Χειρισμός Αρχείων στην Python (File Handling in Python)

Τρόποι Ανοίγματος Αρχείων (File Opening Modes)

Βασικοί Τρόποι (Basic Modes):

- `'r'`: Άνοιγμα για ανάγνωση (read) - προεπιλογή
- `'w'`: Άνοιγμα για εγγραφή (write) - διαγράφει το περιεχόμενο
- `'a'`: Άνοιγμα για προσθήκη (append) - προσθέτει στο τέλος
- `'x'`: Άνοιγμα για αποκλειστική δημιουργία (exclusive creation)

Επιπρόσθετες Επιλογές (Additional Options):

- `'b'`: Δυαδική λειτουργία (binary mode)
- `'t'`: Λειτουργία κειμένου (text mode) - προεπιλογή
- `'+'`: Ανάγνωση και εγγραφή (read and write)

Η Εντολή with (The with Statement)

Η χρήση του `with` εξασφαλίζει το σωστό κλείσιμο του αρχείου ακόμα και σε περίπτωση σφάλματος. (Using `with` ensures proper file closing even in case of errors.)

```
# Παραδειγματα χρήσης with (Examples of using with)
# Ανάγνωση (Reading)
with open('file.txt', 'r', encoding='utf-8') as file:
    content = file.read()

# Εγγραφή (Writing)
with open('file.txt', 'w', encoding='utf-8') as file:
    file.write('Γεια σας!')

# Πολλαπλά αρχεία (Multiple files)
with open('input.txt', 'r') as input_file, open('output.txt', 'w') as
output_file:
    content = input_file.read()
    output_file.write(content.upper())
```

Μέθοδοι Ανάγνωσης (Reading Methods)

```
with open('example.txt', 'r', encoding='utf-8') as file:
    # Διάβασμα ολόκληρου αρχείου (Read entire file)
    whole_file = file.read()

    # Διάβασμα συγκεκριμένου αριθμού χαρακτήρων (Read specific number of
characters)
    first_10_chars = file.read(10)

    # Διάβασμα μίας γραμμής (Read one line)
    one_line = file.readline()

    # Διάβασμα όλων των γραμμών σε λίστα (Read all lines into a list)
```

```
    all_lines = file.readlines()

    # Επανάληψη ανά γραμμή (Iterate through lines)
    for line in file:
        print(line.strip())
```

Μέθοδοι Εγγραφής (Writing Methods)

```
with open('output.txt', 'w', encoding='utf-8') as file:
    # Εγγραφή κειμένου (Write text)
    file.write('Γεια σας!\n')

    # Εγγραφή πολλαπλών γραμμών (Write multiple lines)
    lines = ['Γραμμή 1\n', 'Γραμμή 2\n', 'Γραμμή 3\n']
    file.writelines(lines)

    # Εγγραφή με print (Write using print)
    print('Χρήση print()', file=file)
```

Χρήση του fseek και Σχετικές Μέθοδοι (Using fseek and Related Methods)

```
with open('file.txt', 'r+', encoding='utf-8') as file:
    # Μετακίνηση δείκτη (Move pointer)
    file.seek(5)  # Μετακίνηση 5 χαρακτήρες μπροστά

    # Τρέχουσα θέση δείκτη (Current pointer position)
    position = file.tell()

    # Διαφορετικοί τρόποι seek (Different seek modes)
    file.seek(0)      # Αρχή αρχείου (Start of file)
    file.seek(0, 0)   # Ίδιο με παραπάνω (Same as above)
    file.seek(0, 1)   # Σχετική με τρέχουσα θέση (Relative to current
position)
    file.seek(0, 2)   # Τέλος αρχείου (End of file)
```

Παραδείγματα Χειρισμού Δυαδικών Αρχείων (Binary File Handling Examples)

```
# Εγγραφή δυαδικών δεδομένων (Writing binary data)
with open('binary.dat', 'wb') as file:
    data = bytes([65, 66, 67, 68])  # ASCII για 'ABCD'
    file.write(data)

# Ανάγνωση δυαδικών δεδομένων (Reading binary data)
with open('binary.dat', 'rb') as file:
    binary_data = file.read()
```

Χειρισμός Σφαλμάτων (Error Handling)

```
try:
    with open('nonexistent.txt', 'r') as file:
        content = file.read()
except FileNotFoundError:
    print("Το αρχείο δεν βρέθηκε!")
    print("File not found!")
```

```
except PermissionError:
    print("Δεν έχετε δικαιώματα πρόσβασης!")
    print("You don't have permission!")
except IOError as e:
    print(f"Σφάλμα I/O: {e}")
    print(f"I/O error: {e}")
```

Πρακτικά Παραδείγματα (Practical Examples)

1. Αντιγραφή Αρχείου (File Copy)

```
def copy_file(source, destination):
    with open(source, 'rb') as src, open(destination, 'wb') as dst:
        dst.write(src.read())
```

2. Καταμέτρηση Γραμμών (Line Counter)

```
def count_lines(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        return sum(1 for line in file)
```

3. Αναζήτηση σε Αρχείο (File Search)

```
def search_in_file(filename, search_term):
    with open(filename, 'r', encoding='utf-8') as file:
        for line_number, line in enumerate(file, 1):
            if search_term in line:
                print(f"Βρέθηκε στη γραμμή {line_number}: {line.strip()}")
                print(f"Found in line {line_number}: {line.strip()}")
```

Συμβουλές και Καλές Πρακτικές (Tips and Best Practices)

1. **Πάντα χρησιμοποιείτε την εντολή `with`**
   - Εξασφαλίζει το σωστό κλείσιμο του αρχείου
   - (Ensures proper file closing)
2. **Καθορίστε την κωδικοποίηση**
   - Χρησιμοποιήστε `encoding='utf-8'` για υποστήριξη Unicode
   - (Use `encoding='utf-8'` for Unicode support)
3. **Χειριστείτε τα σφάλματα**
   - Περιβάλλετε τις λειτουργίες αρχείων με try-except
   - (Wrap file operations in try-except)
4. **Επιλέξτε τον σωστό τρόπο λειτουργίας**
   - Χρησιμοποιήστε δυαδική λειτουργία ('b') για μη κειμενικά αρχεία
   - (Use binary mode ('b') for non-text files)
5. **Διαχείριση μεγάλων αρχείων**
   - Χρησιμοποιήστε επανάληψη αντί για read() για μεγάλα αρχεία
   - (Use iteration instead of read() for large files)

In [ ]:

Python-libmagic και MIME Types

Η βιβλιοθήκη python-libmagic είναι η Python διεπαφή για το libmagic, που χρησιμοποιείται για την αναγνώριση τύπων αρχείων και MIME types. Το libmagic προέρχεται από την εντολή `file` του Unix, που αναπτύχθηκε στο Bell Labs τη δεκαετία του 1970.

- Αρχικός σκοπός: Αναγνώριση τύπων αρχείων χωρίς εξάρτηση από την κατάληξή τους
- Προέλευση ονόματος: "magic numbers" - χαρακτηριστικές ακολουθίες bytes στην αρχή των αρχείων

MIME Types - Τι είναι; (What are MIME Types?)

Τα MIME types (Multipurpose Internet Mail Extensions) είναι πρότυπα που δείχνουν τον τύπο του περιεχομένου ενός αρχείου. Παραδείγματα / Examples:

- text/plain: Απλό κείμενο / Plain text
- image/jpeg: Εικόνα JPEG / JPEG image
- application/pdf: Αρχείο PDF / PDF file
- application/json: Αρχείο JSON / JSON file

Βασική Χρήση (Basic Usage)

```
import magic

# Δημιουργία αντικειμένου Magic / Create Magic object
mime = magic.Magic(mime=True)

# Έλεγχος τύπου αρχείου / Check file type
filename = "example.pdf"
mime_type = mime.from_file(filename)
print(f"MIME type: {mime_type}")  # Θα εμφανίσει: application/pdf

# Έλεγχος από περιεχόμενα buffer / Check from buffer content
with open(filename, 'rb') as f:
    content = f.read()
    mime_type = mime.from_buffer(content)

#Λεπτομερής αναγνώριση / Detailed recognition
detailed_magic = magic.Magic(mime=False)
description = detailed_magic.from_file('document.pdf')
```

In [ ]:

In [ ]:

In [ ]:

Εξαιρέσεις στην Python (Python Exceptions)

Οι εξαιρέσεις είναι σφάλματα που συμβαίνουν κατά την εκτέλεση του προγράμματος. Η Python παρέχει ένα ισχυρό μηχανισμό για τον χειρισμό τους.

Βασικές Έννοιες

Τι είναι μια εξαίρεση;

Μια εξαίρεση είναι ένα γεγονός που διακόπτει την κανονική ροή του προγράμματος. Συμβαίνει όταν προκύπτει ένα σφάλμα κατά την εκτέλεση.

Συχνές Εξαιρέσεις (Common Exceptions)

```
# TypeError – Σφάλμα Τύπου
x = "5" + 5  # Λάθος: δεν μπορούμε να προσθέσουμε string με αριθμό
             # Error: cannot add string and number

# ZeroDivisionError – Διαίρεση με το μηδέν
x = 10 / 0   # Λάθος: διαίρεση με το μηδέν
             # Error: division by zero

# IndexError – Σφάλμα Δείκτη
list = [1, 2, 3]
print(list[5])  # Λάθος: ο δείκτης 5 δεν υπάρχει
                # Error: index 5 is out of range
```

Χειρισμός Εξαιρέσεων (Exception Handling)

Η δομή try-except (The try-except structure)

```
try:
    # Κώδικας που μπορεί να προκαλέσει εξαίρεση
    # Code that might raise an exception
    number = int(input("Δώσε έναν αριθμό: "))

except ValueError:
    # Εκτελείται αν προκύψει ValueError
    # Executed if ValueError occurs
    print("Αυτό δεν είναι έγκυρος αριθμός!")
    print("This is not a valid number!")
```

Πολλαπλές εξαιρέσεις (Multiple exceptions)

```
try:
    number = int(input("Δώσε αριθμό: "))
    result = 10 / number

except ValueError:
    print("Μη έγκυρος αριθμός!")
    print("Invalid number!")

except ZeroDivisionError:
    print("Δεν γίνεται διαίρεση με το μηδέν!")
    print("Division by zero is not allowed!")

except:  # Πιάνει όλες τις άλλες εξαιρέσεις / Catches all other exceptions
    print("Κάτι πήγε στραβά!")
    print("Something went wrong!")
```

Η εντολή else (The else statement)

```
try:
    number = int(input("Δώσε αριθμό: "))
except ValueError:
    print("Μη έγκυρη είσοδος!")
    print("Invalid input!")
else:
    # Εκτελείται μόνο αν δεν προκύψει εξαίρεση
    # Executed only if no exception occurred
    print(f"Έδωσες τον αριθμό: {number}")
    print(f"You entered the number: {number}")
```

Η εντολή finally (The finally statement)

```
try:
    file = open("test.txt")
    # κάποιες λειτουργίες με το αρχείο
    # some operations with the file
except FileNotFoundError:
    print("Το αρχείο δεν βρέθηκε!")
    print("File not found!")
finally:
    # Εκτελείται πάντα, είτε υπάρχει εξαίρεση είτε όχι
    # Always executed, whether there was an exception or not
    file.close()
```

Δημιουργία Προσαρμοσμένων Εξαιρέσεων (Creating Custom Exceptions)

```
class NegativeNumberError(Exception):
    """Εξαίρεση για αρνητικούς αριθμούς"""
    pass

def check_num(num):
    if num < 0:
        raise NegativeNumberError("Ο αριθμός δεν μπορεί να είναι
αρνητικός!")
    return num

# Χρήση / Usage
try:
    print(check_num(-5))
except NegativeNumberError as e:
    print(f"Σφάλμα: {e}")
    print(f"Error: {e}")
```

Καλές Πρακτικές (Best Practices)

1. Χρησιμοποιείτε συγκεκριμένες εξαιρέσεις αντί για γενικό except Use specific exceptions instead of bare except
2. Κρατήστε το μπλοκ try όσο πιο μικρό γίνεται Keep the try block as small as possible
3. Χρησιμοποιήστε finally για καθαρισμό πόρων Use finally for cleanup operations
4. Τεκμηριώστε τις προσαρμοσμένες εξαιρέσεις Document your custom exceptions

In [ ]: