COMP10001 Foundations of Computing
Object-oriented Programming
(Advanced Lecture)

Semester 2, 2018

Chris Leckie & Nic Geard

THE UNIVERSITY OF MELBOURNE

Lecture Agenda

- This lecture:
  - Object-oriented programming

NB: All advanced lectures are unexaminable

Objects and Classes I

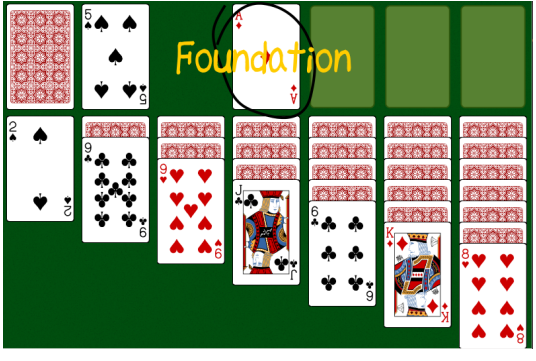
- We are used to everything being an "object" in Python, but what is an "object"? An object is an instance of a "type"/class with pre-defined:
  - associated "fields" (e.g. `complex.imag`)
  - associated "methods" (e.g. `dict.keys()`
- This is based on the notion of "object-oriented programming", which encourages:
  - "encapsulation" of different data types, "inheritance" between related objects, strict "modularity" and (deliberate) "obfuscation" of the internal details of how the data type is implemented

Objects and Classes II

- Encapsulation = expose only the details of each object that are necessary for it to interact with other objects (and hide the "messy" internal details)
- Inheritance = support for one "type" to inherit all the properties of another "type" (and then have additional properties of its own)
- Modularity = break down the components of a problem into conceptual parts
- Obfuscation = hide away the messy internal details

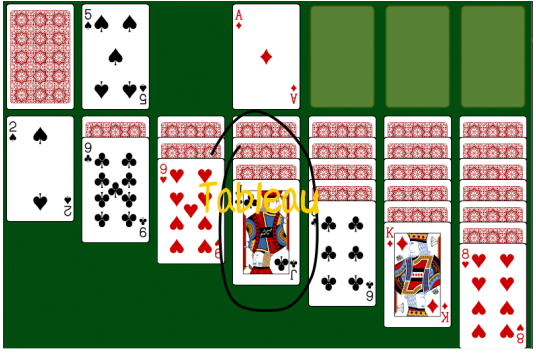
Case Study: Solitaire I

- Components of the game:
  - "Foundation": 4 stacks of cards of the same suit, each in order Ace up to King; initially empty:



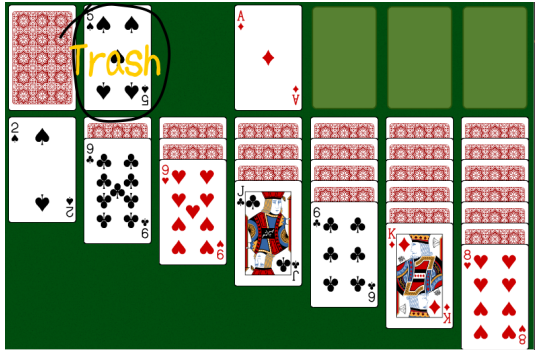
Case Study: Solitaire II

- "Tableau": 7 stacks of cards of varying height, which has a single visible "top" card which can be built on with cards of immediately preceding value of alternating colour



Case Study: Solitaire III

- "Trash": stack of cards that the player iterates through, where the "top" card can be placed on either a Tableau or a Foundation stack



- Each of these stacks is made up of playing cards (0 or more)

Class: Card

- A card is made up of a "value" and a "suit" (which also defines a "colour")
- Cards can be:
  - created
  - tested for value/suit/colour
  - checked for "next" or "previous" values
  - displayed

Python Classes: The Basics I

- A "class" is defined similarly to a function:

```
class Card():
    # FIELDS_AND_METHODS
```

- "Methods" are simply functions associated with instances of a class, and defined within the body of the class:

```
class Card():
    def method(self):
        # FUNCTION BODY
```

The first argument to a method is always `self`, so as to be able to refer to the calling object

Python Classes: The Basics II

- "Fields" are associated with instances of a given class, and created as part of initialising the object using the `__init__` method (which is called when the class is called):

```
class Card():
    def __init__(self):
        self.suit = ...
        self.value = ...

card = Card()
```

Python Classes: The Basics III

- Methods, fields and constants can be made "private" (i.e. inaccessible to other classes) by prefixing them with a double-underscore (__) (with no double-underscore suffix)

Python Classes: Inheritance

- A class can inherit from other classes by including them as arguments in the class statement:

```
class RedCard(Card):
    # FIELDS_AND_METHODS
```

It inherits all methods and fields from the "parent" classes, although it is possible to override them by redefining the method/field using the same name in the child

Class: Stack

- A stack is made up of cards
- Stacks can:
  - have cards added to them
  - have cards removed from them
  - be checked for the "top" card(s)
  - be displayed

Class: Tableau

- A tableau is a type of stack:
  - with a "top" card and a "bottom" card, where the top card can be moved onto other tableaux, and the bottom card can be moved to the foundation stacks or stacked on
  - with unique stacking constraints
  - have the bottom card removed
  - have Kings moved to them if they are empty

Class: Foundation

- A foundation is a type of stack:
  - with particular stacking constraints
  - which has a notion of "completeness" (when the final King is placed on the stack)
  - with unique stacking constraints

Class: Stack

- A stack is made up of cards
- Stacks can:
  - have cards added to them
  - have cards removed from them
  - be checked for the "top" card(s)
  - be displayed

Class: Deck

- A deck is a type of stack:
  - which is initialised to a full pack of cards
  - which can be shuffled
  - which cards can be drawn from (one or *n* at a time)

Class: Game

- A game is made up of the (trash) deck, tableaux and foundation stacks, and:
  - can be displayed
  - has a notion of "solvedness"