

Introduction to Python

Tan Sein Jone

University of British Columbia

July 16, 2024

Table of contents

1. Object Oriented Programming
2. Pythonics
3. Data Handling

Table of Contents

1. Object Oriented Programming

2. Pythonics

3. Data Handling

Classes and Objects

- A class is a blueprint for creating object
- Classes define the properties and behaviours of objects
- Objects have attributes and methods
- Attributes are variables that store data
- Methods are functions that perform actions

Attributes

- Attributes are variables that store data
- They are defined in the `__init__` method
- They are accessed using the dot operator

Attributes

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age
```

Attributes

```
person = Person("John", 36)
print(person.name)
print(person.age)
```

Methods

- Methods are functions that perform actions
- They are defined in the class
- They are accessed using the dot operator

Methods

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello , my_name_is {self.name}")
```

Methods

```
person = Person("John", 36)  
person.greet()
```

Benefits of OOP

- Encapsulation
- Inheritance
- Polymorphism

Encapsulation

- Encapsulation is the bundling of data and methods that operate on the data
- It restricts access to some of the object's components
- It prevents the accidental modification of data

Inheritance

- Inheritance is the mechanism of basing a class upon another class
- It allows a class to inherit attributes and methods from another class
- It allows a class to override methods of another class

Polymorphism

- Polymorphism is the ability to present the same interface for different data types
- It allows a function to accept different data types
- It allows a class to override methods of another class

1. Object Oriented Programming
2. Pythonics
3. Data Handling

Pythonics

- Pythonic code is code that follows the conventions of the Python language
- It is code that is clean, readable, and maintainable
- It is code that is idiomatic and expressive

Zen of Python

- The Zen of Python is a collection of aphorisms that capture the philosophy of Python
- It is a set of guiding principles for writing computer programs
- It is a set of rules for writing Pythonic code

Value Swapping and Multiple Assignment

- Python allows you to swap the values of two variables in a single line
- It also allows you to assign multiple values to multiple variables in a single line

Value Swapping and Multiple Assignment

```
a = 1  
b = 2  
a, b = b, a  
print(a, b)
```

Passing Multiple Arguments

- Python allows you to pass multiple arguments to a function
- It also allows you to pass keyword arguments to a function

Passing Multiple Arguments

```
def greet(*names):  
    for name in names:  
        print(f"Hello , {name}")  
  
greet("John", "Jane", "Jack")
```

List Comprehension

- List comprehension is a concise way to create lists
- It allows you to create lists using a single line of code
- It is more readable and expressive than traditional loops

List Comprehension

```
squares = [x ** 2 for x in range(10)]  
print(squares)
```

A Note on Indentation

- Python uses indentation to define blocks of code
- It uses whitespace to delimit code

1. Object Oriented Programming
2. Pythonics
3. Data Handling

Data Handling

- Data handling is the process of managing data
- It involves reading, writing, and processing data
- It involves working with files, databases, and APIs

Lists and Dictionaries

- Lists are ordered collections of items
- Dictionaries are unordered collections of key-value pairs
- Lists are indexed by integers
- Dictionaries are indexed by keys

Combining Lists and Dictionaries

- You can combine lists and dictionaries to create complex data structures
- You can nest lists and dictionaries to create hierarchical data structures

Combining Lists and Dictionaries

```
person = {  
    "name": "John",  
    "age": 36,  
    "friends": ["Jane", "Jack"]  
}  
print(person["name"])  
print(person["age"])  
print(person["friends"])
```

Reading and Writing Files

- Python allows you to read and write files
- It allows you to open files in read mode, write mode, or append mode
- It allows you to read files line by line or all at once

Reading and Writing Files

```
with open("data.txt", "w") as file:  
    file.write("Hello ,_world!")
```

```
with open("data.txt", "r") as file:  
    data = file.read()  
    print(data)
```

NumPy

- NumPy is a library for numerical computing
- It provides support for arrays and matrices
- It allows you to perform mathematical operations on arrays and matrices
- It is the foundation of many other libraries

NumPy

```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b
print(c)

d = np.dot(a, b)
print(d)
```

Pandas

- Pandas is a library for data manipulation and analysis
- It provides support for data structures like Series and DataFrame
- It allows you to read and write data from various sources
- It is built on top of NumPy

Pandas IO

- Pandas allows you to read and write data from various sources
- It allows you to read and write data from CSV files, Excel files, SQL databases, and APIs
- It allows you to read and write data from URLs, HTML tables, and clipboard

Pandas IO

```
import pandas as pd

data = pd.read_csv("data.csv")
print(data)

data.to_csv("data.csv", index=False)
```

Manipulating Dataframes

- Pandas allows you to manipulate dataframes
- It allows you to filter, sort, group, and aggregate data
- It allows you to merge, join, and concatenate data
- It allows you to reshape, pivot, and melt data

Manipulating Dataframes

```
import pandas as pd

data = pd.read_csv("age.csv")
data = data[data["age"] > 30]
data = data.sort_values("age")
data['rank'] = data['age'].rank()
data.loc['total'] = data.sum()
print(data)
```

pd.apply()

- Pandas allows you to apply functions to dataframes
- It allows you to apply functions to rows, columns, or cells
- It allows you to apply lambda functions, user-defined functions, or built-in functions

pd.apply()

```
import pandas as pd

data = pd.read_csv("age.csv")
data['age'] = data['age'].apply(lambda x: x + 1)
print(data)
```


Merge and Join

- Pandas allows you to merge and join dataframes
- It allows you to merge dataframes on columns or indices
- It allows you to merge dataframes using inner, outer, left, or right joins

Merge and Join

```
import pandas as pd

data1 = pd.read_csv("age.csv")
data2 = pd.read_csv("blood_type.csv")
data = pd.merge(data1, data2, on="name")
print(data)
```

Null Values

- Pandas allows you to handle null values
- It allows you to drop null values, fill null values, or interpolate null values
- It allows you to check for null values, count null values, or filter null values

Null Values

```
import pandas as pd

data = pd.read_csv("temp.csv")
data = data.dropna()
data = data.fillna(0)
data = data.interpolate()
print(data)
```

Rules of Thumb for Missing Data

- If the missing data is random, drop the rows
- If the missing data is systematic, fill the missing values
- If the missing data is time-dependent, interpolate the missing values