

```
20231103 > exam > _30_1.py > ...
1 class Tmoney: "Tmoney: Unknown word."
2
3     def __init__(self, deposit):
4         self.deposit = deposit
5
6     def charge(self, deposit):
7         self.deposit += deposit
8
9     def use(self, deposit):
10        if self.deposit >= deposit:
11            self.deposit -= deposit
12
13    def __repr__(self):
14        return self.deposit
15
16    def __str__(self):
17        return self.deposit
18
19
20
21
22 class Tmoney2: "Tmoney: Unknown word."
23     deposit = 0
24
25     @classmethod
26     def charge(cls, deposit):
27         cls.deposit += deposit
28
29     @classmethod
30     def use(cls, deposit):
31         if cls.deposit >= deposit:
32             cls.deposit -= deposit
33
34 a, b = Tmoney(0), Tmoney(0) "Tmoney: Unknown word."
35 c, d = Tmoney2(0), Tmoney2(0) "Tmoney: Unknown word."
36 c.charge(10000)
37 print(d.deposit)
38 d.use(1000)
39 print(c.deposit)
40
41 a.charge(10000)
42 b.charge(10000)
43
44 # print(a.deposit)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

2 files changed, 76 insertions(+), 19 deletions(-)

git push

Enumerating objects: 11, done.

Counting objects: 100% (11/11), done.

Delta compression using up to 10 threads

Compressing objects: 100% (6/6), done.

Writing objects: 100% (6/6), 1.24 KiB | 1.24 MiB/s, done.

Total 6 (delta 4), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (4/4), completed with 4 local objects.

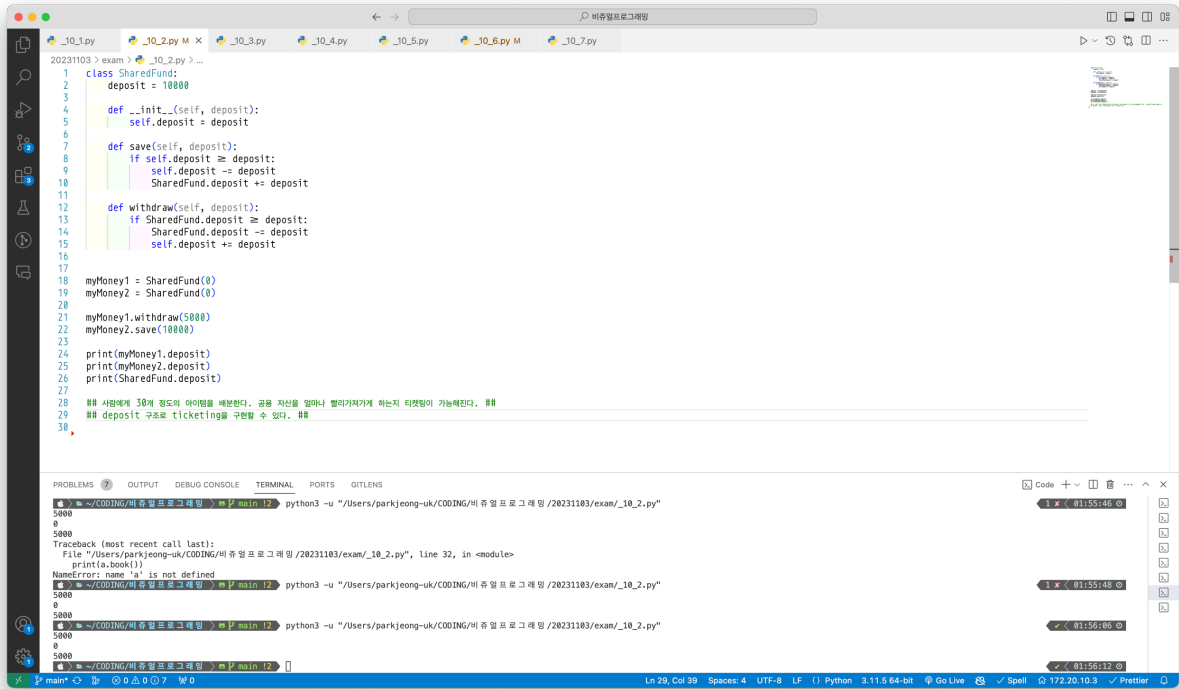
To https://github.com/000up/visual_programming.git

8077d61..d0be138 main -> main

python3 -u "/Users/parkjeong-uk/CODING/H

18000

Ln 48, Col 14, Spaces: 4 UTF-8 LF Python 3.11.5 64-bit Go Live Spell 172.20.10.3 Premier



```
20231103 > exam > _30_3.py >
1 class Course:
2     # 클래스에서 객체 하나를 생성할때 가변 객체 참조,
3     def __init__(self, name, code, grade):
4         self.name = name
5         self.code = code
6         self.grade = grade
7         match grade:
8             case 4.5:
9                 gpa = 4.5
10            case 4:
11                gpa = 4
12            case 3.5:
13                gpa = 3.5
14            case 3:
15                gpa = 3
16            case 2.5:
17                gpa = 2.5
18            case 2:
19                gpa = 2
20            case 1.5:
21                gpa = 1.5
22            case 1:
23                gpa = 1
24            case 0.5:
25                gpa = 0.5
26            case 0:
27                gpa = 0
28            case _:
29                gpa = 0
30        self.gpa = gpa
31
32    def __repr__(self):
33        return str(self)
34
35    def __str__(self):
36        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
37
38    def __format__(self, format):
39        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
40
41    def __len__(self):
42        return len(self.name)
43
44    def __getitem__(self, item):
45        return self.__dict__[item]
46
47    def __setitem__(self, item, value):
48        self.__dict__[item] = value
49
50    def __delitem__(self, item):
51        del self.__dict__[item]
52
53    def __contains__(self, item):
54        return item in self.__dict__
55
56    def __iter__(self):
57        return iter(self.__dict__.items())
58
59    def __reversed__(self):
60        return reversed(self.__dict__.items())
61
62    def __hash__(self):
63        return hash(self.name)
64
65    def __eq__(self, other):
66        return self.name == other.name
67
68    def __lt__(self, other):
69        return self.name < other.name
70
71    def __le__(self, other):
72        return self.name <= other.name
73
74    def __gt__(self, other):
75        return self.name > other.name
76
77    def __ge__(self, other):
78        return self.name >= other.name
79
80    def __ne__(self, other):
81        return self.name != other.name
82
83    def __and__(self, other):
84        return self.name & other.name
85
86    def __or__(self, other):
87        return self.name | other.name
88
89    def __xor__(self, other):
90        return self.name ^ other.name
91
92    def __radd__(self, other):
93        return self.name + other.name
94
95    def __rsub__(self, other):
96        return self.name - other.name
97
98    def __rmul__(self, other):
99        return self.name * other.name
100    def __rdiv__(self, other):
101        return self.name / other.name
102    def __rfloordiv__(self, other):
103        return self.name // other.name
104    def __rmod__(self, other):
105        return self.name % other.name
106    def __rpow__(self, other):
107        return self.name ** other.name
108    def __lshift__(self, other):
109        return self.name << other.name
110    def __rshift__(self, other):
111        return self.name >> other.name
112    def __bitand__(self, other):
113        return self.name & other.name
114    def __bitor__(self, other):
115        return self.name | other.name
116    def __bitxor__(self, other):
117        return self.name ^ other.name
118    def __bitnot__(self):
119        return ~self.name
120
121    def __call__(self):
122        return self.name
123
124    def __getattr__(self, name):
125        return self.__dict__[name]
126
127    def __setattr__(self, name, value):
128        self.__dict__[name] = value
129
130    def __delattr__(self, name):
131        del self.__dict__[name]
132
133    def __copy__(self):
134        return self.__dict__.copy()
135
136    def __deepcopy__(self, memo):
137        return self.__dict__.copy()
138
139    def __reduce__(self):
140        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
141
142    def __reduce_ex__(self, protocol):
143        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
144
145    def __sizeof__(self):
146        return len(self.__dict__)
147
148    def __dir__(self):
149        return dir(self.__dict__)
150
151    def __doc__(self):
152        return self.__dict__
153
154    def __repr__(self):
155        return str(self)
156
157    def __str__(self):
158        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
159
160    def __format__(self, format):
161        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
162
163    def __len__(self):
164        return len(self.name)
165
166    def __getitem__(self, item):
167        return self.__dict__[item]
168
169    def __setitem__(self, item, value):
170        self.__dict__[item] = value
171
172    def __delitem__(self, item):
173        del self.__dict__[item]
174
175    def __contains__(self, item):
176        return item in self.__dict__
177
178    def __iter__(self):
179        return iter(self.__dict__.items())
180
181    def __reversed__(self):
182        return reversed(self.__dict__.items())
183
184    def __hash__(self):
185        return hash(self.name)
186
187    def __eq__(self, other):
188        return self.name == other.name
189
190    def __lt__(self, other):
191        return self.name < other.name
192
193    def __le__(self, other):
194        return self.name <= other.name
195
196    def __gt__(self, other):
197        return self.name > other.name
198
199    def __ge__(self, other):
200        return self.name >= other.name
201
202    def __ne__(self, other):
203        return self.name != other.name
204
205    def __and__(self, other):
206        return self.name & other.name
207
208    def __or__(self, other):
209        return self.name | other.name
210
211    def __xor__(self, other):
212        return self.name ^ other.name
213
214    def __radd__(self, other):
215        return self.name + other.name
216
217    def __rsub__(self, other):
218        return self.name - other.name
219
220    def __rmul__(self, other):
221        return self.name * other.name
222    def __rdiv__(self, other):
223        return self.name / other.name
224    def __rfloordiv__(self, other):
225        return self.name // other.name
226    def __rmod__(self, other):
227        return self.name % other.name
228    def __rpow__(self, other):
229        return self.name ** other.name
230    def __lshift__(self, other):
231        return self.name << other.name
232    def __rshift__(self, other):
233        return self.name >> other.name
234    def __bitand__(self, other):
235        return self.name & other.name
236    def __bitor__(self, other):
237        return self.name | other.name
238    def __bitxor__(self, other):
239        return self.name ^ other.name
240    def __bitnot__(self):
241        return ~self.name
242
243    def __call__(self):
244        return self.name
245
246    def __getattr__(self, name):
247        return self.__dict__[name]
248
249    def __setattr__(self, name, value):
250        self.__dict__[name] = value
251
252    def __delattr__(self, name):
253        del self.__dict__[name]
254
255    def __copy__(self):
256        return self.__dict__.copy()
257
258    def __deepcopy__(self, memo):
259        return self.__dict__.copy()
260
261    def __reduce__(self):
262        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
263
264    def __reduce_ex__(self, protocol):
265        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
266
267    def __sizeof__(self):
268        return len(self.__dict__)
269
270    def __dir__(self):
271        return dir(self.__dict__)
272
273    def __doc__(self):
274        return self.__dict__
275
276    def __repr__(self):
277        return str(self)
278
279    def __str__(self):
280        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
281
282    def __format__(self, format):
283        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
284
285    def __len__(self):
286        return len(self.name)
287
288    def __getitem__(self, item):
289        return self.__dict__[item]
290
291    def __setitem__(self, item, value):
292        self.__dict__[item] = value
293
294    def __delitem__(self, item):
295        del self.__dict__[item]
296
297    def __contains__(self, item):
298        return item in self.__dict__
299
300    def __iter__(self):
301        return iter(self.__dict__.items())
302
303    def __reversed__(self):
304        return reversed(self.__dict__.items())
305
306    def __hash__(self):
307        return hash(self.name)
308
309    def __eq__(self, other):
310        return self.name == other.name
311
312    def __lt__(self, other):
313        return self.name < other.name
314
315    def __le__(self, other):
316        return self.name <= other.name
317
318    def __gt__(self, other):
319        return self.name > other.name
320
321    def __ge__(self, other):
322        return self.name >= other.name
323
324    def __ne__(self, other):
325        return self.name != other.name
326
327    def __and__(self, other):
328        return self.name & other.name
329
330    def __or__(self, other):
331        return self.name | other.name
332
333    def __xor__(self, other):
334        return self.name ^ other.name
335
336    def __radd__(self, other):
337        return self.name + other.name
338
339    def __rsub__(self, other):
340        return self.name - other.name
341
342    def __rmul__(self, other):
343        return self.name * other.name
344    def __rdiv__(self, other):
345        return self.name / other.name
346    def __rfloordiv__(self, other):
347        return self.name // other.name
348    def __rmod__(self, other):
349        return self.name % other.name
350    def __rpow__(self, other):
351        return self.name ** other.name
352    def __lshift__(self, other):
353        return self.name << other.name
354    def __rshift__(self, other):
355        return self.name >> other.name
356    def __bitand__(self, other):
357        return self.name & other.name
358    def __bitor__(self, other):
359        return self.name | other.name
360    def __bitxor__(self, other):
361        return self.name ^ other.name
362    def __bitnot__(self):
363        return ~self.name
364
365    def __call__(self):
366        return self.name
367
368    def __getattr__(self, name):
369        return self.__dict__[name]
370
371    def __setattr__(self, name, value):
372        self.__dict__[name] = value
373
374    def __delattr__(self, name):
375        del self.__dict__[name]
376
377    def __copy__(self):
378        return self.__dict__.copy()
379
380    def __deepcopy__(self, memo):
381        return self.__dict__.copy()
382
383    def __reduce__(self):
384        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
385
386    def __reduce_ex__(self, protocol):
387        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
388
389    def __sizeof__(self):
390        return len(self.__dict__)
391
392    def __dir__(self):
393        return dir(self.__dict__)
394
395    def __doc__(self):
396        return self.__dict__
397
398    def __repr__(self):
399        return str(self)
400
401    def __str__(self):
402        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
403
404    def __format__(self, format):
405        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
406
407    def __len__(self):
408        return len(self.name)
409
410    def __getitem__(self, item):
411        return self.__dict__[item]
412
413    def __setitem__(self, item, value):
414        self.__dict__[item] = value
415
416    def __delitem__(self, item):
417        del self.__dict__[item]
418
419    def __contains__(self, item):
420        return item in self.__dict__
421
422    def __iter__(self):
423        return iter(self.__dict__.items())
424
425    def __reversed__(self):
426        return reversed(self.__dict__.items())
427
428    def __hash__(self):
429        return hash(self.name)
430
431    def __eq__(self, other):
432        return self.name == other.name
433
434    def __lt__(self, other):
435        return self.name < other.name
436
437    def __le__(self, other):
438        return self.name <= other.name
439
440    def __gt__(self, other):
441        return self.name > other.name
442
443    def __ge__(self, other):
444        return self.name >= other.name
445
446    def __ne__(self, other):
447        return self.name != other.name
448
449    def __and__(self, other):
450        return self.name & other.name
451
452    def __or__(self, other):
453        return self.name | other.name
454
455    def __xor__(self, other):
456        return self.name ^ other.name
457
458    def __radd__(self, other):
459        return self.name + other.name
460
461    def __rsub__(self, other):
462        return self.name - other.name
463
464    def __rmul__(self, other):
465        return self.name * other.name
466    def __rdiv__(self, other):
467        return self.name / other.name
468    def __rfloordiv__(self, other):
469        return self.name // other.name
470    def __rmod__(self, other):
471        return self.name % other.name
472    def __rpow__(self, other):
473        return self.name ** other.name
474    def __lshift__(self, other):
475        return self.name << other.name
476    def __rshift__(self, other):
477        return self.name >> other.name
478    def __bitand__(self, other):
479        return self.name & other.name
480    def __bitor__(self, other):
481        return self.name | other.name
482    def __bitxor__(self, other):
483        return self.name ^ other.name
484    def __bitnot__(self):
485        return ~self.name
486
487    def __call__(self):
488        return self.name
489
490    def __getattr__(self, name):
491        return self.__dict__[name]
492
493    def __setattr__(self, name, value):
494        self.__dict__[name] = value
495
496    def __delattr__(self, name):
497        del self.__dict__[name]
498
499    def __copy__(self):
500        return self.__dict__.copy()
501
502    def __deepcopy__(self, memo):
503        return self.__dict__.copy()
504
505    def __reduce__(self):
506        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
507
508    def __reduce_ex__(self, protocol):
509        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
510
511    def __sizeof__(self):
512        return len(self.__dict__)
513
514    def __dir__(self):
515        return dir(self.__dict__)
516
517    def __doc__(self):
518        return self.__dict__
519
520    def __repr__(self):
521        return str(self)
522
523    def __str__(self):
524        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
525
526    def __format__(self, format):
527        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
528
529    def __len__(self):
530        return len(self.name)
531
532    def __getitem__(self, item):
533        return self.__dict__[item]
534
535    def __setitem__(self, item, value):
536        self.__dict__[item] = value
537
538    def __delitem__(self, item):
539        del self.__dict__[item]
540
541    def __contains__(self, item):
542        return item in self.__dict__
543
544    def __iter__(self):
545        return iter(self.__dict__.items())
546
547    def __reversed__(self):
548        return reversed(self.__dict__.items())
549
550    def __hash__(self):
551        return hash(self.name)
552
553    def __eq__(self, other):
554        return self.name == other.name
555
556    def __lt__(self, other):
557        return self.name < other.name
558
559    def __le__(self, other):
560        return self.name <= other.name
561
562    def __gt__(self, other):
563        return self.name > other.name
564
565    def __ge__(self, other):
566        return self.name >= other.name
567
568    def __ne__(self, other):
569        return self.name != other.name
570
571    def __and__(self, other):
572        return self.name & other.name
573
574    def __or__(self, other):
575        return self.name | other.name
576
577    def __xor__(self, other):
578        return self.name ^ other.name
579
580    def __radd__(self, other):
581        return self.name + other.name
582
583    def __rsub__(self, other):
584        return self.name - other.name
585
586    def __rmul__(self, other):
587        return self.name * other.name
588    def __rdiv__(self, other):
589        return self.name / other.name
590    def __rfloordiv__(self, other):
591        return self.name // other.name
592    def __rmod__(self, other):
593        return self.name % other.name
594    def __rpow__(self, other):
595        return self.name ** other.name
596    def __lshift__(self, other):
597        return self.name << other.name
598    def __rshift__(self, other):
599        return self.name >> other.name
600    def __bitand__(self, other):
601        return self.name & other.name
602    def __bitor__(self, other):
603        return self.name | other.name
604    def __bitxor__(self, other):
605        return self.name ^ other.name
606    def __bitnot__(self):
607        return ~self.name
608
609    def __call__(self):
610        return self.name
611
612    def __getattr__(self, name):
613        return self.__dict__[name]
614
615    def __setattr__(self, name, value):
616        self.__dict__[name] = value
617
618    def __delattr__(self, name):
619        del self.__dict__[name]
620
621    def __copy__(self):
622        return self.__dict__.copy()
623
624    def __deepcopy__(self, memo):
625        return self.__dict__.copy()
626
627    def __reduce__(self):
628        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
629
630    def __reduce_ex__(self, protocol):
631        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
632
633    def __sizeof__(self):
634        return len(self.__dict__)
635
636    def __dir__(self):
637        return dir(self.__dict__)
638
639    def __doc__(self):
640        return self.__dict__
641
642    def __repr__(self):
643        return str(self)
644
645    def __str__(self):
646        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
647
648    def __format__(self, format):
649        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
650
651    def __len__(self):
652        return len(self.name)
653
654    def __getitem__(self, item):
655        return self.__dict__[item]
656
657    def __setitem__(self, item, value):
658        self.__dict__[item] = value
659
660    def __delitem__(self, item):
661        del self.__dict__[item]
662
663    def __contains__(self, item):
664        return item in self.__dict__
665
666    def __iter__(self):
667        return iter(self.__dict__.items())
668
669    def __reversed__(self):
670        return reversed(self.__dict__.items())
671
672    def __hash__(self):
673        return hash(self.name)
674
675    def __eq__(self, other):
676        return self.name == other.name
677
678    def __lt__(self, other):
679        return self.name < other.name
680
681    def __le__(self, other):
682        return self.name <= other.name
683
684    def __gt__(self, other):
685        return self.name > other.name
686
687    def __ge__(self, other):
688        return self.name >= other.name
689
690    def __ne__(self, other):
691        return self.name != other.name
692
693    def __and__(self, other):
694        return self.name & other.name
695
696    def __or__(self, other):
697        return self.name | other.name
698
699    def __xor__(self, other):
700        return self.name ^ other.name
701
702    def __radd__(self, other):
703        return self.name + other.name
704
705    def __rsub__(self, other):
706        return self.name - other.name
707
708    def __rmul__(self, other):
709        return self.name * other.name
710    def __rdiv__(self, other):
711        return self.name / other.name
712    def __rfloordiv__(self, other):
713        return self.name // other.name
714    def __rmod__(self, other):
715        return self.name % other.name
716    def __rpow__(self, other):
717        return self.name ** other.name
718    def __lshift__(self, other):
719        return self.name << other.name
720    def __rshift__(self, other):
721        return self.name >> other.name
722    def __bitand__(self, other):
723        return self.name & other.name
724    def __bitor__(self, other):
725        return self.name | other.name
726    def __bitxor__(self, other):
727        return self.name ^ other.name
728    def __bitnot__(self):
729        return ~self.name
730
731    def __call__(self):
732        return self.name
733
734    def __getattr__(self, name):
735        return self.__dict__[name]
736
737    def __setattr__(self, name, value):
738        self.__dict__[name] = value
739
740    def __delattr__(self, name):
741        del self.__dict__[name]
742
743    def __copy__(self):
744        return self.__dict__.copy()
745
746    def __deepcopy__(self, memo):
747        return self.__dict__.copy()
748
749    def __reduce__(self):
750        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
751
752    def __reduce_ex__(self, protocol):
753        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
754
755    def __sizeof__(self):
756        return len(self.__dict__)
757
758    def __dir__(self):
759        return dir(self.__dict__)
760
761    def __doc__(self):
762        return self.__dict__
763
764    def __repr__(self):
765        return str(self)
766
767    def __str__(self):
768        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
769
770    def __format__(self, format):
771        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
772
773    def __len__(self):
774        return len(self.name)
775
776    def __getitem__(self, item):
777        return self.__dict__[item]
778
779    def __setitem__(self, item, value):
780        self.__dict__[item] = value
781
782    def __delitem__(self, item):
783        del self.__dict__[item]
784
785    def __contains__(self, item):
786        return item in self.__dict__
787
788    def __iter__(self):
789        return iter(self.__dict__.items())
790
791    def __reversed__(self):
792        return reversed(self.__dict__.items())
793
794    def __hash__(self):
795        return hash(self.name)
796
797    def __eq__(self, other):
798        return self.name == other.name
799
800    def __lt__(self, other):
801        return self.name < other.name
802
803    def __le__(self, other):
804        return self.name <= other.name
805
806    def __gt__(self, other):
807        return self.name > other.name
808
809    def __ge__(self, other):
810        return self.name >= other.name
811
812    def __ne__(self, other):
813        return self.name != other.name
814
815    def __and__(self, other):
816        return self.name & other.name
817
818    def __or__(self, other):
819        return self.name | other.name
820
821    def __xor__(self, other):
822        return self.name ^ other.name
823
824    def __radd__(self, other):
825        return self.name + other.name
826
827    def __rsub__(self, other):
828        return self.name - other.name
829
830    def __rmul__(self, other):
831        return self.name * other.name
832    def __rdiv__(self, other):
833        return self.name / other.name
834    def __rfloordiv__(self, other):
835        return self.name // other.name
836    def __rmod__(self, other):
837        return self.name % other.name
838    def __rpow__(self, other):
839        return self.name ** other.name
840    def __lshift__(self, other):
841        return self.name << other.name
842    def __rshift__(self, other):
843        return self.name >> other.name
844    def __bitand__(self, other):
845        return self.name & other.name
846    def __bitor__(self, other):
847        return self.name | other.name
848    def __bitxor__(self, other):
849        return self.name ^ other.name
850    def __bitnot__(self):
851        return ~self.name
852
853    def __call__(self):
854        return self.name
855
856    def __getattr__(self, name):
857        return self.__dict__[name]
858
859    def __setattr__(self, name, value):
860        self.__dict__[name] = value
861
862    def __delattr__(self, name):
863        del self.__dict__[name]
864
865    def __copy__(self):
866        return self.__dict__.copy()
867
868    def __deepcopy__(self, memo):
869        return self.__dict__.copy()
870
871    def __reduce__(self):
872        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
873
874    def __reduce_ex__(self, protocol):
875        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
876
877    def __sizeof__(self):
878        return len(self.__dict__)
879
880    def __dir__(self):
881        return dir(self.__dict__)
882
883    def __doc__(self):
884        return self.__dict__
885
886    def __repr__(self):
887        return str(self)
888
889    def __str__(self):
890        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
891
892    def __format__(self, format):
893        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
894
895    def __len__(self):
896        return len(self.name)
897
898    def __getitem__(self, item):
899        return self.__dict__[item]
900
901    def __setitem__(self, item, value):
902        self.__dict__[item] = value
903
904    def __delitem__(self, item):
905        del self.__dict__[item]
906
907    def __contains__(self, item):
908        return item in self.__dict__
909
910    def __iter__(self):
911        return iter(self.__dict__.items())
912
913    def __reversed__(self):
914        return reversed(self.__dict__.items())
915
916    def __hash__(self):
917        return hash(self.name)
918
919    def __eq__(self, other):
920        return self.name == other.name
921
922    def __lt__(self, other):
923        return self.name < other.name
924
925    def __le__(self, other):
926        return self.name <= other.name
927
928    def __gt__(self, other):
929        return self.name > other.name
930
931    def __ge__(self, other):
932        return self.name >= other.name
933
934    def __ne__(self, other):
935        return self.name != other.name
936
937    def __and__(self, other):
938        return self.name & other.name
939
940    def __or__(self, other):
941        return self.name | other.name
942
943    def __xor__(self, other):
944        return self.name ^ other.name
945
946    def __radd__(self, other):
947        return self.name + other.name
948
949    def __rsub__(self, other):
950        return self.name - other.name
951
952    def __rmul__(self, other):
953        return self.name * other.name
954    def __rdiv__(self, other):
955        return self.name / other.name
956    def __rfloordiv__(self, other):
957        return self.name // other.name
958    def __rmod__(self, other):
959        return self.name % other.name
960    def __rpow__(self, other):
961        return self.name ** other.name
962    def __lshift__(self, other):
963        return self.name << other.name
964    def __rshift__(self, other):
965        return self.name >> other.name
966    def __bitand__(self, other):
967        return self.name & other.name
968    def __bitor__(self, other):
969        return self.name | other.name
970    def __bitxor__(self, other):
971        return self.name ^ other.name
972    def __bitnot__(self):
973        return ~self.name
974
975    def __call__(self):
976        return self.name
977
978    def __getattr__(self, name):
979        return self.__dict__[name]
980
981    def __setattr__(self, name, value):
982        self.__dict__[name] = value
983
984    def __delattr__(self, name):
985        del self.__dict__[name]
986
987    def __copy__(self):
988        return self.__dict__.copy()
989
990    def __deepcopy__(self, memo):
991        return self.__dict__.copy()
992
993    def __reduce__(self):
994        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
995
996    def __reduce_ex__(self, protocol):
997        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
998
999    def __sizeof__(self):
1000        return len(self.__dict__)
1001
1002    def __dir__(self):
1003        return dir(self.__dict__)
1004
1005    def __doc__(self):
1006        return self.__dict__
1007
1008    def __repr__(self):
1009        return str(self)
1010
1011    def __str__(self):
1012        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
1013
1014    def __format__(self, format):
1015        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
1016
1017    def __len__(self):
1018        return len(self.name)
1019
1020    def __getitem__(self, item):
1021        return self.__dict__[item]
1022
1023    def __setitem__(self, item, value):
1024        self.__dict__[item] = value
1025
1026    def __delitem__(self, item):
1027        del self.__dict__[item]
1028
1029    def __contains__(self, item):
1030        return item in self.__dict__
1031
1032    def __iter__(self):
1033        return iter(self.__dict__.items())
1034
1035    def __reversed__(self):
1036        return reversed(self.__dict__.items())
1037
1038    def __hash__(self):
1039        return hash(self.name)
1040
1041    def __eq__(self, other):
1042        return self.name == other.name
1043
1044    def __lt__(self, other):
1045        return self.name < other.name
1046
1047    def __le__(self, other):
1048        return self.name <= other.name
1049
1050    def __gt__(self, other):
1051        return self.name > other.name
1052
1053    def __ge__(self, other):
1054        return self.name >= other.name
1055
1056    def __ne__(self, other):
1057        return self.name != other.name
1058
1059    def __and__(self, other):
1060        return self.name & other.name
1061
1062    def __or__(self, other):
1063        return self.name | other.name
1064
1065    def __xor__(self, other):
1066        return self.name ^ other.name
1067
1068    def __radd__(self, other):
1069        return self.name + other.name
1070
1071    def __rsub__(self, other):
1072        return self.name - other.name
1073
1074    def __rmul__(self, other):
1075        return self.name * other.name
1076    def __rdiv__(self, other):
1077        return self.name / other.name
1078    def __rfloordiv__(self, other):
1079        return self.name // other.name
1080    def __rmod__(self, other):
1081        return self.name % other.name
1082    def __rpow__(self, other):
1083        return self.name ** other.name
1084    def __lshift__(self, other):
1085        return self.name << other.name
1086    def __rshift__(self, other):
1087        return self.name >> other.name
1088    def __bitand__(self, other):
1089        return self.name & other.name
1090    def __bitor__(self, other):
1091        return self.name | other.name
1092    def __bitxor__(self, other):
1093        return self.name ^ other.name
1094    def __bitnot__(self):
1095        return ~self.name
1096
1097    def __call__(self):
1098        return self.name
1099
1100    def __getattr__(self, name):
1101        return self.__dict__[name]
1102
1103    def __setattr__(self, name, value):
1104        self.__dict__[name] = value
1105
1106    def __delattr__(self, name):
1107        del self.__dict__[name]
1108
1109    def __copy__(self):
1110        return self.__dict__.copy()
1111
1112    def __deepcopy__(self, memo):
1113        return self.__dict__.copy()
1114
1115    def __reduce__(self):
1116        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
1117
1118    def __reduce_ex__(self, protocol):
1119        return (self.__class__, (self.name, self.code, self.grade, self.gpa))
1120
1121    def __sizeof__(self):
1122        return len(self.__dict__)
1123
1124    def __dir__(self):
1125        return dir(self.__dict__)
1126
1127    def __doc__(self):
1128        return self.__dict__
1129
1130    def __repr__(self):
1131        return str(self)
1132
1133    def __str__(self):
1134        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
1135
1136    def __format__(self, format):
1137        return str(self.name) + " " + str(self.code) + " " + str(self.grade) + " " + str(self.gpa)
1138
1139    def __len__(self):
1140        return len(self.name)
1141
1142    def __getitem__(self, item):
1143        return self.__dict__[item]
1144
1145    def __setitem__(self, item, value):
1146        self.__dict__[item] = value
1147
1148    def __delitem__(self, item):
1149        del self.__dict__[item]
1150
1151    def __contains__(self, item):
1152        return item in self.__dict__
1153
1154    def __iter__(self):
1155        return iter(self.__dict__.items())
1156
1157    def __reversed__(self):
```

[illegible]

The image shows a Windows desktop with a VS Code editor open. The editor has a dark theme and displays a Python script named `20231103_1exam_3.py`. The script defines a class `Library` with methods for borrowing and returning books. The terminal window at the bottom shows the execution of the script, including a traceback for an `AttributeError` and a successful execution of the `average()` method.

```
20231103_1exam_3.py
1 class Library:
2
3     def __init__(self, student_id):
4         self.books = ["python", "C", "R", "해킹 해킹", "R", "해킹 해킹", "C"]
5         self.borrowed = {}
6         self.borrowed[student_id] = 0
7
8     def rent_process(self, book_name, student_id):
9         if self.books[book_name] == 0:
10             print("해킹 해킹", "해킹 해킹")
11             self.borrowed[student_id] = 1
12
13         elif self.books[book_name] == 1:
14             print("해킹 해킹", "해킹 해킹")
15
16         elif self.borrowed[student_id] == 1:
17             print("해킹 해킹", "해킹 해킹")
18
19         else:
20             print("해킹 해킹", "해킹 해킹")
21
22     def return_process(self, book_name, student_id):
23         if self.books[book_name] == 1:
24             self.books[book_name] = 0
25             print("해킹 해킹", "해킹 해킹")
26             self.borrowed[student_id] = 0
27
28         elif self.books[book_name] == 1:
29             print("해킹 해킹", "해킹 해킹")
30
31         elif self.borrowed[student_id] == 1:
32             print("해킹 해킹", "해킹 해킹")
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
92
```

