Kolokwium 1

Informacje wstępne

1. Klonujemy repozytorium o nazwie PO-nr_indeksu-zaliczenia do wybranego katalogu.
2. W PyCharm w sklonowanym katalogu z repozytorium tworzymy projekt o nazwie K1-nr_indeksu, np. K1-12345.
3. W projekcie tworzymy 3 pliki: pizza.py, slice.py oraz main.py
4. W trakcie kolokwium można korzystać z narzędzi wbudowanych w PyCharm lub zainstalowanych pakietów takich jak: flake8 lub mypy w celu sprawdzenia typowania i PEP8.
5. W przypadku braku internetu student zapisuje projekt jako archiwum zip. Kolokwia będą zebrane np. za pomocą pendrive.
6. Rozwiązania po terminie: z uwagi na rozbieżność czasu na stanowiskach w pracowni bieżący czas będzie wyświetlony za pomocą rzutnika. Studenci zostaną uprzedzeni ustnie o zbliżającym się końcu czasu w okresie między 10 a 5 minut przed końcem kolokwium. Brak przesłania repozytorium w wyznaczonym czasie powoduje dodanie dodatkowych 5 minut, ale wtedy maksymalna ocena to 3,5. Powyżej 5 minut spóźnienia - kolokwium pozostaje bez sprawdzenia z oceną niedostateczną. Czas liczony jest od momentu udostępnienia poleceń do poprawnego wypchnięcia na zdalne repo na Githubie lub utworzenia pliku zip w wypadku braku internetu.

Zadanie 1. Klasa Pizza (pol. Pizza) (10pt max.)

1. Napisz klasę Pizza. Klasa powinna posiadać prywatne atrybuty instancyjne:
    1. price typu float (pol. cena)
    2. toppings typu dict[str, int] (pol. dodatki)
    3. diameter typu float (pol. średnica pizzy)
2. Zaimplementuj:
    1. **(3pt)** Inicjalizator z dwoma argumentami diameter oraz toppings. Cena zależy od powierzchni pizzy. Napisz statyczną metodę pomocniczą area liczącą powierzchnię pizzy. Powierzchnia pizzy powinna być obliczona wg wzoru: $pi*(diameter/2)^2$. Cena powinna zostać ustawiona wg wzoru: 0.05 * area + ilosc_dodatkow * 2. Średnica nie powinna być mniejsza niż 20. Jeśli będzie mniejsza niż 20 program powinien kończyć działanie z informacją „błędny promień” i kodem błędu -10.

       Przykład słownika przetrzymującego dodatki {‘ser’: 2, szynka: ‘1’}. Wartości nie powinny być ujemne, a maksymalna wartość może wynosić 3. Jeśli słownik nie spełnia tych warunków powinien być wrócony błąd o kodzie -20.

2. **(2pt)** Zaimplementuj właściwość (propercję) setter dla diameter. Średnica nie powinna być mniejsza niż 20. Jeśli będzie mniejsza niż 20 program powinien kończyć działanie z informacją „błędna średnica” i kodem błędu -10. Jako, że od średnicy pizzy zależy jej cena, uaktualnij również wartość atrybutu cena.

3. **(1pt)** Zaimplementuj metodę:
    1. add_topping przyjmującą jako argument topping (typu str). Metoda ta powinna dodawać do słownika podany dodatek, a gdy dodatek już jest w słowniku zwiększać jego ilość oraz modyfikować cenę pizzy dodając do niej 2 za dodatek. Po każdym dodaniu dodatku, cena powinna być uaktualniana.
4. Nadpisz:
    1. **(2pt)** Odpowiednią metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy:

       „Pizza:

       średnica: 30

       dodatki: ser x 2, szynka, pieczarki

       cena: 60”

       lub w razie braku dodatków:

       „Pizza:

       średnica: 30

       cena: 30”

    2. **(2pt)** Metodę magiczną __add__, zwracającą nowy obiekt stworzony przez: wybranie z większych średnic, połączeniu słowników składników.

Zadanie 2. Klasa Slice (pol. Kawałek) (8pt max.)

1. Zaimplementuj klasę Slice dziedziczącą po klasie Pizza. Klasa ta powinna posiadać atrybut instancyjny prywatny:
    1. how_many_slices (pol. Ile kawałków), typu int
2. **(3pt)** Nadpisz inicjalizator nazywając dodatkowy argument how_many_slices. Ilość kawałków, na ile może być podzielona pizza nie może być mniejsza od 4 i wieksza od 12 i musi być parzysta. Jeśli how_many_slices będzie nieodpowiednie program powinien kończyć działanie z informacją „błędna ilość kawałków” i kodem błędu -10.
3. **(2pt)** Zaimplementuj właściwości (propercje) setter i getter dla nowego atrybutu. Jeśli how_many_slices będzie nieodpowiednie program powinien kończyć działanie z informacją „błędna ilość kawałków” i kodem błędu -10.

4. **(1pt)** Napisz metodę part_price(ordered_slices) która obliczy cenę ordered_slices zamówionych kawałków pizzy. Tj. jeżeli pizza może być podzielona na 6 kawałków i zamówimy 3 to cena powinna wynosić połowę całej pizzy.
5. **(2pt) Nadpisz** metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy:

   „Pizza:

   średnica: 30

   dodatki: ser x 2, szynka, pieczarki

   cena: 60

   kawałki 6

   cena za kawałek 10”

   lub w razie braku dodatków:

   „Pizza:

   średnica: 30

   cena: 30

   kawalki 6

   cena za kawałek 5”

Polecenia pomocnicze do testowania klas. Plik main.py z funkcją main() (Polecenia mają na celu pomoc w testowaniu napisanych klas)

Część do klasy Pizza:

1. Stwórz obiekt o nazwie p_1. Stwórz obiekt o nazwie p_2.
2. Wypisz na ekran reprezentację obiektów p_1 i p_2.
3. Zmień wartość atrybutu diameter na 35 i dodaj jakiś dodatek i wypisz na ekran reprezentację obiektu p_1 po zmianie stanu.
4. Dodaj dodatek do pizzy.
5. Wypisz cenę pizzy p_1.
6. Dodaj obiekty p_1 i obiekt p_2. Wypisz wynik.

Część do klasy Slice:

1. Stwórz obiekt o nazwie s_1.
2. Wypisz na ekran reprezentację obiektu s_1.
3. Wypisz na ekran cenę za 5 kawałków s_1.