**Проект "игра UNO"**

* Правила игры Uno [Rus](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%BE) [Eng](https://en.wikipedia.org/wiki/Uno_(card_game))
* онлайн версия игры <http://www.play-uno.com/board.php>
* где взять код <https://github.com/bennuttall/uno/blob/master/uno.py> или из локальной сети:

scp [student@s41700.vdi.miptNOSPAM.ru](mailto:student@s41700.vdi.miptNOSPAM.ru):/home/student/shared/oop1/oop1\_card\_uno\_orig.py .

|  |  |
| --- | --- |
| [**Task**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonStatic?sortcol=0&table=1&up=0#sorted_table) | [**Developer**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonStatic?sortcol=1&table=1&up=0#sorted_table) |
| Подсчитать счет в конце игры по всем игрокам и вывести его. |  |
| Вводить не номер выбранной карты, а ее написание в вашей руке. |  |
| Требовать писать UNO если после сыгранной карты на руке остается 1 карта. Если игрок забыл это сделать, начислять пенальти. Автоматические игроки должны делать UNO всегда |  |
| Показывать руку игроку отсортированной по цветам (RGBY), и цифре на карте, потом идут специальные, потом черные. |  |
| ? Когда банкующая колода заканчивается, замешать в нее (перемешать!) колоду отбоя (кроме верхней карты). |  |
| ?? Локализация |  |
| Тестирование: запускать состояние игры, прочитав его из файла (json). При этом ввести моду TEST, которая позволит обходить требование, что в руках строго 7 карт и нет отбоя. |  |
| Тестирование: recorder, который пишет лог игры для последующего проигрывания его плеером |  |
| Тестирование: player, который шаг за шагом проигрывает лог, записанный recorder. |  |
|  |  |

**Поля класса**

**Методы класса**

**Перегрузка операторов**

<https://docs.python.org/3/reference/datamodel.html>

| **Operator** | **Function** | **Method Description** |
| --- | --- | --- |
| + | \_\_add\_\_(self, other) | Addition |
| \* | \_\_mul\_\_(self, other) | Multiplication |
| - | \_\_sub\_\_(self, other) | Subtraction |
| % | \_\_mod\_\_(self, other) | Remainder |
| / | \_\_truediv\_\_(self, other) | Division |
| < | \_\_lt\_\_(self, other) | Less than |
| <= | \_\_le\_\_(self, other) | Less than or equal to |
| == | \_\_eq\_\_(self, other) | Equal to |
| != | \_\_ne\_\_(self, other) | Not equal to |
| > | \_\_gt\_\_(self, other) | Greater than |
| >= | \_\_ge\_\_(self, other) | Greater than or equal to |
| [index] | \_\_getitem\_\_(self, index) | Index operator |
| in | \_\_contains\_\_(self, value) | Check membership |
| len | \_\_len\_\_(self) | The number of elements |
| str | \_\_str\_\_(self) | The string representation |

**Как работает @property и зачем он нужен**

property([fget[, fset[, fdel[, doc]]]]) -> property

* **fget** : Функция, реализующая возврат значения свойства.
* **fset** : Функция, реализующая установку значения свойства.
* **fdel** : Функция, реализующая удаление значения свойства.
* **doc** : Строка документации для создаваемого свойства. Если не задано , будет использовано описание от fget (если оно существует).

Позволяет использовать методы в качестве свойств объектов — порождает дескриптор, позволяющий создавать «вычисляемые» свойства (тип property).

Пример использования в классическом виде:

class Mine(object):

def \_\_init\_\_(self):

self.\_x = None

def get\_x(self):

return self.\_x

def set\_x(self, value):

self.\_x = value

def del\_x(self):

self.\_x = 'No more'

x = property(get\_x, set\_x, del\_x, 'Это свойство x.')

type(Mine.x) # property

mine = Mine()

mine.x # None

mine.x = 3

mine.x # 3

del mine.x

mine.x # No more

Используя функцию в качестве декоратора можно легко создавать вычисляемые свойства только для чтения:

class Mine(object):

def \_\_init\_\_(self):

self.\_x = 'some value'

@property

def prop(self):

return self.\_x

mine = Mine()

mine.prop # some value

mine.prop = 'other value' # AttributeError

del mine.prop # AttributeError[?](http://acm.mipt.ru/twiki/bin/edit/Cintro/AttributeError?topicparent=Cintro.PythonStatic)

Объект свойства также предоставляет методы getter, setter, deleter, которые можно использовать в качестве декораторов для указания функций реализующих получение, установку и удаление свойства соответственно. Следующий код эквивалентен коду из первого примера:

class Mine(object):

def \_\_init\_\_(self):

self.\_x = None

x = property()

@x.getter

def x(self):

"""Это свойство x."""

return self.\_x

@x.setter

def x(self, value):

self.\_x = value

@x.deleter

def x(self):

self.\_x = 'No more'

DEA! Как видно из приведённого выше примера, функции-вычислители свойства должны иметь то же имя, что и само свойство.

**Задачи**

**Задача 0 - "Справедливый" купец**

Купец нанял N работников (не менее 1 и не более 100). Все работники делали одинаковую работу. По окончании работ купец расплатился с работниками монетами. Но он дал каждому монет не поровну, а как-нибудь: кому-то много, а кому-то мог вообще не заплатить.

Работники возмутились и стали требовать справедливости.

Купец обещал уладить дело.

Он поставил всех работников в круг. Затем у каждой пары работников он брал деньги и складывал. Если число монет двух работников нечетное, то одну монету купец забирал себе, а остальное поровну делил между этими двумя работниками. Купец начинал круг с пары 1-2, затем 2-3, ... и заканчивал круг парой N-1

После каждого такого "уравнивания" все проверяли поровну ли у них монет. Если у всех было поровну, процесс заканчивался.

Рабочие, получив РАВНОЕ количество монет уходили домой, а купец получал еще немножечко денег.

Написать программу, которая вычисляет сколько монет получил купец и сколько каждый работник.

Автор: Овсянникова Т.В.

**Задача 1. Сдача долларами**

За 1 доллар дают 65 рублей 33 коп. В одном долларе 100 центов. Автомат по продаже мороженного принимает деньги в рублях и долларах. Сдачу отдает только в долларах и центах. Мороженное стоит D долларов, p центов. Человек положил в автомат R рублей.

Написать программу, чтобы вычислить сколько он получит сдачи в долларах и центах.

**Задача 2. Сдача той же валютой**

За 1 доллар дают 65 рублей 33 коп. В одном долларе 100 центов. Автомат по продаже мороженного принимает деньги в рублях и долларах. Сдачу отдает только в долларах и центах. Мороженное стоит D долларов, p центов. Человек положил в автомат R рублей.

Написать программу, чтобы вычислить сколько он получит сдачи в той же валюте, что платил за мороженое.

**Задача 3. Сдача той же валютой - 2**

Автомат по продаже мороженого стоит на территории посольства России в Мьянме.

За 1 доллар дают 65 рублей 33 коп.

За 1 доллар дают 1364 kyat 50 pya.

В одном долларе 100 центов.

Автомат по продаже мороженного принимает деньги в рублях, kyat и долларах. Сдачу отдает той же валютой. Мороженное стоит D долларов, p центов.

Человек положил в автомат R рублей, kyat или долларов.

Написать программу, чтобы вычислить сколько он получит сдачи в той же валюте, что платил за мороженое.

**Задача 5. Юань**

Добавить в автомат юани. 1 доллар 6.63 юаня.

"Копеек" у юаня нет.

Добавить юани в автомат mile

-- [TatyanaDerbysheva](http://acm.mipt.ru/twiki/bin/view/Main/TatyanaDerbysheva) - 19 Nov 2017