get into the world of Python with confidence, using this cheatsheet to streamline your coding tasks and enhance your Python proficiency.

> *Top 5 Must-read Computer Vision Books in 2024*
>
> *How To Become a Computer Vision Engineer*
>
> *Mastering Gradient Descent*
>
> *Fine-tuning RoBERTa for Sentiment Analysis*
>
> *Pandas Cheatsheet*

01 — *Data Types*

```
int_num = 42
float_num = 3.14
string_var = "Hello, Python!"
bool_var = True
```

02 — *Variables and Assignment*

```
x = 10
y = "Python"
```

03 — *Lists & Tuples*

```
my_list = [1, 2, 3, "Python"]
```

```
my_tuple = (1, 2, 3, "Tuple")
```

04 — *Dictionaries*

```
my_dict = {'name': 'John', 'age': 25, 'city': 'Pythonville'}
```

05 — *Control Flow*

```
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
for item in my_list:
    print(item)
while condition:
    # code
```

06 — *Functions*

```
def greet(name="User"):
    return f"Hello, {name}!"
result = greet("John")
```

07 — *Classes & Objects*

```
class Dog:
    def __init__(self, name):
        self.name = name
def bark(self):
        print("Woof!")
```

```
my_dog = Dog("Buddy")
my_dog.bark()
```

08 — *File Handling*

```
with open("file.txt", "r") as file:
    content = file.read()
with open("new_file.txt", "w") as new_file:
    new_file.write("Hello, Python!")
```

09 — *Exception Handling*

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero!")
finally:
    print("Execution completed.")
```

10 — *Libraries & Modules*

```
import math
from datetime import datetime
result = math.sqrt(25)
current_time = datetime.now()
```

11 — *List Comprehensions*

```
squares = [x**2 for x in range(5)]
```

12 — *Lambda Functions*

```
add = lambda x, y: x + y
result = add(2, 3)
```

13 — *Virtual Environment*

```
# Create a virtual environment
python -m venv myenv
# Activate the virtual environment
source myenv/bin/activate  # On Unix or MacOS
myenv\Scripts\activate  # On Windows
# Deactivate the virtual environment
deactivate
```

14 — *Package Management*

```
# Install a package
pip install package_name
# List installed packages
pip list
# Create requirements.txt
pip freeze > requirements.txt
# Install packages from requirements.txt
pip install -r requirements.txt
```

15 — *Working with JSON*

```
import json
# Convert Python object to JSON
json_data = json.dumps({"name": "John", "age": 25})
# Convert JSON to Python object
python_obj = json.loads(json_data)
```

16 — *Regular Expressions*

```
import re
pattern = r'\d+'  # Match one or more digits
result = re.findall(pattern, "There are 42 apples and 123 oranges.")
```

17 — *Working with Dates*

```
from datetime import datetime, timedelta
current_date = datetime.now()
future_date = current_date + timedelta(days=7)
```

18 — *List Manipulations*

```
numbers = [1, 2, 3, 4, 5]
# Filter
evens = list(filter(lambda x: x % 2 == 0, numbers))
# Map
squared = list(map(lambda x: x**2, numbers))
# Reduce (requires functools)
from functools import reduce
product = reduce(lambda x, y: x * y, numbers)
```

19 — *Dictionary Manipulations*

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
# Get value with default
value = my_dict.get('d', 0)
# Dictionary comprehension
squared_dict = {key: value**2 for key, value in my_dict.items()}
```

20 — *Concurrency with Threading*

```
import threading
def print_numbers():
```

```
    for i in range(5):
        print(i)
thread = threading.Thread(target=print_numbers)
thread.start()
```

21 — *Concurrency with Asyncio*

```
import asyncio
async def print_numbers():
    for i in range(5):
        print(i)
        await asyncio.sleep(1)
asyncio.run(print_numbers())
```

22 — *Web Scraping with Beautiful Soup*

```
from bs4 import BeautifulSoup
import requests
url = "https://example.com"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
title = soup.title.text
```

23 — *RESTful API with Flask*

```
from flask import Flask, jsonify, request
app = Flask(__name__)
@app.route('/api/data', methods=['GET'])
def get_data():
    data = {'key': 'value'}
    return jsonify(data)
if __name__ == '__main__':
    app.run(debug=True)
```

24 — *Unit Testing with unittest*

```
import unittest
def add(x, y):
    return x + y
class TestAddition(unittest.TestCase):
    def test_add_positive_numbers(self):
        self.assertEqual(add(2, 3), 5)
if __name__ == '__main__':
    unittest.main()
```

25 — *Database Interaction with SQLite*

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()
# Execute SQL query
cursor.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name
# Commit changes
conn.commit()
# Close connection
conn.close()
```

26 — *File Handling*

```
# Writing to a file
with open('example.txt', 'w') as file:
    file.write('Hello, Python!')
# Reading from a file
with open('example.txt', 'r') as file:
    content = file.read()
```

27 — *Error Handling*

```
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f"Unexpected Error: {e}")
```

```
else:
    print("No errors occurred.")
finally:
    print("This block always executes.")
```

28 — *Working with JSON*

```
import json
data = {'name': 'John', 'age': 30}
# Convert Python object to JSON
json_data = json.dumps(data)
# Convert JSON to Python object
python_object = json.loads(json_data)
```

29 — *Python Decorators*

```
def decorator(func):
    def wrapper():
        print("Before function execution")
        func()
        print("After function execution")
    return wrapper
@decorator
def my_function():
    print("Inside the function")
my_function()
```

30 — *Working with Enums*

```
from enum import Enum
class Color(Enum):
    RED = 1
    GREEN = 2
    BLUE = 3
print(Color.RED)
```

31 — *Working with Sets*

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
# Union
union_set = set1 | set2
# Intersection
intersection_set = set1 & set2
# Difference
difference_set = set1 - set2
```

32 — *List Comprehensions*

```
numbers = [1, 2, 3, 4, 5]
# Squares of even numbers
squares = [x**2 for x in numbers if x % 2 == 0]
```

33 — *Lambda Functions*

```
add = lambda x, y: x + y
result = add(3, 5)
```

34 — *Threading with Concurrent.futures*

```
from concurrent.futures import ThreadPoolExecutor
def square(x):
    return x**2
with ThreadPoolExecutor() as executor:
    results = executor.map(square, [1, 2, 3, 4, 5])
```

35 — *Internationalization (i18n) with gettext*

```
import gettext
# Set language
lang = 'en_US'
```

```
_ = gettext.translation('messages', localedir='locale', languages=[lang]).gette
print(_("Hello, World!"))
```

36 — *Virtual Environment*

```
# Create a virtual environment
python -m venv myenv
# Activate virtual environment
source myenv/bin/activate  # On Unix/Linux
myenv\Scripts\activate  # On Windows
# Deactivate virtual environment
deactivate
```

37 — *Working with Dates*

```
from datetime import datetime, timedelta
now = datetime.now()
# Format date
formatted_date = now.strftime('%Y-%m-%d %H:%M:%S')
# Add days to a date
future_date = now + timedelta(days=7)
```

38 — *Working with Dictionaries*

```
my_dict = {'name': 'John', 'age': 30}
# Get value with default
age = my_dict.get('age', 25)
# Iterate over keys and values
for key, value in my_dict.items():
    print(f"{key}: {value}")
```

39 — *Regular Expressions*

```
import re
text = "Hello, 123 World!"
```

```
# Match numbers
numbers = re.findall(r'\d+', text)
```

40 — *Working with Generators*

```
def square_numbers(n):
    for i in range(n):
        yield i**2
squares = square_numbers(5)
```

41 — *Database Interaction with SQLite*

```
import sqlite3
# Connect to SQLite database
conn = sqlite3.connect('mydatabase.db')
cursor = conn.cursor()
# Execute SQL query
cursor.execute('SELECT * FROM mytable')
```

42 — *Working with ZIP Files*

```
import zipfile
with zipfile.ZipFile('archive.zip', 'w') as myzip:
    myzip.write('file.txt')
with zipfile.ZipFile('archive.zip', 'r') as myzip:
    myzip.extractall('extracted')
```

43 — *Web Scraping with requests and BeautifulSoup*

```
import requests
from bs4 import BeautifulSoup
url = 'https://example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
```

```
# Extract data from HTML
title = soup.title.text
```

44 — *Sending Email with smtplib*

```
import smtplib
from email.mime.text import MIMEText
# Set up email server
server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
# Log in to email account
server.login('your_email@gmail.com', 'your_password')
# Send email
msg = MIMEText('Hello, Python!')
msg['Subject'] = 'Python Email'
server.sendmail('your_email@gmail.com', 'recipient@example.com', msg.as_string(
```

45 — *Working with JSON Files*

```
import json
data = {'name': 'John', 'age': 30}
# Write to JSON file
with open('data.json', 'w') as json_file:
    json.dump(data, json_file)
# Read from JSON file
with open('data.json', 'r') as json_file:
    loaded_data = json.load(json_file)
```

Thanks for reading✨

Python | Python Programming | Python3 | Machine Learning | Deep Learning