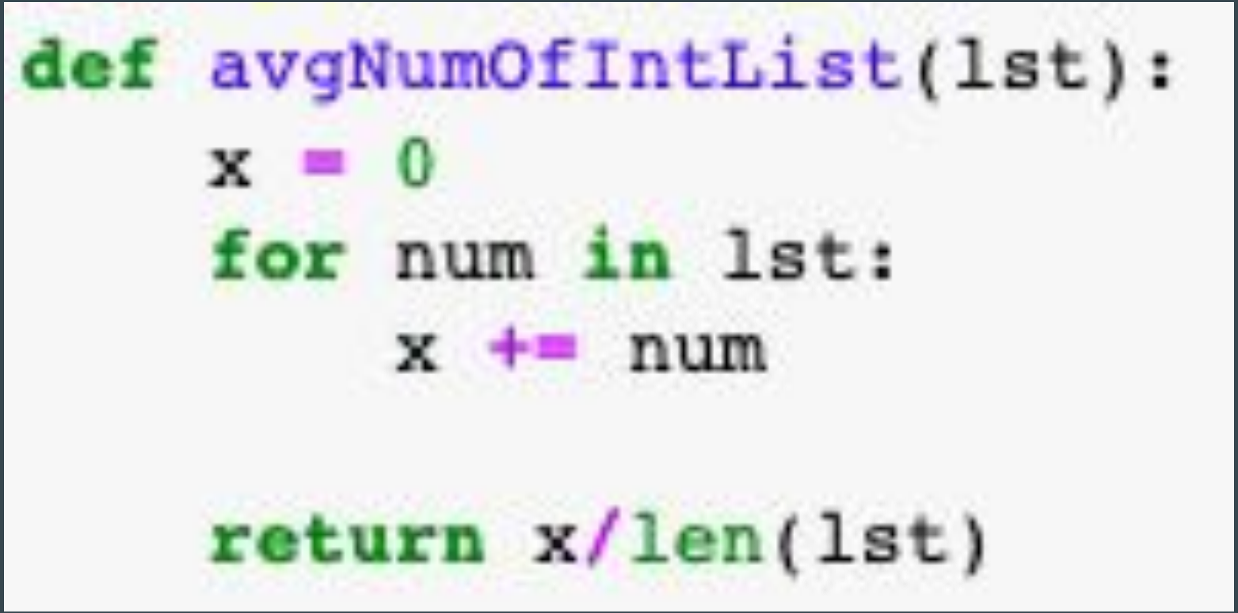
```
def removeDupes(lst):
    return list( set(lst) )
```

Python Function - Second Largest Num In List



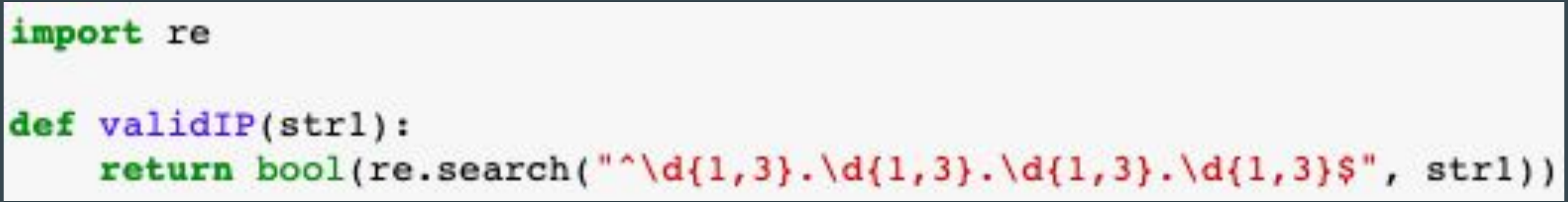
```python
def findSecondMax(lst):
    lst = sorted(lst)
    maxNum = lst.pop()

    return lst.pop()
```

Python Function - Average Num Of Number List



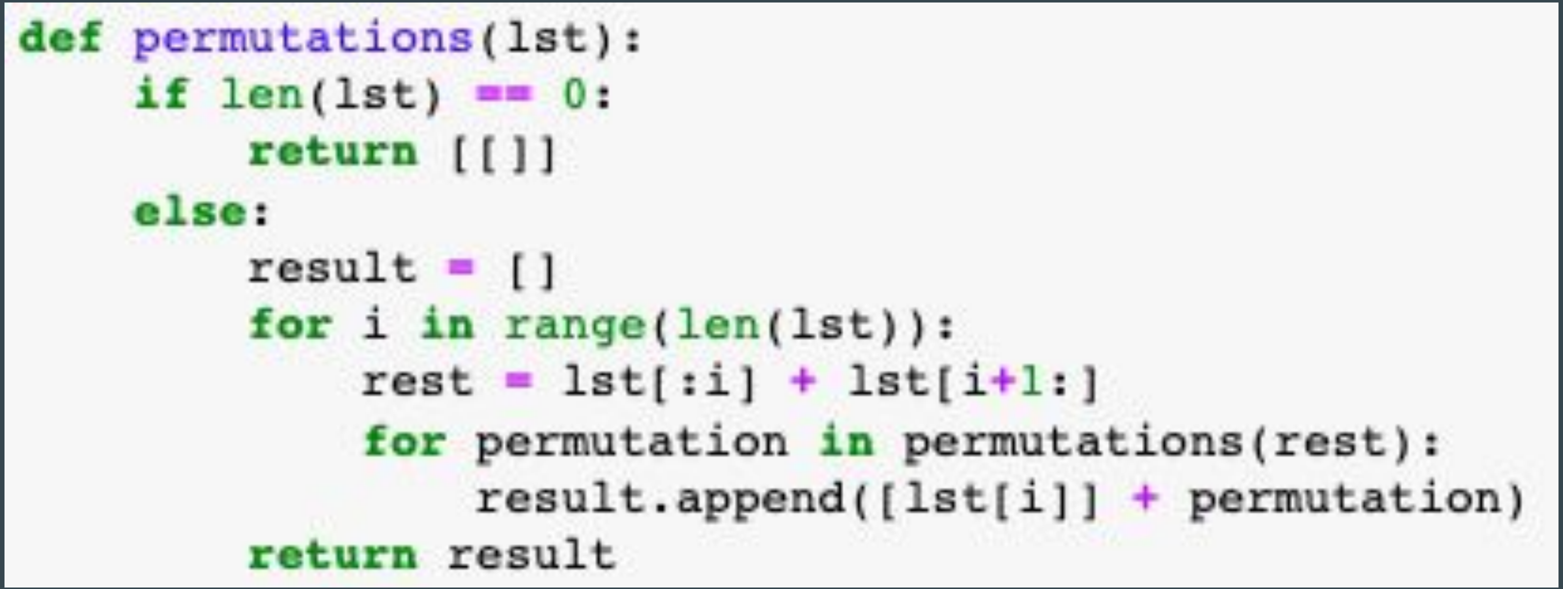
```python
def avgNumOfIntList(lst):
    x = 0
    for num in lst:
        x += num

    return x/len(lst)
```

Python Function - String is Valid IP



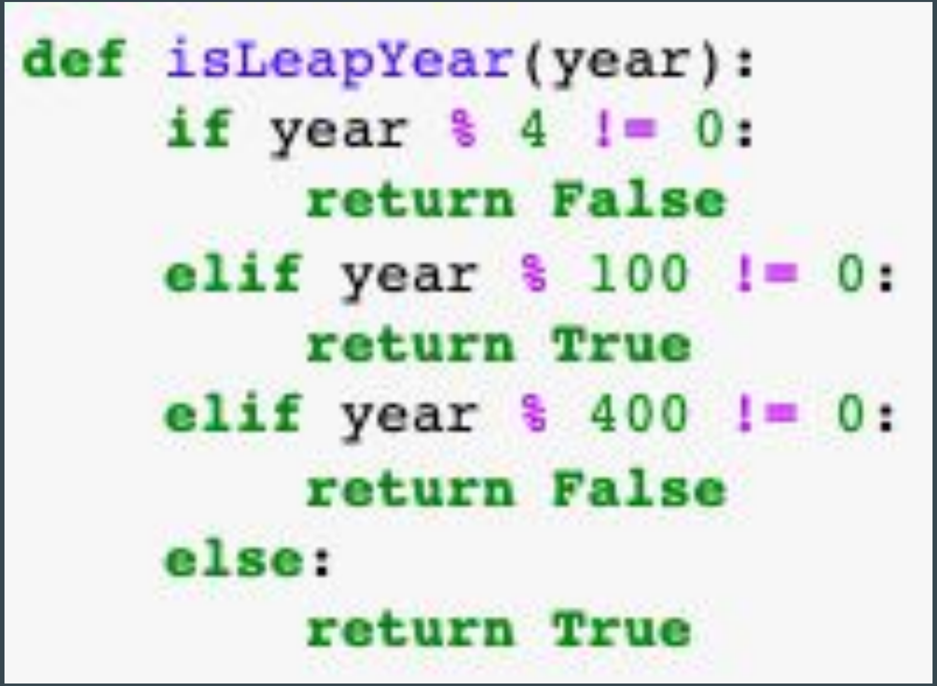
```
import re

def validIP(strl):
    return bool(re.search("^\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}$", strl))
```

Python Function - Generate All Permutations of List



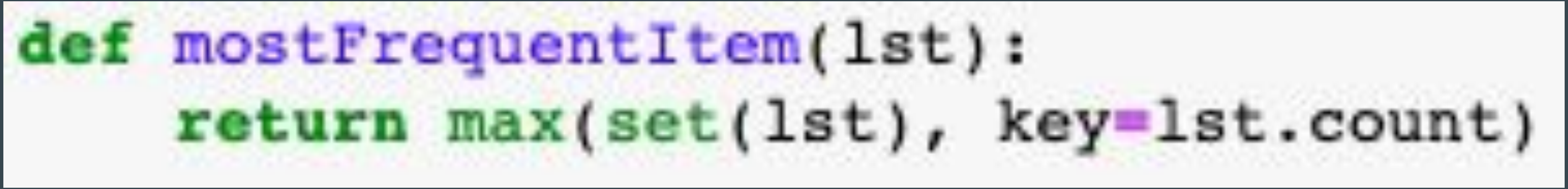
```python
def permutations(lst):
    if len(lst) == 0:
        return [[]]
    else:
        result = []
        for i in range(len(lst)):
            rest = lst[:i] + lst[i+1:]
            for permutation in permutations(rest):
                result.append([lst[i]] + permutation)
        return result
```

Python Function - Given Year is Leap Year



```
def isLeapYear(year):
    if year % 4 != 0:
        return False
    elif year % 100 != 0:
        return True
    elif year % 400 != 0:
        return False
    else:
        return True
```

Python Function - Most Frequent Element in List



```python
def mostFrequentItem(lst):
    return max(set(lst), key=lst.count)
```