Python শিখুন সহজ উপায়ে

লেখক: শাকিল আহমেদ সাগর

সূচিপত্র:

1. ভূমিকা 2
2. পাইথনের পরিচিতি ………………………………………………………………….. 3
3. পাইথন ইনস্টলেশন এবং সেটআপ ………………………………………………… 4
4. পাইথনের মৌলিক ধারণা ………………………………………………………… 5
5. কন্ডিশনাল স্টেটমেন্ট (if-else) ………………………………………………... 7
6. লুপ (for, while) …………………………………………………………………... 8
7. ফাংশন এবং মডিউল ……………………………………………………..….. 10
8. ডাটা স্ট্রাকচার (লিস্ট, টাপল, সেট, ডিকশনারি) ………………………………..… 12
9. ফাইল হ্যান্ডলিং ……………………………………………………………….. 17
10. অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং (OOP) ……………………………………….. 19
11. পাইথনের লাইব্রেরি (NumPy, Pandas ইত্যাদি) ………………………………… 22
12. অ্যাডভান্সড টপিক (ডেকোরেটর, জেনারেটর, মাল্টিথ্রেডিং) ………………….. 23
13. প্রজেক্ট ভিত্তিক লার্নিং ………………………………………………………… 25
14. শেষ কথা ও পরবর্তী ধাপ ………………………………………………...… 28

অধ্যায় ১: ভূমিকা

পাইথন বর্তমানে জনপ্রিয়তম প্রোগ্রামিং ভাষাগুলোর মধ্যে একটি। সহজ সিনট্যাক্স, বিশাল লাইব্রেরি এবং বহুমুখী ব্যবহারের কারণে এটি শিক্ষার্থীদের জন্য আদর্শ। এই বইতে ধাপে ধাপে পাইথন শেখানো হবে, যাতে আপনি সহজে বোঝতে পারেন এবং নিজে কোড লিখতে পারেন।

কেন পাইথন শিখবেন**?**

- সহজ ও স্বচ্ছ সিনট্যাক্স: পাইথনের কোড অন্যান্য ভাষার তুলনায় পড়তে ও বুঝতে সহজ।
- বহুমুখী ব্যবহার: ওয়েব ডেভেলপমেন্ট, ডাটা সায়েন্স, মেশিন লার্নিং, গেম ডেভেলপমেন্ট, স্ক্রিপ্টিংসহ আরও অনেক কিছুতে ব্যবহৃত হয়।
- বিশাল কমিউনিটি: পাইথনের ব্যবহারকারী ও ডেভেলপারদের একটি বিশাল কমিউনিটি রয়েছে, যারা নতুনদের সাহায্য করতে প্রস্তুত।
- উন্নত লাইব্রেরি সংগ্রহ: পাইথনের বিভিন্ন লাইব্রেরি যেমন NumPy, Pandas, TensorFlow, Flask, Django ইত্যাদি বিভিন্ন কাজে ব্যবহার করা যায়।

এই বই থেকে কী শিখবেন**?**

- পাইথনের মৌলিক বিষয়গুলি যেমন ভেরিয়েবল, ডাটা টাইপ, কন্ডিশনাল স্টেটমেন্ট, লুপ ইত্যাদি।
- ফাংশন, মডিউল ও ফাইল হ্যান্ডলিং নিয়ে কাজ করা।
- অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং (OOP) এবং অ্যাডভান্সড টপিক সম্পর্কে ধারণা।
- প্রকল্প ভিত্তিক লার্নিংয়ের মাধ্যমে বাস্তব উদাহরণ ব্যবহার করে শেখা।

অধ্যায় ২: পাইথনের পরিচিতি

পাইথন হলো একটি উচ্চ-স্তরের, ইন্টারপ্রেটেড এবং অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ভাষা। এটি গুইডো ভ্যান রসুম ১৯৯১ সালে তৈরি করেন। পাইথন বর্তমানে ওয়েব ডেভেলপমেন্ট, ডাটা সায়েন্স, আর্টিফিশিয়াল ইন্টেলিজেন্স, সফটওয়্যার ডেভেলপমেন্ট, গেম ডেভেলপমেন্ট, এবং আরও অনেক ক্ষেত্রে ব্যবহৃত হয়।

পাইথনের বৈশিষ্ট্য:

- সহজ এবং স্পষ্ট সিনট্যাক্স: এটি সহজেই শেখা ও পড়া যায়।
- ওপেন সোর্স এবং ফ্রি: এটি ব্যবহার করতে কোনো লাইসেন্স ফি দিতে হয় না।
- ক্রস-প্ল্যাটফর্ম: পাইথন উইন্ডোজ, লিনাক্স, ম্যাক ওএস, এবং মোবাইল প্ল্যাটফর্মেও কাজ করে।
- বৈচিত্র্যময় লাইব্রেরি সমৃদ্ধ: পাইথনের বিশাল স্ট্যান্ডার্ড লাইব্রেরি রয়েছে, যা বিভিন্ন কাজে ব্যবহৃত হয়।
- ইন্টারপ্রেটেড ভাষা: পাইথন কোড রান করার জন্য কম্পাইল করার প্রয়োজন নেই, সরাসরি ইন্টারপ্রেটার ব্যবহার করা যায়।

পাইথন ব্যবহার ক্ষেত্র:

- ওয়েব ডেভেলপমেন্ট: Django, Flask, FastAPI ইত্যাদি ফ্রেমওয়ার্ক দিয়ে ওয়েব অ্যাপ তৈরি করা যায়।
- ডাটা সায়েন্স: NumPy, Pandas, Matplotlib, Seaborn ইত্যাদি লাইব্রেরি ব্যবহৃত হয়।
- মেশিন লার্নিং এবং এআই: TensorFlow, Keras, PyTorch ইত্যাদি লাইব্রেরি রয়েছে।
- গেম ডেভেলপমেন্ট: Pygame ব্যবহার করে গেম তৈরি করা যায়।
- অটোমেশন ও স্ক্রিপ্টিং: পাইথন ব্যবহার করে বিভিন্ন কাজ স্বয়ংক্রিয়ভাবে সম্পন্ন করা যায়।

প্রথম পাইথন প্রোগ্রাম

আমরা পাইথনে প্রথম প্রোগ্রাম রান করব, যা "Hello, Python!" প্রিন্ট করবে।

```
print("Hello, Python!")
```

এই কোড রান করালে নিম্নলিখিত আউটপুট আসবে:

```
Hello, Python!
```

অধ্যায় ৩: পাইথন ইনস্টলেশন এবং সেটআপ **(Python, Pycharm)**

এই অধ্যায়ে পাইথন ইনস্টল করা এবং সেটআপ করার পদ্ধতি সম্পর্কে আলোচনা করা হবে।

৩.১ পাইথন ইনস্টলেশন

১. Python অফিসিয়াল ওয়েবসাইট এ যান। ২. আপনার অপারেটিং সিস্টেম অনুযায়ী সঠিক ভার্সন ডাউনলোড করুন। ৩. ডাউনলোড করা ফাইলটি চালিয়ে ইন্সটল করুন। ৪. Add Python to PATH অপশনটি সিলেক্ট করুন যাতে কমান্ড লাইনে সহজেই ব্যবহার করা যায়।

৩.২ **Pycharm** ইনস্টলেশন ও সেটআপ

Pycharm হলো একটি শক্তিশালী IDE, যা পাইথন প্রোগ্রামিংয়ের জন্য বিশেষভাবে ব্যবহৃত হয়।

১. JetBrains অফিসিয়াল ওয়েবসাইট এ যান। ২. Community Edition (ফ্রি ভার্সন) অথবা Professional Edition ডাউনলোড করুন। ৩. ডাউনলোড করা ফাইলটি চালিয়ে ইন্সটল করুন। ৪. Pycharm চালু করে নতুন প্রোজেক্ট তৈরি করুন। ৫. Interpreter হিসেবে আপনার ইনস্টল করা পাইথন ভার্সন সেট করুন।

এখন আপনি Pycharm ব্যবহার করে সহজেই পাইথন প্রোগ্রামিং শুরু করতে পারবেন।

৩.৩ প্রথম প্রোগ্রাম রান করা

Pycharm-এ প্রথম প্রোগ্রাম রান করতে নিচের ধাপ অনুসরণ করুন:

১. Pycharm ওপেন করুন এবং একটি নতুন প্রোজেক্ট তৈরি করুন। ২. নতুন Python ফাইল তৈরি করুন (hello.py নামে)। ৩. ফাইলটিতে নিচের কোড লিখুন:

```
print("Hello, Python World!")
```

৪. Run বাটনে ক্লিক করুন অথবা Shift + F10 চাপুন।

আপনার স্ক্রিনে Hello, Python World! প্রিন্ট হবে। 🎉

অধ্যায় ৪: পাইথনের মৌলিক ধারণা

এই অধ্যায়ে পাইথনের মৌলিক বিষয়গুলো আলোচনা করা হবে, যা একজন নতুন প্রোগ্রামারের জন্য অপরিহার্য।

৪.১ ভেরিয়েবল এবং ডাটা টাইপ

ভেরিয়েবল হল একটি কন্টেইনার যা ডাটা সংরক্ষণ করে। পাইথনে বিভিন্ন ধরনের ডাটা টাইপ রয়েছে, যেমন:

```
name = "Rahim"  # স্ট্রিং

age = 25  # পূর্ণসংখ্যা (integer)

height = 5.7  # দশমিক সংখ্যা (float)

is_student = True  # বুলিয়ান (boolean)
```

৪.২ ইনপুট এবং আউটপুট

ব্যবহারকারীর ইনপুট নেওয়া এবং স্ক্রিনে আউটপুট দেখানোর জন্য input() এবং print() ফাংশন ব্যবহৃত হয়।

```
name = input("আপনার নাম লিখুন: ")

print("স্বাগতম,", name)
```

৪.৩ অপারেটর

পাইথনে গণিত, তুলনা, লজিক্যাল, অ্যাসাইনমেন্ট এবং অন্যান্য অপারেটর রয়েছে।

```
x = 10

y = 5
```

গণিত অপারেটর

```
sum = x + y

print("যোগফল:", sum)
```

তুলনা অপারেটর

```
print(x > y)  # True
```

লজিক্যাল অপারেটর

```
print(x > 0 and y > 0)  # True
```

৪.৪ কমেন্ট

কমেন্ট ব্যবহার করে কোডের ব্যাখ্যা লেখা যায়, যা পরে বুঝতে সহায়ক হয়।

```
# এটি একটি একলাইন কমেন্ট
print("Hello, World!")  # এটি ইনলাইন কমেন্ট
```

```
"""
এটি মাল্টিলাইন কমেন্ট
যা একাধিক লাইনে লেখা যায়
"""
```

৪.৫ টাইপ কনভার্শন

পাইথনে এক ধরনের ডাটা টাইপ থেকে অন্যটিতে রূপান্তর করা যায়।

```
num_str = "25"
num_int = int(num_str)  # স্ট্রিং থেকে পূর্ণসংখ্যায় রূপান্তর
print(num_int + 5)  # 30
```

অধ্যায় ৫: কন্ডিশনাল স্টেটমেন্ট **(if-else)**

কন্ডিশনাল স্টেটমেন্ট প্রোগ্রামিংয়ে গুরুত্বপূর্ণ ভূমিকা রাখে। এটি প্রোগ্রামের লজিক নির্ধারণ করে এবং নির্দিষ্ট শর্ত অনুযায়ী সিদ্ধান্ত গ্রহণ করে।

if স্টেটমেন্ট

```
x = 10

if x > 5:

   print("x বড় 5 এর চেয়ে")
```

ব্যাখ্যা: এখানে if স্টেটমেন্ট চেক করছে যে x এর মান ৫-এর চেয়ে বড় কি না। যদি বড় হয়, তাহলে print ফাংশনের মাধ্যমে মেসেজ প্রদর্শন করবে।

if-else স্টেটমেন্ট

```
x = 3

if x > 5:

   print("x বড় 5 এর চেয়ে")

else:

   print("x ছোট বা সমান 5")
```

ব্যাখ্যা: যদি x এর মান ৫-এর বেশি হয়, তাহলে প্রথম print চলবে, নাহলে else ব্লকের মধ্যে থাকা print চলবে।

if-elif-else স্টেটমেন্ট

```
x = 7

if x > 10:

   print("x বড় 10 এর চেয়ে")

elif x > 5:

   print("x বড় 5 এর চেয়ে কিন্তু 10 এর কম")

else:

   print("x 5 বা তার কম")
```

ব্যাখ্যা:

- যদি x ১০-এর বেশি হয়, তাহলে প্রথম if ব্লক চলবে।
- যদি ৫-এর বেশি কিন্তু ১০-এর কম হয়, তাহলে elif ব্লক চলবে।
- অন্যথায়, else ব্লক চলবে।

অধ্যায় ৬: লুপ (for, while)

লুপ ব্যবহার করে পুনরাবৃত্তিমূলক কাজ সহজ করা যায়। পাইথনে প্রধানত দুটি লুপ ব্যবহৃত হয়: for এবং while।

for লুপ

```
for i in range(5):
  print("Hello, Python!")
```

ব্যাখ্যা: for লুপ range(5) অর্থাৎ ০ থেকে ৪ পর্যন্ত চলবে এবং প্রতিবার "Hello, Python!" প্রিন্ট করবে।

while লুপ

```
x = 0
while x < 5:
  print("Counting:", x)
  x += 1
```

ব্যাখ্যা: এখানে while লুপ চলতে থাকবে যতক্ষণ না x ৫-এর সমান বা বেশি হয়। প্রতিবার x এর মান এক করে বৃদ্ধি পাবে।

লুপ কন্ট্রোল স্টেটমেন্ট

- break লুপ থামানোর জন্য
- continue বর্তমান পুনরাবৃত্তি বাদ দিয়ে পরবর্তী পুনরাবৃত্তিতে যাওয়ার জন্য

```
for i in range(10):
  if i == 5:
    break  # লুপ থেমে যাবে
  print(i)
```

ব্যাখ্যা: যখন i এর মান ৫ হবে, তখন break স্টেটমেন্ট লুপ বন্ধ করে দেবে এবং এরপরের কোনো মান প্রিন্ট হবে না।

```
for i in range(10):
  if i == 5:
    continue  # ৫ বাদ দিয়ে পরবর্তী পুনরাবৃত্তিতে যাবে
  print(i)
```

ব্যাখ্যা: যখন i == 5, তখন continue স্টেটমেন্ট কার্যকর হবে এবং print(i) স্কিপ করে পরবর্তী লুপ চলবে।

অধ্যায় ৭: ফাংশন এবং মডিউল

প্রোগ্রামিংয়ে ফাংশন হলো কোড পুনরায় ব্যবহারযোগ্য করার একটি মাধ্যম। পাইথনে বিল্ট-ইন এবং ইউজার-ডিফাইন্ড ফাংশন থাকে।

ফাংশন তৈরি

```
def greet(name):
    return f"Hello, {name}!"


message = greet("Alice")
print(message)
```

ব্যাখ্যা:

- def কীওয়ার্ড দিয়ে ফাংশন ডিফাইন করা হয়।
- greet ফাংশন একটি প্যারামিটার name গ্রহণ করে এবং একটি স্ট্রিং রিটার্ন করে।

ডিফল্ট প্যারামিটার

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


print(greet())
print(greet("Bob"))
```

ব্যাখ্যা: যদি name প্যারামিটার দেওয়া না হয়, তাহলে Guest ব্যবহার করা হবে।

আরগুমেন্টের ভেরিয়েবল সংখ্যা (*args, **kwargs)

```
def add_numbers(*numbers):
    return sum(numbers)


print(add_numbers(1, 2, 3, 4))
```

ব্যাখ্যা: *args ব্যবহার করে একাধিক সংখ্যা গ্রহণ করা যায় এবং sum() ফাংশন ব্যবহার করে যোগফল বের করা হয়।

মডিউল ব্যবহার

মডিউল হলো কোড পুনরায় ব্যবহারের একটি পদ্ধতি। পাইথনের বিল্ট-ইন ও কাস্টম মডিউল ব্যবহার করা যায়।

```
import math

print(math.sqrt(16))  # স্কয়ার রুট গণনা
```

ব্যাখ্যা: math মডিউল ইমপোর্ট করে sqrt() ফাংশন ব্যবহার করা হয়েছে।

অধ্যায় ৮: ডাটা স্ট্রাকচার (লিস্ট, টাপল, সেট, ডিকশনারি)

পাইথনে বিভিন্ন ধরনের ডাটা স্ট্রাকচার রয়েছে, যা ডাটা সংরক্ষণ এবং ব্যবস্থাপনার জন্য ব্যবহৃত হয়। এই অধ্যায়ে আমরা লিস্ট, টাপল, সেট, এবং ডিকশনারি সম্পর্কে বিস্তারিত জানব।

৮.১ লিস্ট **(List)**

লিস্ট হলো সাজানো (Ordered) এবং পরিবর্তনযোগ্য (Mutable) ডাটা স্ট্রাকচার, যেখানে একাধিক মান রাখা যায়। লিস্টে একই বা ভিন্ন ধরনের ডাটা রাখা যায় এবং এটি পরিবর্তন করা সম্ভব।

লিস্ট তৈরি ও ব্যবহার:

```
# একটি লিস্ট তৈরি
fruits = ["আপেল", "কলা", "আম"]
print(fruits)  # ['আপেল', 'কলা', 'আম']

# নির্দিষ্ট ইনডেক্সের মান বের করা
print(fruits[1])  # কলা

# লিস্টের মান পরিবর্তন
fruits[1] = "কমলা"
print(fruits)  # ['আপেল', 'কমলা', 'আম']

# নতুন মান যোগ করা
fruits.append("লিচু")
print(fruits)  # ['আপেল', 'কমলা', 'আম', 'লিচু']

# নির্দিষ্ট স্থানে মান ঢোকানো
fruits.insert(1, "আঙ্গুর")
print(fruits)  # ['আপেল', 'আঙ্গুর', 'কমলা', 'আম', 'লিচু']

# মান মুছে ফেলা
fruits.remove("আম")
```

```
print(fruits)  # ['আপেল', 'আঙ্গুর', 'কমলা', 'লিচু']

# লিস্ট উল্টানো
fruits.reverse()
print(fruits)  # ['লিচু', 'কমলা', 'আঙ্গুর', 'আপেল']

# লিস্ট সাজানো
fruits.sort()
print(fruits)  # ['আপেল', 'আঙ্গুর', 'কমলা', 'লিচু']
```

৮.২ টাপল **(Tuple)**

টাপল হলো সাজানো (Ordered) কিন্তু অপরিবর্তনযোগ্য (Immutable) ডাটা স্ট্রাকচার। একবার টাপল তৈরি হলে, এর মান পরিবর্তন করা যায় না।

টাপল তৈরি ও ব্যবহার:

```
# একটি টাপল তৈরি
colors = ("লাল", "নীল", "সবুজ")
print(colors)  # ('লাল', 'নীল', 'সবুজ')

# নির্দিষ্ট ইনডেক্সের মান বের করা
print(colors[1])  # নীল

# টাপল পরিবর্তন করা যায় না
# colors[1] = "হলুদ"  # এটি ত্রুটি সৃষ্টি করবে

# টাপলের দৈর্ঘ্য বের করা
```

```
print(len(colors))  # ৩

# একটি টাপলে একমাত্র মান থাকলে অবশ্যই কমা দিতে হবে
single_element_tuple = ("একটি মান",)
print(type(single_element_tuple))  # <class 'tuple'>
```

৮.৩ সেট **(Set)**

সেট হলো অসাজানো (Unordered) এবং ইউনিক (Unique) মান ধারণকারী ডাটা স্ট্রাকচার। এতে ডুপ্লিকেট মান সংরক্ষণ করা হয় না।

সেট তৈরি ও ব্যবহার:

```
# একটি সেট তৈরি
numbers = {1, 2, 3, 4, 4, 5}
print(numbers)  # {1, 2, 3, 4, 5} (ডুপ্লিকেট স্বয়ংক্রিয়ভাবে সরিয়ে দেয়)

# নতুন মান যোগ করা
numbers.add(6)
print(numbers)  # {1, 2, 3, 4, 5, 6}

# মান মুছে ফেলা
numbers.remove(3)
print(numbers)  # {1, 2, 4, 5, 6}

# সেটের দুটি উপাদানের সংযোগ (Union)
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
print(union_set)  # {1, 2, 3, 4, 5}
```

```
# দুটি সেটের সাধারণ উপাদান (Intersection)

intersection_set = set1.intersection(set2)

print(intersection_set)  # {3}
```

৮.৪ ডিকশনারি **(Dictionary)**

ডিকশনারি হলো কি-ভ্যালু (Key-Value) পেয়ার ভিত্তিক ডাটা স্ট্রাকচার। এটি সাজানো থাকে এবং পরিবর্তনযোগ্য।

ডিকশনারি তৈরি ও ব্যবহার:

```
# একটি ডিকশনারি তৈরি

student = {

   "নাম": "রাকিব",

   "বয়স": ২০,

   "শ্রেণী": "একাদশ"

}

print(student)  # {'নাম': 'রাকিব', 'বয়স': ২০, 'শ্রেণী': 'একাদশ'}


# নির্দিষ্ট মান বের করা

print(student["নাম"])  # রাকিব


# নতুন মান যোগ করা

student["রোল"] = ৫

print(student)  # {'নাম': 'রাকিব', 'বয়স': ২০, 'শ্রেণী': 'একাদশ', 'রোল': ৫}


# ডিকশনারির মান পরিবর্তন করা

student["বয়স"] = ২১

print(student)  # {'নাম': 'রাকিব', 'বয়স': ২১, 'শ্রেণী': 'একাদশ', 'রোল': ৫}


# একটি কী-এর মান সরানো
```

```
del student["শ্রেণী"]

print(student)  # {'নাম': 'রাকিব', 'বয়স': ২১, 'রোল': ৫}


# কেবল কী বের করা

print(student.keys())  # dict_keys(['নাম', 'বয়স', 'রোল'])


# কেবল মান বের করা

print(student.values())  # dict_values(['রাকিব', ২১, ৫])
```

৮.৫ ডাটা স্ট্রাকচারের তুলনা

৮.৬ নেস্টেড ডাটা স্ট্রাকচার **(Nested Data Structures)**

আমরা পাইথনে নেস্টেড ডাটা স্ট্রাকচার ব্যবহার করতে পারি, যেখানে একটি লিস্ট বা ডিকশনারির মধ্যে অন্য লিস্ট বা ডিকশনারি থাকতে পারে।

```
students = {

   "রাকিব": {"বয়স": ২০, "রোল": ৫},

   "করিম": {"বয়স": ২১, "রোল": ৬}

}


print(students["রাকিব"]["বয়স"])  # ২০
```

অধ্যায় ৯: ফাইল হ্যান্ডলিং

ফাইল হ্যান্ডলিং হলো প্রোগ্রামে ডাটা সংরক্ষণ ও রিট্রিভ করার একটি গুরুত্বপূর্ণ পদ্ধতি। পাইথনে ফাইল পড়া, লেখা, এবং পরিবর্তন করা সহজেই করা যায়।

ফাইল খোলা এবং লেখা

```
with open("example.txt", "w") as file:
    file.write("এইটি একটি নমুনা ফাইল।")
```

ব্যাখ্যা:

- open("filename", "mode") ফাংশন ফাইল খুলতে ব্যবহৃত হয়।
- "w" মোড ফাইল লেখার জন্য ব্যবহৃত হয়।
- with কিওয়ার্ড স্বয়ংক্রিয়ভাবে ফাইল বন্ধ করে।

ফাইল পড়া

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

ব্যাখ্যা:

- "r" মোড ফাইল পড়ার জন্য ব্যবহৃত হয়।
- .read() মেথড পুরো ফাইলের কনটেন্ট পড়ে।

ফাইল লাইনে লাইনে পড়া

```
with open("example.txt", "r") as file:
    for line in file:
        print(line.strip())
```

ব্যাখ্যা:

- .strip() ব্যবহার করে নতুন লাইনের (\n) মতো অপ্রয়োজনীয় ক্যারেক্টার সরানো হয়।
- for line in file লুপ প্রতিটি লাইন পড়ার জন্য ব্যবহৃত হয়।

ফাইল সংযোজন (Append)

```
with open("example.txt", "a") as file:
    file.write("\nনতুন লাইন যোগ করা হলো।")
```

ব্যাখ্যা:

- "a" মোড ফাইলের শেষে নতুন ডাটা যোগ করতে ব্যবহৃত হয়।

ফাইল মুছে ফেলা

```
import os

os.remove("example.txt")
```

ব্যাখ্যা:

- os.remove("filename") ফাইল মুছে ফেলার জন্য ব্যবহৃত হয়।

অধ্যায় ১০: অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং **(OOP)**

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং (OOP) হলো একটি প্রোগ্রামিং প্যারাডাইম যেখানে ডাটা এবং ফাংশনকে একত্রিত করে অবজেক্ট তৈরি করা হয়। পাইথনে OOP-এর মূল বৈশিষ্ট্য হলো ক্লাস, অবজেক্ট, ইনহেরিটেন্স, পলিমরফিজম, এনক্যাপসুলেশন।

ক্লাস এবং অবজেক্ট

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display_info(self):
        print(f"গাড়ির ব্র্যান্ড: {self.brand}, মডেল: {self.model}")
```

অবজেক্ট তৈরি

```
car1 = Car("Toyota", "Corolla")
car1.display_info()
```

ব্যাখ্যা:

- class কিওয়ার্ড ব্যবহার করে একটি নতুন ক্লাস তৈরি করা হয়।
- __init__ মেথড অবজেক্ট তৈরির সময় চলতে থাকে এবং ইনিশিয়াল ভ্যালু সেট করে।
- self হলো ক্লাসের ইনস্ট্যান্স যা বৈশিষ্ট্য ও মেথড অ্যাক্সেস করতে ব্যবহৃত হয়।

ইনহেরিটেন্স (Inheritance)

ইনহেরিটেন্স হলো একটি ক্লাসের বৈশিষ্ট্য অন্য ক্লাসে উত্তরাধিকার সূত্রে পাওয়া।

```
class ElectricCar(Car):
    def __init__(self, brand, model, battery_capacity):
        super().__init__(brand, model)
        self.battery_capacity = battery_capacity

    def display_battery(self):
```

```
    print(f"ব্যাটারি ক্যাপাসিটি: {self.battery_capacity} kWh")
```

অবজেক্ট তৈরি

```
car2 = ElectricCar("Tesla", "Model S", 100)
car2.display_info()
car2.display_battery()
```

ব্যাখ্যা:

- super() ফাংশন প্যারেন্ট ক্লাসের __init__ মেথড কল করতে ব্যবহৃত হয়।
- ElectricCar ক্লাস Car ক্লাস থেকে উত্তরাধিকার সূত্রে প্রাপ্ত।

এনক্যাপসুলেশন (Encapsulation)

এনক্যাপসুলেশন হলো ডাটা হাইড করা, যাতে এটি সরাসরি পরিবর্তন করা না যায়।

```
class BankAccount:
  def __init__(self, balance):
    self.__balance = balance  # প্রাইভেট ভেরিয়েবল


  def deposit(self, amount):
    self.__balance += amount
    print(f"আপনার নতুন ব্যালেন্স: {self.__balance}")


  def get_balance(self):
    return self.__balance
```

অবজেক্ট তৈরি

```
account = BankAccount(1000)
account.deposit(500)
print(account.get_balance())
```

ব্যাখ্যা:

- __balance হলো প্রাইভেট ভেরিয়েবল, যা সরাসরি পরিবর্তন করা যাবে না।

- get_balance() মেথড দিয়ে ব্যালেন্স রিট্রিভ করা হয়।

পলিমরফিজম (Polymorphism)

পলিমরফিজম একই মেথডের বিভিন্ন কার্যকারিতা বোঝায়।

```
class Animal:
    def make_sound(self):
        pass


class Dog(Animal):
    def make_sound(self):
        print("ভৌ! ভৌ!")


class Cat(Animal):
    def make_sound(self):
        print("মিয়াও! মিয়াও!")
```

অবজেক্ট তৈরি

```
animals = [Dog(), Cat()]
for animal in animals:
    animal.make_sound()
```

ব্যাখ্যা:

- make_sound() মেথড প্রতিটি ক্লাসের জন্য আলাদাভাবে কাজ করে।
- এটি পলিমরফিজমের একটি উদাহরণ।

অধ্যায় ১১: পাইথনের লাইব্রেরি (NumPy, Pandas ইত্যাদি)

পাইথনের শক্তিশালী লাইব্রেরিগুলো ডাটা প্রসেসিং, বিশ্লেষণ এবং গণনার জন্য ব্যবহৃত হয়। এই অধ্যায়ে আমরা NumPy এবং Pandas নিয়ে আলোচনা করব।

NumPy

NumPy (Numerical Python) হলো একটি শক্তিশালী লাইব্রেরি যা অ্যারে এবং ম্যাট্রিক্স ভিত্তিক গণনার জন্য ব্যবহৃত হয়।

```
import numpy as np
```

এক মাত্রিক অ্যারে তৈরি

```
arr = np.array([1, 2, 3, 4, 5])
print("NumPy Array:", arr)
```

অ্যারের মৌলিক অপারেশন

```
print("গুণফল:", arr * 2)
```

Pandas

Pandas হলো ডাটা বিশ্লেষণের জন্য ব্যবহৃত একটি জনপ্রিয় লাইব্রেরি। এটি ডাটা ফ্রেম এবং সিরিজ অবজেক্ট সরবরাহ করে।

```
import pandas as pd
```

ডাটা ফ্রেম তৈরি

```
data = {
    "নাম": ["আলম", "জাবেদ", "সুমন"],
    "বয়স": [25, 30, 28]
}
df = pd.DataFrame(data)
print(df)
```

এই অধ্যায়ে আমরা আরও বিস্তারিতভাবে NumPy এবং Pandas ব্যবহার করে ডাটা ম্যানিপুলেশন নিয়ে আলোচনা করব।

অধ্যায় ১২: অ্যাডভান্সড টপিক (ডেকোরেটর, জেনারেটর, মাল্টিথ্রেডিং)

এই অধ্যায়ে আমরা পাইথনের কিছু উন্নত বিষয় নিয়ে আলোচনা করব, যেমন ডেকোরেটর, জেনারেটর, এবং মাল্টিথ্রেডিং।

১. ডেকোরেটর (Decorator)

ডেকোরেটর হলো এমন একটি ফাংশন, যা অন্য একটি ফাংশনের উপরে অতিরিক্ত কার্যকারিতা যোগ করতে ব্যবহৃত হয়।

```
def decorator_function(original_function):
    def wrapper_function():
        print(f"{original_function.__name__} ফাংশনের আগে কিছু হচ্ছে।")
        original_function()
        print(f"{original_function.__name__} ফাংশনের পরে কিছু হচ্ছে।")
    return wrapper_function


@decorator_function
def say_hello():
    print("হ্যালো, পাইথন!")


say_hello()
```

২. জেনারেটর (Generator)

জেনারেটর একটি বিশেষ ধরণের ফাংশন যা yield কীওয়ার্ড ব্যবহার করে একাধিক মান প্রদান করতে পারে।

```
def my_generator():
    yield 1
    yield 2
    yield 3


for value in my_generator():
    print(value)
```

৩. মাল্টিথ্রেডিং (Multithreading)

মাল্টিথ্রেডিং ব্যবহার করে একই সাথে একাধিক কাজ করা যায়।

```
import threading


def print_numbers():
    for i in range(1, 6):
        print(i)


def print_letters():
    for letter in "ABCDE":
        print(letter)
```

থ্রেড তৈরি

```
thread1 = threading.Thread(target=print_numbers)
thread2 = threading.Thread(target=print_letters)
```

থ্রেড শুরু করা

```
thread1.start()
thread2.start()
```

থ্রেড শেষ হওয়া পর্যন্ত অপেক্ষা করা

```
thread1.join()
thread2.join()
```

অধ্যায় ১৩: প্রজেক্ট ভিত্তিক লার্নিং

এই অধ্যায়ে আমরা পাইথনের মাধ্যমে কিছু প্রজেক্ট তৈরি করব, যা বাস্তব জীবনে প্রোগ্রামিং দক্ষতা বাড়াতে সাহায্য করবে। এখানে আমরা তিনটি ছোট-বড় প্রজেক্ট দেখব।

১. টেক্সট বেসড ক্যালকুলেটর

একটি সাধারণ ক্যালকুলেটর প্রোগ্রাম তৈরি করা যা ব্যবহারকারীর ইনপুট গ্রহণ করবে এবং ফলাফল প্রদর্শন করবে।

```
def calculator():
    while True:
        try:
            num1 = float(input("প্রথম সংখ্যা লিখুন: "))
            operator = input("অপারেটর (+, -, *, /) লিখুন: ")
            num2 = float(input("দ্বিতীয় সংখ্যা লিখুন: "))

            if operator == '+':
                result = num1 + num2
            elif operator == '-':
                result = num1 - num2
            elif operator == '*':
                result = num1 * num2
            elif operator == '/':
                result = num1 / num2
            else:
                print("অবৈধ অপারেটর!")
                continue

            print(f"ফলাফল: {result}")
        except ValueError:
            print("সঠিক সংখ্যা লিখুন!")
        except ZeroDivisionError:
```

```
        print("শূন্য দিয়ে ভাগ করা যায় না!")

calculator()
```

২. টু-ডু লিস্ট অ্যাপ

একটি সহজ প্রোগ্রাম যা ব্যবহারকারীর টু-ডু লিস্ট সংরক্ষণ ও দেখাতে পারবে।

```
tasks = []

def add_task(task):
    tasks.append(task)
    print("কাজ সংযুক্ত করা হয়েছে।")

def show_tasks():
    print("আপনার টু-ডু লিস্ট:")
    for idx, task in enumerate(tasks, 1):
        print(f"{idx}. {task}")

while True:
    choice = input("1. নতুন কাজ যোগ করুন, 2. কাজ দেখুন, 3. বের হন: ")
    if choice == '1':
        task = input("নতুন কাজ লিখুন: ")
        add_task(task)
    elif choice == '2':
        show_tasks()
    elif choice == '3':
        break
    else:
        print("অবৈধ ইনপুট!")
```

৩. ওয়েব স্ক্র্যাপিং

BeautifulSoup ব্যবহার করে একটি সাধারণ ওয়েব স্ক্র্যাপিং টুল।

```
import requests
from bs4 import BeautifulSoup

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

print("ওয়েবসাইট শিরোনাম:", soup.title.string)
```

অধ্যায় ১৫: পাইথনের মাধ্যমে অটোমেশন

এই অধ্যায়ে পাইথন ব্যবহার করে কীভাবে বিভিন্ন রকম অটোমেশন করা যায় তা নিয়ে আলোচনা করা হবে।

১. ফাইল এবং ফোল্ডার ব্যবস্থাপনা

```
import os


def create_folder(folder_name):
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)
        print(f"ফোল্ডার '{folder_name}' তৈরি করা হয়েছে।")
    else:
        print(f"ফোল্ডার '{folder_name}' ইতিমধ্যে বিদ্যমান।")


create_folder("MyNewFolder")
```

২. ইমেইল পাঠানো

```
import smtplib
from email.message import EmailMessage


def send_email():
    msg = EmailMessage()
    msg.set_content("এই ইমেইলটি পাইথনের মাধ্যমে পাঠানো হয়েছে।")
    msg['Subject'] = "অটোমেশন ইমেইল"
    msg['From'] = "your_email@gmail.com"
    msg['To'] = "receiver_email@gmail.com"

    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
        server.login("your_email@gmail.com", "your_password")
        server.send_message(msg)
```

```
    print("ইমেইল পাঠানো হয়েছে!")

# send_email() (এই লাইনটি চালানোর আগে সঠিক ইমেইল এবং পাসওয়ার্ড সেট করুন)
```

৩. ওয়েব স্ক্র্যাপিং এবং ডাটা সংগ্রহ

```
import requests
from bs4 import BeautifulSoup

def scrape_website(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    print("ওয়েবসাইট শিরোনাম:", soup.title.string)

scrape_website("https://example.com")
```