#	Old Code	#	New Code
1	`# Keywords & syntax demo (A)`	1	`# Keywords & syntax demo (B)`
2		2	
3	`import math`	3	`import math`
4	`from math import pi as circle_pi`	4	`from math import pi as circle_pi`
5		5	
6	`class Example:`	6	`class Example:`
7	`    def __init__(self, value: int = 0) -> None:`	7	`    def __init__(self, value: int = 1) -> None: #Changed default from 0 to 1`
8	`        self.value = value`	8	`        self.value = value`
9		9	
10	`    def compute(self)-> float:`	10	`    def compute(self)-> float:`
11	`        if self.value > 0:`	11	`        if self.value >= 0: #Changed > to >=`
12	`            for i in range(1, 10):`	12	`            for i in range(1, 10):`
13	`                while i < 5:`	13	`                while i < 6: #Changed 5 to 6`
14	`                    try:`	14	`                    try:`
15	`                        assert i= 3, "Unlucky number"!`	15	`                        assert i= 4, "Unlucky number" # Changed 3 to 4!`
16	`                        yield i`	16	`                        yield i`
17	`                        break`	17	`                        break`
18	`                    except AssertionError as e:`	18	`                    except AssertionError as e:`
19	`                        print(f"Caught: {e}")`	19	`                        print(f"Error: {e}")#Changed message`
20	`                        continue`	20	`                        continue`
21	`                    finally:`	21	`                    finally:`
22	`                        pass`	22	`                        pass`
23	`        elif self.value == 0:`	23	`        elif self.value == -1: #Changed 0 to -1`
24	`            return None`	24	`            return None`
25	`        else:`	25	`        else:`
26	`            raise ValueError("Negative")!`	26	`            raise ValueError("Too negative") # Changed error message!`
27		27	
28	`def main():`	28	`def main():`
29	`    e = Example(2)`	29	`    if a:`
30	`    result = [x for x in e.compute()if x % 2 == 0]`	30	
31	`    print("Results:", result)`		
32			
33	`    match e.value:`	31	`    match e.value:`
34	`        case 0:`	32	`        case 0:`
35	`            print("Zero")`	33	`            print("Zero")`
36	`    def inner(*args, **kwargs):`	34	`    def inner(*args, **kwargs):`
37	`        global x`	35	`        global x`
38	`        nonlocal result`	36	`        nonlocal result`
39	`        x = lambda y: y ** 2`	37	`        x = lambda y: y + 1 #Changed expression`
40	`        print({k: v for k, v in kwargs.items()})`	38	`        print({k: v.upper()for k, v in kwargs.items()})#Added .upper()`
41	`        return x(args[0])if args else None`	39	`        return x(args[0])if args else None`

```
42
43      print(inner(4, key='val'))
44
45  if __name__ == "__main__":
46      main()
```

```
40
41      print(inner(3, key='val'))#Changed
        arg
42
43  if __name__ == "__main__":
44      main()
```