Review for Quiz 2

Welcome back to CS 2100!

Prof. Rasika Bhalerao

Recommeded Review Topics

- Classes
    - Constructors, methods, attributes
    - `__str__()` and `__eq__()`
- Using Objects
    - State and aliasing
    - `None` and `Optional`
- Stakeholder-value matrices
    - Selecting stakeholders
    - Selecting values
- Lists, sets, and dictionaries
    - List comprehension
    - Iterating through lists, sets, and dictionaries
    - Rules about contents of lists, sets, and dictionaries
    - Sorting and filtering
    - Binary operators (`|`, `-`, etc.)
- Correlation
    - Definition of correlation
    - Pearson's correlation coefficient

How to make a class

- Class header is `class Name:` (capital letter)
- Methods: functions inside a class
    - First parameter is `self`
- Attributes: variables shared among all methods
    - Name starts with `self.`
- Constructor: special method that is called when the object is "instantiated"
    - To initialize the attributes
    - `def __init__(self, <args>):`

`__str__()` and `__eq__()`

- Print something -> it calls `def __str__(self) -> str`
- Check if something is equal -> it calls `def __eq__(self, other: object) -> bool`
  - True if `self == other`
  - Check that `other` is the right type first
- If you don't define these methods, it uses the built-in ones

```python
class TwoNumbers:
    def __init__(self, num1: int, num2: int):
        self.num1 = num1
        self.num2 = num2

    def __eq__(self, other: object) -> bool:
        if not isinstance(other, TwoNumbers):
            return False
        return self.num1 == other.num1 and \
            self.num2 == other.num2

    def __str__(self) -> str:
        return f'({num1}, {num2})'

class TestTwoNumbers(unittest.TestCase):
    def test_one_two(self) -> None:
        expected = "(2, 3)"
        self.assertEqual(
            expected, TwoNumbers(2, 3))
```

Poll: Why is this test failing?

1. We have not written the method necessary for checking whether two `TwoNumbers` are equal
2. We have not written the method necessary for converting a `TwoNumbers` into a `str`
3. The test is implicitly converting the `TwoNumbers` into a `str`
4. The test is comparing a `TwoNumbers` with a `str`

State and aliasing

- Variable that holds an object actually holds a *reference* to the object
- Can have multiple variables hold references to the same object
- Modify it using one variable -> all references to it get the modified version

`None` and `Optional`

- `None` is a value that represents the absence of a value
- `Optional[type]` is the type for a variable that might have the value `None`

Poll: What is printed?

```python
class TwoNumbers:
    def __init__(self, num1: int, num2: int):
        self.num1 = num1
        self.num2 = num2

    def __str__(self) -> str:
        return f'({num1}, {num2})'

var1 = TwoNumbers(1, 2)
var2 = TwoNumbers(1, 2)
var1.num1 = 600
print(var2)
```

1. `(1, 2)`
2. `(600, 2)`

List comprehension

Move the body of a `for loop` to right before the `for` (after the opening bracket `[`)

```
my_nums: List[int] = [6, 7, 8, 9]

increased_nums: List[int] = [i + 1 for i in my_nums] # list comprehension

print(increased_nums)  # [7, 8, 9, 10]
```

If we want the resulting list to **filter elements**, we add the `if` clause after the `for` clause.

```
def positive_copy(nums: List[int]) -> List[int]:
    return [i for i in nums if i >= 0]
```

Poll: Which list comprehension correctly creates a list of only the even numbers from 1 to 10?

1. `[x if x % 2 == 0 for x in range(1, 11)]`
2. `[x for x in range(1, 11) where x % 2 == 0]`
3. `[x for x in range(1, 11) if x % 2 == 0]`
4. `[x % 2 == 0 for x in range(1, 11)]`

Sorting

```python
words: List[str] = 'never gonna give you up'.split()

sorted_alphabetically: List[str] = sorted(words)
print(' '.join(sorted_alphabetically)) # give gonna never up you

sorted_by_length: List[str] = sorted(words, key = lambda word: len(word))
print(' '.join(sorted_by_length))  # up you give gonna never
```

The `sorted()` function has an optional argument `key`, which is a function that is applied to each element when determining the sorted order.

Poll: Given this list of tuples representing students and their scores:

```
[('Mini', 85), ('Binnie', 92), ('Ginnie', 78), ('Spleenie', 92)]
```

Which code will sort the students by score in descending order (highest to lowest)?

1. `sorted(students, key=lambda x: -x[1], reverse=True)`
2. `sorted(students, key=lambda x: x[1], reverse=True)`
3. `sorted(students, key=lambda x: -x[1])`
4. `sorted(students, reverse=True)`

Dictionaries

- Maps `key` --> `value`
- Can have the same `value` twice, but not the same `key` twice (`key`s are a set)
- If we map a `key` to a `value`, and then later on, map the same `key` to a different `value`, it overwrites the old `value` with the new one
- `dictionary['cat']` vs `dictionary.get('cat', 4)`:
    - get method will return `4` if `'cat'` isn't in the dictionary (or `None` if we didn't specify the `4`)
    - brackets will raise `KeyError` if `'cat'` isn't in the dictionary

Poll: What happens?

```python
student_grades = {"Binnie": 85, "Ginnie": 92, "Mini": 78}

result1 = student_grades.get("Spleenie")
result2 = student_grades.get("Spleenie", 0)
result3 = student_grades["Spleenie"]
```

1. All three results will be None
2. `result1` is `None`, `result2` is 0, and the third result line raises a `KeyError`
3. The first result line raises a `KeyError`, `result2` is 0, and `result3` is `None`
4. All three result lines raise a `KeyError` because 'Spleenie' is not in the dictionary

Dictionary syntax: iteration

```
ages: Dict[str, int] = {'cat': 10, 'elephant': 14, 'dog': 3}
```

- iterate over its `key` s

```
for key in ages:
    print(f"{key}'s age is {ages.get(key)}")
```

- iterate over its `key-value` pairs

```
for key, value in ages.items():
    print(f"{key}'s age is {value}")
```

Poll: Which correctly prints each student's name and grade?

```
student_grades = {"Binnie": 85, "Ginnie": 92, "Mini": 78}
```

1.

```
for student in grades:
    print(student, grades[student])
```

2.

```
for student, grade in grades.items():
    print(student, grade)
```

3.

```
for pair in grades:
    print(pair[0], pair[1])
```

4.

```
for pair in grades:
    print(pair[0], pair[1])
```

Set operations

- Union (`a | b`): a set that has all elements that are in either set `a` or set `b`
- Intersection (`a & b`): a set that has all elements that are in both set `a` and set `b`
- Subset (`a <= b`): `True` if all elements in `a` are also in `b`, and `False` otherwise
    - Strict subset (`a < b`): `True` if `a <= b` **and `a` is not equal to `b`**, and `False` otherwise
- Subtraction (`a - b`): a set that has all elements in `a` that are not in `b`

Poll: What is in `result1` and `result2`?

```
students_in_math = {'Mini', 'Binnie', 'Ginnie'}
students_in_cs = {'Mini', 'Mega', 'Micro'}

result1 = students_in_math & students_in_cs
result2 = students_in_math | students_in_cs
```

1.

```
result1 = {'Mini'}
result2 = {'Mini', 'Binnie',
    'Ginnie', 'Mega', 'Micro'}
```

2.

```
result1 = {'Mini', 'Binnie',
    'Ginnie', 'Mega', 'Micro'}
result2 = {'Mini'}
```

3.

```
result1 = {'Mini'}
result2 = {'Mini'}
```

4.

```
result1 = {'Mini', 'Binnie',
    'Ginnie', 'Mega', 'Micro'}
result2 = {'Mini', 'Binnie',
    'Ginnie', 'Mega', 'Micro'}
```

Correlation

Correlation measures the strength of a linear relationship between two variables.

Correlation does not imply that an increase in one variable *causes* the other to increase (or decrease). We cannot know whether one of the variables is the cause.

Pearson's Correlation Coefficient

Value	Meaning	Relationship between variables
1	Strong positive correlation	As one increases, so does the other
-1	Strong negative correlation	As one increases, the other decreases
0	No correlation	Change in one variable says nothing about the other

Stakeholder-Value Matrices

1. **Stakeholders**: people who are affected by the software in any way
2. **Values**: values at stake for those stakeholders when considering the software
3. **Cells in the stakeholder-value matrix**: (columns are values, rows are stakeholders), cell contains how each stakeholder's value relates to the software
4. **Analyze conflicts**: e.g., one cell says to increase `x` and another says to decrease `x`

Let's go through Practice Quiz 2!

Poll:

1. What is your main takeaway from today?
2. What would you like to revisit next time?