1. File Operations

Read and write files with context manager.

```
# Read a file
with open('file.txt', 'r') as file:
    content = file.read()
    print(content)

# Write to a file
with open('output.txt', 'w') as file:
    file.write('Hello, DevOps!')
```

2. Environment Variables

Get and set environment variables via os.

```
import os

# Get an env var
db_user = os.getenv('DB_USER')
print(db_user)

# Set an env var
os.environ['NEW_VAR'] = 'value'
```

3. Subprocess Automation

Run shell commands and capture output.

```
import subprocess

result = subprocess.run(["ls", "-l"], capture_output=True, text=True)
print(result.stdout)
```

4. API Requests

Simple GET request using requests.

```
import requests

response = requests.get('https://api.example.com/data')
print(response.json())
```

5. JSON Handling

Load and dump JSON from/to files.

```
import json

# Read JSON from a file
with open('data.json', 'r') as file:
    data = json.load(file)
    print(data)

# Write JSON to a file
data = {'name': 'DevOps', 'type': 'Workflow'}
with open('output.json', 'w') as file:
    json.dump(data, file, indent=4)
```

6. Logging

Basic logging setup and an info message.

```
import logging

logging.basicConfig(level=logging.INFO)
logging.info('This is an informational message')
```

7. Working with Databases (SQLite)

Create/connect to SQLite DB and ensure table.
```
import sqlite3

conn = sqlite3.connect('example.db')
cursor = conn.cursor()
cursor.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)')
conn.commit()
conn.close()
```

8. Automation with Libraries (Paramiko)

SSH into a host and run a command.
```
import paramiko

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect('hostname', username='user', password='password')

stdin, stdout, stderr = ssh.exec_command('ls')
print(stdout.read().decode())
ssh.close()
```

9. Error Handling

Wrap risky code in try/except.
```
try:
    # code that may raise an exception
    risky_code()
except Exception as e:
    print(f'Error occurred: {e}')
```

10. Docker Integration

List running containers with docker SDK.
```
import docker

client = docker.from_env()
containers = client.containers.list()
for container in containers:
    print(container.name, container.status)
```

11. Working with YAML Files

Read and write YAML using PyYAML.
```
import yaml

# Read YAML
with open('config.yaml', 'r') as file:
    config = yaml.safe_load(file)
    print(config)

# Write YAML
data = {'name': 'DevOps', 'version': '1.0'}
with open('output.yaml', 'w') as file:
    yaml.dump(data, file)
```

12. Parsing Command-Line Arguments

Use argparse to parse options.
```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('--num', type=int, help='an integer for the accumulator')
args = parser.parse_args()
print(args.num)
```

13. Monitoring System Resources

Quick CPU and memory usage with psutil.

```
import psutil

print(f"CPU Usage: {psutil.cpu_percent()}%")
print(f"Memory Usage: {psutil.virtual_memory().percent}%")
```

14. Handling HTTP Requests with Flask

Tiny Flask app exposing a /health endpoint.

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/health', methods=['GET'])
def health_check():
    return jsonify({'status': 'healthy'})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

15. Creating Docker Containers

Run a short-lived Ubuntu container and read logs.

```
import docker

client = docker.from_env()
container = client.containers.run('ubuntu', 'echo Hello World', detach=True)
print(container.logs())
```

16. Scheduling Tasks

Run a function every minute with schedule.

```
import schedule
import time

def job():
    print("Running scheduled job...")

schedule.every(1).minutes.do(job)

while True:
    schedule.run_pending()
    time.sleep(1)
```

17. Version Control with Git (GitPython)

Add + commit a file.

```
import git

repo = git.Repo('/path/to/repo')
repo.git.add('file.txt')
repo.index.commit('Added file.txt')
```

18. Email Notifications

Send a plain-text email via SMTP.

```
import smtplib
from email.mime.text import MIMEText

msg = MIMEText('This is the body of the email')
msg['Subject'] = 'Email Subject'
msg['From'] = 'you@example.com'
msg['To'] = 'recipient@example.com'
with smtplib.SMTP('smtp.example.com', 587) as server:
```

```
    server.starttls()
    server.login('your_username', 'your_password')
    server.send_message(msg)
```

19. Creating Virtual Environments

Create and (shell) activate a venv.

```
import subprocess

# Create virtual environment
subprocess.run(['python3', '-m', 'venv', 'myenv'])

# NOTE: Activation happens in the shell, not inside Python.
# Windows: myenv\Scripts\activate
# Linux/Mac: source myenv/bin/activate
```

20. Integrating with CI/CD Tools (Jenkins trigger)

Trigger a Jenkins build with token.

```
import requests

url = "http://your-jenkins-url/job/your-job-name/build"
response = requests.post(url, auth=('user', 'token'))
print(response.status_code)
```

21. Database Migration (Alembic CLI)

Initialize and upgrade DB schema.

```
# Run in shell:
# alembic revision -m "initial migration"
# alembic upgrade head
```

22. Testing Code (unittest)

Simple unittest example.

```
import unittest

def add(a, b):
    return a + b

class TestMathFunctions(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(2, 3), 5)

if __name__ == '__main__':
    unittest.main()
```

23. Data Transformation with Pandas

Read CSV, add column, write CSV.

```
import pandas as pd

df = pd.read_csv('data.csv')
df['new_column'] = df['existing_column'] * 2
df.to_csv('output.csv', index=False)
```

24. Python for IaC (boto3 EC2)

List running EC2 instances.

```
import boto3

ec2 = boto3.resource('ec2')
instances = ec2.instances.filter(Filters=[{'Name': 'instance-state-name', 'Values': ['running']}])
for instance in instances:
    print(instance.id, instance.state)
```

25. Web Scraping (BeautifulSoup)

Fetch a page and print its title.

```
import requests
from bs4 import BeautifulSoup

response = requests.get('http://example.com')
soup = BeautifulSoup(response.content, 'html.parser')
print(soup.title.string)
```

26. Remote Execution (Fabric)

Run a command over SSH with Fabric.

```
from fabric import Connection

conn = Connection(host='user@hostname', connect_kwargs={'password': 'your_password'})
conn.run('uname -s')
```

27. Automating AWS S3 Operations

Upload and download S3 files with boto3.

```
import boto3

s3 = boto3.client('s3')
s3.upload_file('local_file.txt', 'bucket_name', 's3_file.txt')
s3.download_file('bucket_name', 's3_file.txt', 'local_file.txt')
```

28. Monitoring Application Logs

tail -f equivalent in Python.

```
import time

def tail_f(file):
    file.seek(0, 2)
    while True:
        line = file.readline()
        if not line:
            time.sleep(0.1)
            continue
        print(line, end='')

with open('app.log') as log_file:
    tail_f(log_file)
```

29. Container Health Checks

Inspect Docker container health status.

```
import docker

client = docker.from_env()
container = client.containers.get('container_id')
print(container.attrs['State']['Health']['Status'])
```

30. Rate-Limited APIs (requests)

Handle HTTP 429 with a wait + retry.

```
import requests, time

url = 'https://api.example.com/data'
while True:
    r = requests.get(url)
    if r.status_code == 200:
        print(r.json())
        break
```

```
    elif r.status_code == 429:
        time.sleep(60)
    else:
        print('Error:', r.status_code)
        break
```

31. Docker Compose Integration

Start/stop services with docker-compose.

```
import subprocess

subprocess.run(['docker-compose', 'up', '-d'])
subprocess.run(['docker-compose', 'down'])
```

32. Terraform Execution

Run terraform init/apply from Python.

```
import subprocess

subprocess.run(['terraform', 'init'])
subprocess.run(['terraform', 'apply', '-auto-approve'])
```

33. Prometheus Metrics

Fetch metrics endpoint and print lines.

```
import requests

resp = requests.get('http://localhost:9090/metrics')
for line in resp.text.splitlines():
    print(line)
```

34. pytest for Testing

Minimal pytest test function.

```
def add(a, b):
    return a + b

def test_add():
    assert add(2, 3) == 5
```

35. Creating Webhooks (Flask)

POST /webhook endpoint and printer.

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/webhook', methods=['POST'])
def webhook():
    data = request.json
    print("Received data:", data)
    return 'OK', 200

if __name__ == '__main__':
    app.run(port=5000)
```

36. Jinja2 for Config Templates

Render a simple template.

```
from jinja2 import Template

template = Template("Hello, {{ name }}!")
print(template.render(name='DevOps'))
```

37. Encrypting and Decrypting Data

Use cryptography.fernet for symmetric crypto.

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()
cipher = Fernet(key)

encrypted = cipher.encrypt(b'Secret Data')
decrypted = cipher.decrypt(encrypted)
print(decrypted.decode())
```

38. Error Monitoring with Sentry

Capture an exception in Sentry.

```
import sentry_sdk

sentry_sdk.init("your_sentry_dsn")

def divide(a, b):
    return a / b

try:
    divide(1, 0)
except ZeroDivisionError as e:
    sentry_sdk.capture_exception(e)
```

39. GitHub Actions (workflow yaml)

Basic CI workflow (YAML).

```
# .github/workflows/ci.yml
name: CI
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.8'
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run tests
        run: pytest
```

40. Creating a Simple API with FastAPI

Async GET endpoint with uvicorn runner.

```
from fastapi import FastAPI

app = FastAPI()

@app.get('/items/{item_id}')
async def read_item(item_id: int):
    return {'item_id': item_id}

if __name__ == '__main__':
    import uvicorn
    uvicorn.run(app, host='0.0.0.0', port=8000)
```

41. Log Aggregation with Elasticsearch

Index a simple log document.

```
from elasticsearch import Elasticsearch

es = Elasticsearch(['http://localhost:9200'])
```

```
doc = {"message": "Hello, ELK!"}
es.index(index="logs", document=doc)
```

42. pandas for ETL

Extract, transform, load CSV.

```
import pandas as pd

data = pd.read_csv('source.csv')
data['new_column'] = data['existing_column'].apply(lambda x: x * 2)
data.to_csv('destination.csv', index=False)
```

43. AWS Lambda (Hello)

Minimal Lambda handler.

```
import json

def lambda_handler(event, context):
    return {'statusCode': 200, 'body': json.dumps('Hello from Lambda!')}
```

44. Working with Redis

Set and get keys via redis-py.

```
import redis

r = redis.StrictRedis(host="localhost", port=6379, db=0)
r.set('foo', 'bar')
print(r.get('foo'))
```

45. Tunneling with pyngrok

Expose local port publicly.

```
from pyngrok import ngrok

public_url = ngrok.connect(5000)
print("Public URL:", public_url)
input("Press Enter to exit...")
```

46. Flask-RESTful API

Hello World resource.

```
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

47. asyncio Basics

Await and sleep example.

```
import asyncio

async def main():
    print("Hello")
    await asyncio.sleep(1)
    print("World")
```

```
asyncio.run(main())
```

48. Network Monitoring with scapy

Sniff packets and print summaries.

```
from scapy.all import sniff

def packet_callback(packet):
    print(packet.summary())

sniff(prn=packet_callback, count=10)
```

49. ConfigParser (INI files)

Read and write INI config.

```
import configparser

config = configparser.ConfigParser()
config.read('config.ini')
print(config['DEFAULT'].get('SomeSetting', 'N/A'))

config['DEFAULT']['NewSetting'] = 'Value'
with open('config.ini', 'w') as f:
    config.write(f)
```

50. WebSocket Client

Connect to echo server and print messages.

```
import websocket

def on_message(ws, message):
    print("Received:", message)

ws = websocket.WebSocketApp("ws://echo.websocket.org", on_message=on_message)
ws.run_forever()
```

51. Build Docker Image (docker SDK)

Build image from a Dockerfile string.

```
import docker

client = docker.from_env()
dockerfile = """
FROM python:3.9-slim
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
CMD ["python","app.py"]
"""
image, logs = client.images.build(fileobj=dockerfile.encode("utf-8"), tag="my-python-app")
for line in logs:
    print(line)
```

52. psutil for System Monitoring

CPU and RAM with interval.

```
import psutil

print("CPU Usage:", psutil.cpu_percent(interval=1), "%")
print("Memory Usage:", psutil.virtual_memory().percent, "%")
```

53. Alembic Programmatic Upgrade

Upgrade to head from Python.

```
from alembic import command, config
```

```
cfg = config.Config("alembic.ini")
command.upgrade(cfg, "head")
```

54. Paramiko SSH

Run 'ls -la' on remote host.

```
import paramiko

client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
client.connect('hostname', username='user', password='your_password')
stdin, stdout, stderr = client.exec_command('ls -la')
print(stdout.read().decode())
client.close()
```

55. CloudFormation Stack Creation

Create a stack from template.

```
import boto3

cf = boto3.client('cloudformation')
with open('template.yaml') as f:
    template_body = f.read()

resp = cf.create_stack(
    StackName='MyStack',
    TemplateBody=template_body,
    Parameters=[{'ParameterKey': 'InstanceType', 'ParameterValue': 't2.micro'}],
    TimeoutInMinutes=5,
    Capabilities=['CAPABILITY_NAMED_IAM']
)
print(resp)
```

56. EC2 Instance Management

Start/stop by instance id.

```
import boto3

ec2 = boto3.resource('ec2')
instance = ec2.Instance('i-xxxxxxxx')
# instance.start()
# instance.stop()
```

57. Automated Backup with shutil

Copy a directory tree.

```
import shutil

source_dir = '/path/to/source'
backup_dir = '/path/to/backup'
shutil.copytree(source_dir, backup_dir)
```

58. watchdog for FS Monitoring

Print on file modifications.

```
import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

class MyHandler(FileSystemEventHandler):
    def on_modified(self, event):
        print(f'File modified: {event.src_path}')

observer = Observer()
observer.schedule(MyHandler(), path='.', recursive=False)
observer.start()
try:
```

```
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    observer.stop()
observer.join()
```

59. Load Testing with locust

Define a simple HttpUser.

```
from locust import HttpUser, task, between

class MyUser(HttpUser):
    wait_time = between(1, 3)
    @task
    def root(self):
        self.client.get("/")
```

60. GitHub API (requests)

Get repo info with token header.

```
import requests
url = "https://api.github.com/repos/user/repo"
r = requests.get(url, headers={"Authorization": "token YOUR_GITHUB_TOKEN"})
print(r.json())
```

61. kubectl via subprocess

Get pods & apply manifest.

```
import subprocess
subprocess.run(['kubectl', 'get', 'pods'])
subprocess.run(['kubectl', 'apply', '-f', 'deployment.yaml'])
```

62. pytest in CI/CD

Run pytest from Python.

```
import subprocess
subprocess.run(['pytest'])
```

63. Simple CLI Tool (argparse)

Positional ints + --sum option.

```
import argparse
p = argparse.ArgumentParser(description='Process some integers.')
p.add_argument('integers', metavar='N', type=int, nargs='+', help='an integer to be processed')
p.add_argument('--sum', dest='accumulate', action='store_const', const=sum, default=max,
               help='sum the integers (default: find the max)')
args = p.parse_args()
print(args.accumulate(args.integers))
```

64. dotenv for Env Vars

Load .env into environment.

```
from dotenv import load_dotenv
import os

load_dotenv()
print(os.getenv('DATABASE_URL'))
```

65. BeautifulSoup Scraper

Print all H1 texts.

```
import requests
```

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(requests.get('http://example.com').text, 'html.parser')
for h in soup.find_all('h1'):
    print(h.text)
```

66. PyYAML Config Files

Load config and dump to output.

```
import yaml
with open('config.yaml') as f:
    cfg = yaml.safe_load(f)
print(cfg)
with open('output.yaml', 'w') as f:
    yaml.dump(cfg, f)
```

67. RabbitMQ (pika)

Send and receive a message.

```
import pika

# Send
conn = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
ch = conn.channel()
ch.queue_declare(queue='hello')
ch.basic_publish(exchange='', routing_key='hello', body='Hello World!')
conn.close()

# Receive
def callback(ch, method, properties, body):
    print("Received:", body)
conn = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
ch = conn.channel()
ch.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)
ch.start_consuming()
```

68. sentry_sdk for Monitoring

Minimal init + capture.

```
import sentry_sdk
sentry_sdk.init("YOUR_SENTRY_DSN")
try:
    1/0
except Exception as e:
    sentry_sdk.capture_exception(e)
```

69. openpyxl Excel Files

Create and read a cell.

```
from openpyxl import Workbook, load_workbook
wb = Workbook(); ws = wb.active
ws['A1'] = 'Hello'
wb.save('sample.xlsx')
wb = load_workbook('sample.xlsx'); ws = wb.active
print(ws['A1'].value)
```

70. SQLAlchemy ORM

Define model and insert a row.

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

Base = declarative_base()

class User(Base):
    __tablename__ = 'users'
```

```
    id = Column(Integer, primary_key=True)
    name = Column(String)

engine = create_engine('sqlite:///example.db')
Base.metadata.create_all(engine)
Session = sessionmaker(bind=engine); session = Session()
session.add(User(name='Alice')); session.commit()
```

71. Monitor Docker Containers

List names + status.

```
import docker
client = docker.from_env()
for c in client.containers.list():
    print(c.name, c.status)
```

72. Flask Simple API

Return a small JSON payload.

```
from flask import Flask, jsonify
app = Flask(__name__)

@app.route('/api/data', methods=['GET'])
def get_data():
    return jsonify({"message": "Hello, World!"})

if __name__ == '__main__':
    app.run(debug=True)
```

73. Renew certbot Certificates

Run certbot renew.

```
import subprocess
subprocess.run(['certbot', 'renew'])
```

74. NumPy Basics

Compute a mean from an array.

```
import numpy as np
data = np.array([1,2,3,4,5])
print("Mean Value:", data.mean())
```

75. Send Email (SMTP)

Send a text email.

```
import smtplib
from email.mime.text import MIMEText
msg = MIMEText("This is a test email.")
msg['Subject'] = "Test Email"; msg['From'] = "you@example.com"; msg['To'] = "recipient@example.com"
with smtplib.SMTP("smtp.example.com", 587) as s:
    s.starttls(); s.login("username","password"); s.send_message(msg)
```

76. schedule for Tasks

Run job every 10 minutes.

```
import schedule, time
def job(): print("Job is running...")
schedule.every(10).minutes.do(job)
while True:
    schedule.run_pending(); time.sleep(1)
```

77. matplotlib Plot

Simple XY line plot.
```
import matplotlib.pyplot as plt
x=[1,2,3,4,5]; y=[2,3,5,7,11]
plt.plot(x,y); plt.xlabel('X-axis'); plt.ylabel('Y-axis'); plt.title('Simple Plot'); plt.show()
```

78. Package Setup (setuptools)

Minimal setup.py example.
```
from setuptools import setup, find_packages
setup(name='my_package', version='0.1', packages=find_packages(), install_requires=['requests','flask'])
```

79. pytest Unit Test

Simple add() test.
```
def add(a,b): return a+b
def test_add(): assert add(1,2)==3
```

80. requests-oauthlib (OAuth1)

Call API with OAuth1Session.
```
from requests_oauthlib import OAuth1Session
oauth = OAuth1Session(client_key="YOUR_CLIENT_KEY", client_secret="YOUR_CLIENT_SECRET")
print(oauth.get("https://api.example.com/user").json())
```

81. pandas Data Manipulation

Read CSV and filter rows.
```
import pandas as pd
df = pd.read_csv('data.csv'); print(df.head())
filtered = df[df['column_name']>10]; print(filtered)
```

82. requests GET/POST

Basic REST calls.
```
import requests
print(requests.get('https://api.example.com/data').json())
print(requests.post('https://api.example.com/data', json={'key':'value'}).json())
```

83. http.server

Serve current directory.
```
from http.server import SimpleHTTPRequestHandler, HTTPServer
PORT=8000
with HTTPServer(('',PORT), SimpleHTTPRequestHandler) as httpd:
    print(f"Serving on port {PORT}"); httpd.serve_forever()
```

84. Flask Webhook Receiver

POST /webhook to log body.
```
from flask import Flask, request
app = Flask(__name__)
@app.route('/webhook', methods=['POST'])
def webhook(): print(request.json); return '', 200
if __name__=='__main__': app.run(port=5000)
```

85. subprocess to Run Bash

Echo from Python.
```
import subprocess
subprocess.run(['echo','Hello, World!'])
```

86. docker-compose via Python

Start services detached.
```
import subprocess
subprocess.run(['docker-compose','up','-d'])
```

87. moto Mock S3 for Tests

Create a bucket and upload.
```
import boto3
from moto import mock_s3

@mock_s3
def test_s3_upload():
    s3=boto3.client('s3', region_name='us-east-1')
    s3.create_bucket(Bucket='my-bucket')
    # s3.upload_file('file.txt','my-bucket','file.txt')
```

88. asyncio Concurrency

Run a coroutine.
```
import asyncio
async def say_hello(): print("Hello"); await asyncio.sleep(1); print("World")
asyncio.run(say_hello())
```

89. flask-cors

Enable CORS on Flask app.
```
from flask import Flask
from flask_cors import CORS
app = Flask(__name__); CORS(app)
@app.route('/data')
def data(): return {"message":"Hello from CORS!"}
if __name__=='__main__': app.run()
```

90. pytest Fixtures

Fixture with setup/teardown.
```
import pytest
@pytest.fixture
def sample_data():
    data={"key":"value"}
    yield data  # teardown after yield if needed
def test_sample_data(sample_data):
    assert sample_data['key']=='value'
```

91. http.client Low-Level

HTTPS GET and print status.
```
import http.client
conn=http.client.HTTPSConnection("www.example.com")
conn.request("GET","/"); resp=conn.getresponse()
print(resp.status, resp.reason); print(resp.read()); conn.close()
```

92. Redis Caching

Set/get and decode value.
```
import redis
r=redis.StrictRedis(host='localhost', port=6379, db=0)
r.set('key','value'); print(r.get('key').decode('utf-8'))
```

93. json Serialization

dict → JSON string.

```
import json
print(json.dumps({"name":"John","age":30}))
```

94. xml.etree.ElementTree

Parse XML and iterate children.

```
import xml.etree.ElementTree as ET
root = ET.parse('data.xml').getroot()
for child in root: print(child.tag, child.attrib)
```

95. venv Programmatic Create

Create venv with pip.

```
import venv
venv.create('myenv', with_pip=True)
```

96. psutil Memory

Show total and available.

```
import psutil
m=psutil.virtual_memory()
print(f"Total: {m.total}, Available: {m.available}")
```

97. sqlite3 Basics

Create table, insert, select.

```
import sqlite3
conn=sqlite3.connect('example.db'); c=conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)")
c.execute("INSERT INTO users (name) VALUES ('Alice')"); conn.commit()
for row in c.execute("SELECT * FROM users"): print(row)
conn.close()
```

98. pytest in Parallel (CLI)

Run tests with 4 workers.

```
# Shell command:
# pytest -n 4
```

99. argparse (sum or max)

Positional integers + --sum.

```
import argparse
p=argparse.ArgumentParser(description='Process some integers.')
p.add_argument('integers', metavar='N', type=int, nargs='+', help='an integer')
p.add_argument('--sum', dest='accumulate', action='store_const', const=sum, default=max,
               help='sum the integers (default: max)')
args=p.parse_args(); print(args.accumulate(args.integers))
```

100. jsonschema Validation

Validate JSON against a schema.

```
from jsonschema import validate
from jsonschema.exceptions import ValidationError

schema={"type":"object","properties":{"name":{"type":"string"},"age":{"type":"integer","minimum":0}}, "required":["
data={"name":"John","age":30}

try:
```

```
    validate(instance=data, schema=schema)
    print("Valid JSON")
except ValidationError as e:
    print("Invalid JSON:", e)
```