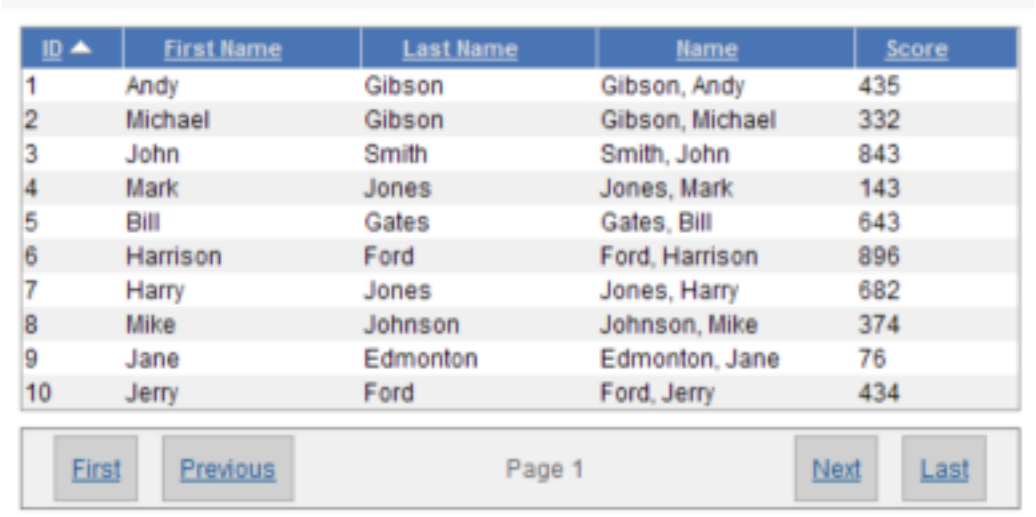
Python Lab (6)

1) Pagination Class with OOP

Your task is to create a class to handle paginated content in a website. A pagination is used to divide long lists of content in a series of pages.



ID ▲	First Name	Last Name	Name	Score
1	Andy	Gibson	Gibson, Andy	435
2	Michael	Gibson	Gibson, Michael	332
3	John	Smith	Smith, John	843
4	Mark	Jones	Jones, Mark	143
5	Bill	Gates	Gates, Bill	643
6	Harrison	Ford	Ford, Harrison	896
7	Harry	Jones	Jones, Harry	682
8	Mike	Johnson	Johnson, Mike	374
9	Jane	Edmonton	Edmonton, Jane	76
10	Jerry	Ford	Ford, Jerry	434

The pagination class will accept 2 parameters:

1. **items** (default: []): A list of contents to paginate.
2. **pageSize** (default: 10): The amount of items to show in each

page. So, for example we could initialize our pagination like this:

```
alphabetList = "abcdefghijklmnopqrstuvwxyz".split('')

p = Pagination(alphabetList, 4)
```

And then use the method getVisibleItems to show the contents of the paginated list.

```
p.getVisibleItems() # ["a", "b", "c", "d"]
```

You will have to implement various methods to go through the pages such

as: • prevPage

• nextPage

- firstPage
- lastPage
- goToPage

Here's a continuation of the example above using nextPage and lastPage:

```
p.nextPage()

p.getVisibleItems()
# ["e", "f", "g", "h"]

p.lastPage()

p.getVisibleItems()
# ["y", "z"]
```

Notes

- The second argument (`pageSize`) could be a float, in that case just convert it to an int (this is also the case for the `goToPage` method)
- The methods used to change page should be chainable, so you can call them one after the other like this: `p.nextPage().nextPage()`
- Please set the `p.totalPages` and `p.currentPage` attributes to the appropriate number as there cannot be a page 0.
- If a page is outside of the `totalPages` attribute, then the `goToPage` method should go to the closest page to the number provided (e.g. there are only 5 total pages, but `p.goToPage(10)` is given: the `p.currentPage` should be set to 5; if 0 or a negative number is given, `p.currentPage` should be set to 1).

2) Coffee Shop

Write a class called CoffeeShop, which has **three instance variables**:

1. name: a string (basically, of the shop)
2. menu: a list of items (of dict type), with each item containing the item (name of the item), type (whether a *food* or a *drink*) and price.
3. orders: an empty list

and **seven methods**:

1. **add_order**: adds the **name** of the item to the end of the **orders** list if it exists on the **menu**, otherwise, return "This item is currently unavailable!"
2. **fulfill_order**: if the **orders** list is **not empty**, return "The {item} is ready!". If the **orders** list is empty, return "All orders have been fulfilled!"
3. **list_orders**: returns the *item* **names** of the **orders** taken, otherwise, an **empty** list.
4. **due_amount**: returns the total amount due for the **orders** taken. 5. **cheapest_item**: returns the **name** of the cheapest item on the menu. 6. **drinks_only**: returns only the *item* **names** of *type* **drink** from the menu. 7. **food_only**: returns only the *item* **names** of *type* **food** from the menu.

IMPORTANT: Orders are fulfilled in a **FIFO** (first-in, first-out) order.

```
tcs.add_order("hot cocoa") → "This item is currently unavailable!"
# Tesha's coffee shop does not sell hot cocoa
tcs.add_order("iced tea") → "This item is currently unavailable!"
# specifying the variant of "iced tea" will help the process

tcs.add_order("cinnamon roll") →  "Order added!"
tcs.add_order("iced coffee") → "Order added!"
tcs.list_orders → ["cinnamon roll", "iced coffee"]
# all items of the current order

tcs.due_amount() → 2.17

tcs.fulfill_order() → "The cinnamon roll is ready!"
tcs.fulfill_order() → "The iced coffee is ready!"
tcs.fulfill_order() → "All orders have been fulfilled!"
# all orders have been presumably served

tcs.list_orders() → []
# an empty list is returned if all orders have been exhausted

tcs.due_amount() → 0.0
# no new orders taken, expect a zero payable

tcs.cheapest_item() → "lemonade"
tcs.drinks_only() → ["orange juice", "lemonade", "cranberry juice", "pineapple
tcs.food_only() → ["tuna sandwich", "ham and cheese sandwich", "bacon and egg",
```