## **We have many chunks of information which we should divide into pages in order to make it easy to navigate content like search results, product listings or articles. We use **math** library to find the nearest integer number of pages.**

## ****Step 1 & 2: Инициализация класса****

class Pagination:

def \_\_init\_\_(self, items=None, page\_size=10):

self.items = items if items is not None else []

self.page\_size = page\_size

self.current\_idx = 0

self.total\_pages = math.ceil(len(self.items) / self.page\_size) if self.items else 0

The math.ceil() function in Python returns the ceiling of a number, which is the smallest integer greater than or equal to the given number.

* **items** — список элементов, которые нужно разбить на страницы.  
  Если None, создаём пустой список.
* **page\_size** — количество элементов на одной странице.
* **current\_idx** — индекс текущей страницы (от 0).
* **total\_pages** — общее число страниц, округлённое вверх через math.ceil.

Пример: если 26 букв и page\_size = 4 → total\_pages = 7 (7 страниц: 6 страниц по 4 элемента + 1 страница с 2 элементами).

## ****Step 3: Получение элементов текущей страницы****

def get\_visible\_items(self):

start = self.current\_idx \* self.page\_size

end = start + self.page\_size

return self.items[start:end]

* Вычисляем **начальный и конечный индекс** текущей страницы.
* Возвращаем **срез списка**, который соответствует текущей странице.

Пример: current\_idx = 0 и page\_size = 4 → элементы с 0 по 3 (items[0:4]).

## ****Step 4: Навигация по страницам****

def go\_to\_page(self, page\_num):

if page\_num < 1 or page\_num > self.total\_pages:

raise ValueError(f"Page number {page\_num} out of range. Total pages: {self.total\_pages}")

self.current\_idx = page\_num - 1

* Принимает номер страницы **1-based** (пользователь видит страницы с 1, а не с 0).
* Проверяет, что номер в допустимом диапазоне.
* Если всё верно, устанавливает current\_idx = page\_num - 1.

Другие методы навигации:

def first\_page(self):

self.current\_idx = 0

return self

def last\_page(self):

self.current\_idx = self.total\_pages - 1

return self

def next\_page(self):

if self.current\_idx < self.total\_pages - 1:

self.current\_idx += 1

return self

def previous\_page(self):

if self.current\_idx > 0:

self.current\_idx -= 1

return self

* **first\_page()** — переход на первую страницу.
* **last\_page()** — переход на последнюю страницу.
* **next\_page()** — переход на следующую страницу, если ещё не последняя.
* **previous\_page()** — переход на предыдущую страницу, если не первая.
* Возвращаем self для **цепочек вызовов**:
* p.first\_page().next\_page().next\_page()

## ****Step 5: Вывод страницы****

def \_\_str\_\_(self):

return "\n".join(self.get\_visible\_items())

* Возвращает строку, где каждый элемент текущей страницы на новой строке.
* Например, для первой страницы с буквами ['a','b','c','d']:
* a
* b
* c
* d

## ****Step 6: Тестирование****

alphabetList = list("abcdefghijklmnopqrstuvwxyz")

p = Pagination(alphabetList, 4)

print(p.get\_visible\_items()) # ['a', 'b', 'c', 'd']

p.next\_page()

print(p.get\_visible\_items()) # ['e', 'f', 'g', 'h']

p.last\_page()

print(p.get\_visible\_items()) # ['y', 'z']

p.go\_to\_page(7)

print(p.current\_idx + 1) # 7

p.go\_to\_page(0) # Raises ValueError

* Проверяем работу методов: переход между страницами, получение элементов, обработку ошибок.

### ****Итог:****

1. Pagination разбивает список на страницы по заданному размеру.
2. Можно переходить между страницами разными способами (go\_to\_page, next\_page, previous\_page, first\_page, last\_page).
3. Метод \_\_str\_\_() красиво выводит текущую страницу.
4. Все индексы страниц **внутри 0-based**, а пользователь работает с **1-based**.