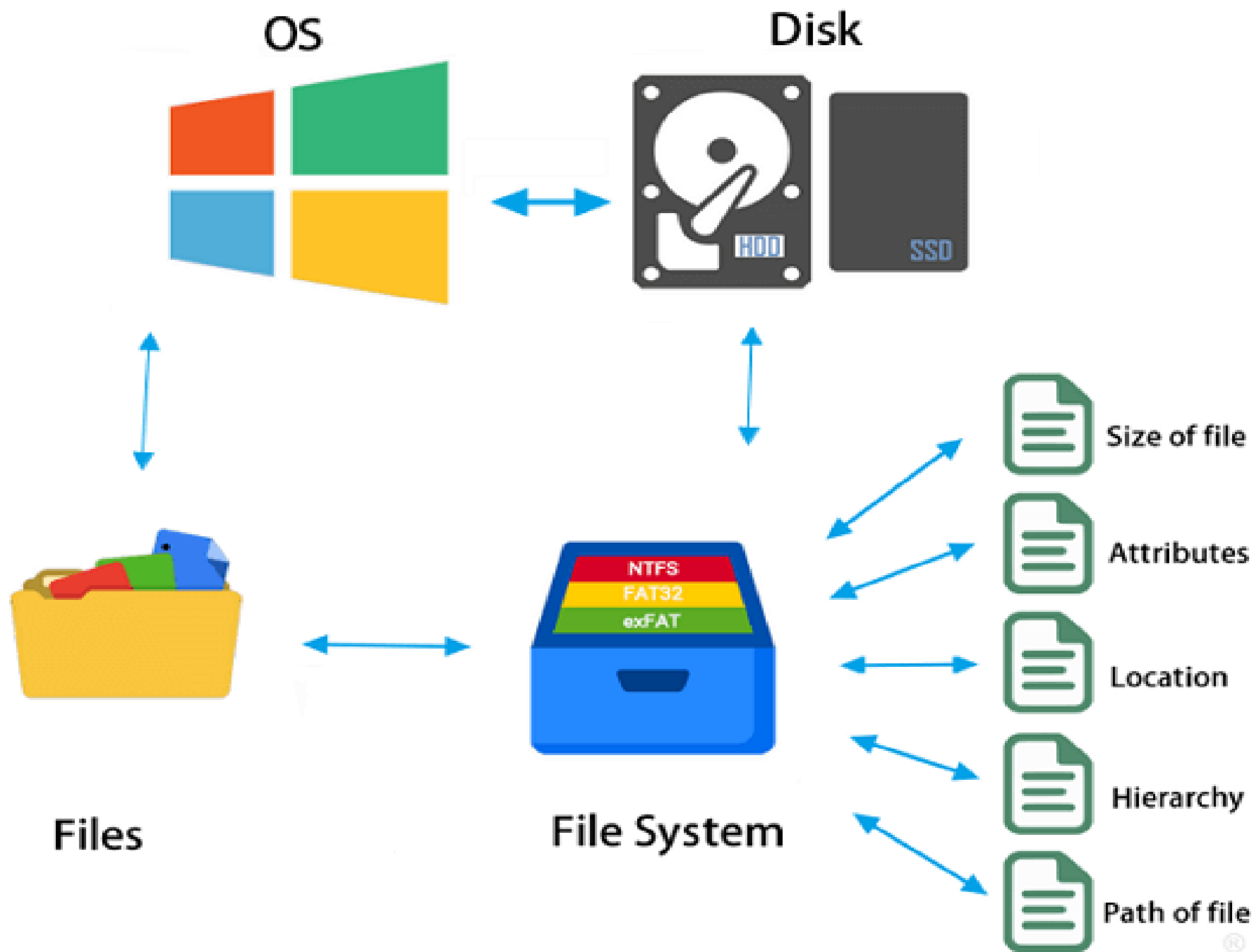




PYTHON

File I/O



ფაილების სისტემა

ფაილური სისტემა არის მეთოდი და სტრუქტურა, რომელსაც ოპერაციული სისტემა იყენებს ფაილების შესანახად, ორგანიზებისა და მართვისთვის შესანახ საშუალებაზე (როგორიცაა მყარი დისკი, SSD და ა.შ.). ფაილური სისტემა უზრუნველყოფს ფაილების შექმნის, წაკითხვის, ჩაწერისა და ორგანიზების მექანიზმებს, რაც უზრუნველყოფს, რომ მომხმარებლებს და პროგრამებს შეუძლიათ მონაცემებზე ეფექტური წვდომა და მანიპულირება.

file system

ფაილი - არის ფაილური სისტემის შენახვის ძირითადი ერთეული. ის შეიძლება შეიცავდეს მონაცემებს, ტექსტს, კოდს ან სხვა სახის ინფორმაციას. ფაილების სახელები არის ადამიანის მიერ წასაკითხი ლეიბლები, რომლებიც მოცემულია ფაილებზე (მაგ., document.txt, image.jpg).

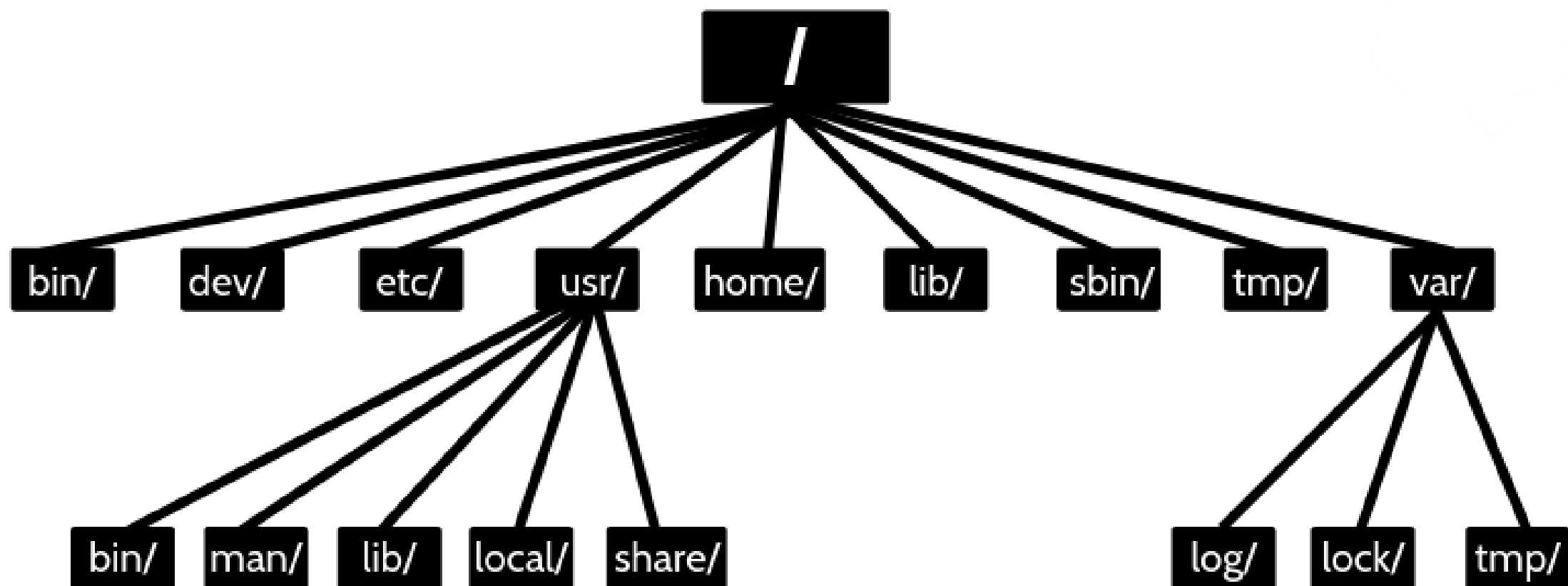
დირექტორია (ან საქაღალდე) - არის სპეციალური ტიპის ფაილი, რომელიც შეიცავს სხვა ფაილების ან დირექტორიების სიას. დირექტორიები საშუალებას აძლევს ფაილების ორგანიზებას იერარქიულ სტრუქტურაში, სადაც დირექტორიები შეიცავს ქვედირექტორიებსა და ფაილებს, რომლებიც ქმნიან ხის მსგავს სტრუქტურას.

ფაილური სისტემების ძირითადი ცნებები



file path - არის ფაილის ან დირექტორიის ადგილმდებარეობა ფაილურ სისტემაში. ის მიუთითებს სად ინახება ფაილი შესაძლო მოწყობილობაზე. Absolute Path: განსაზღვრავს სრულ გზას root დირექტორიიდან (მაგ., /home/user/documents/file.txt). Relative path: განსაზღვრავს მდებარეობას მიმდინარე სამუშაო დირექტორიასთან (მაგ., დოკუმენტები/file.txt) მიმართებაში.

ფაილის გაფართოებები - ფაილის ტიპის მითითებისთვის გამოიყენება ფაილის გაფართოება (მაგ., .txt, .jpg, .py)

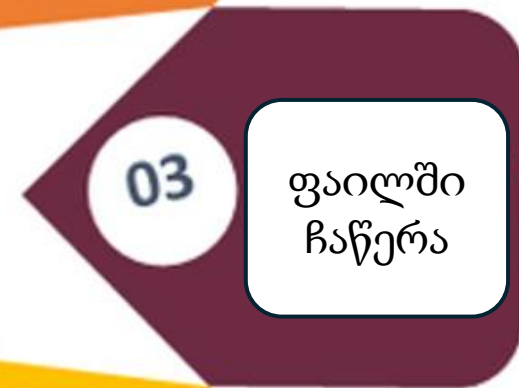


ფაილის ფორმატის რეალიზაციის

მახასიათებლები ეხება ტექნიკურ ასპექტებს, თუ როგორ ხდება მონაცემების ორგანიზება, შენახვა და წარმოდგენილი ფაილში. ეს ფუნქციები გავლენას ახდენს ფაილის გამოყენებადობაზე, ეფექტურობაზე, თავსებადობასა და უსაფრთხოებაზე. ამ მახასიათებლების გაგება გადამწყვეტია ფაილის სხვადასხვა ტიპებთან მუშაობისას, იქნება ეს ტექსტთან, მედიასთან თუ სპეციალიზებულ ფორმატებთან, როგორიცაა მონაცემთა ბაზები და ცხრილები.



File handling in Python



File.read()


python

 Copy

```
# Open the file in read mode ('r')
with open('filename.txt', 'r') as file:
    content = file.read() # Read the entire content of the file
    print(content)         # Print the content
```


File.write()

python

 Copy

```
# Open the file in exclusive creation mode ('x')
try:
    with open('filename.txt', 'x') as file:
        file.write("This file was created and written to using the 'x' mode.\n")
        file.write("If the file already existed, this would raise a FileExistsError.\n")
except FileExistsError:
    print("Error: The file already exists.")
```

python

 Copy

```
with open('example.txt', 'w') as file:  
    file.write("Hello, world!\n")  
    file.write("This is a Python file writing example.\n")  
    file.write("Goodbye!")
```

python

 Copy

```
with open('example.txt', 'a') as file:  
    file.write("Appending this line.\n")
```

Insert()

python

 Copy

```
# Insert text at the end
new_line = "This is the last line.\n"
position = len(lines) # Position after the last line

# Read the file
with open('example.txt', 'r') as file:
    lines = file.readlines()

# Insert the new line at the end
lines.insert(position, new_line) # or lines.append(new_line)

# Write back to the file
with open('example.txt', 'w') as file:
    file.writelines(lines)
```



del

python

 Copy

```
# Define the file path and the position to delete
file_path = 'example.txt'
position_to_delete = 1 # Position of the line to delete (0-based index)


# Read the file
with open(file_path, 'r') as file:
    lines = file.readlines() # Read all lines into a list

# Delete the line at the specified position
del lines[position_to_delete]

# Write the modified content back to the file
with open(file_path, 'w') as file:
    file.writelines(lines) # Write the updated lines back to the file
```

File.close()

python

 Copy

```
file = open('example.txt', 'r')  
content = file.read()  
print(content)  
file.close()
```

Python Context Manager





არის რესურსების მართვის საშუალება (როგორცაა ფაილების დამუშავება, მონაცემთა ბაზის კავშირები და ა.შ.) ის ავტომატურად ახორციელებს ოპერაციებს, რაც ამცირებს რესურსების ხელით მართვის საჭიროებას..

https://book.pythontips.com/en/latest/context_managers.html

<https://docs.python.org/3/library/contextlib.html>

with



python

 Copy

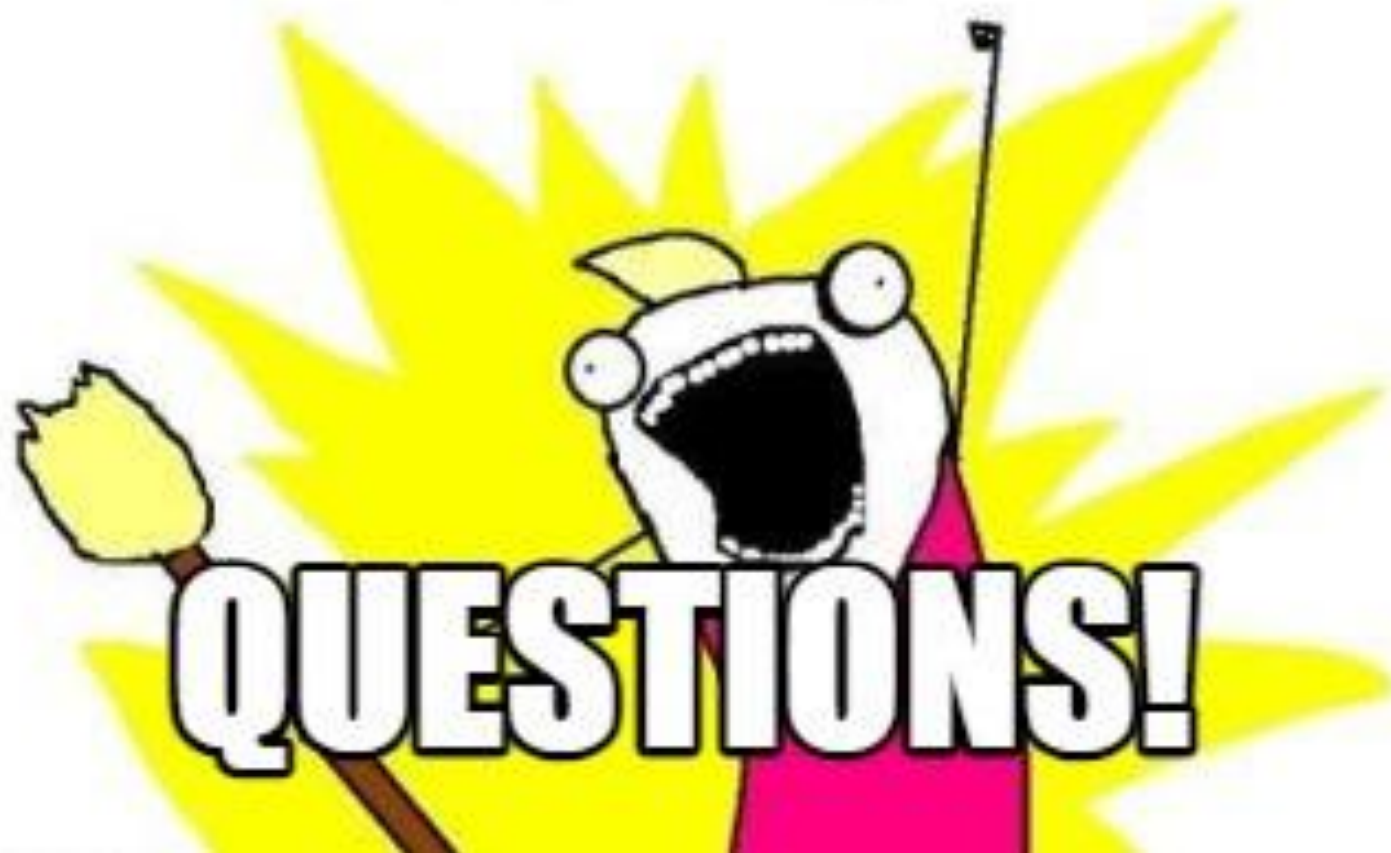
```
from contextlib import contextmanager

@contextmanager
def open_file(filename, mode):
    file = open(filename, mode)
    yield file
    file.close()

with open_file('example.txt', 'w') as file:
    file.write("Hello from the context manager!")

with open_file('example.txt', 'r') as file:
    print(file.read())
```


TIME FOR



QUESTIONS!