# Carta de poker

---

∗ Escribe una clase `Card` que represente una carta de poker ∗

---

Implementa, *al menos*, los siguientes **métodos**:

```
def __init__(self, value: int | str, suit: str):
```

- Crea un atributo `value` que almacene el "valor" (*entero*) de la carta:

| *Carta* | A | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | J | Q | K |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Valor* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

- El argumento `value` puede tomar dos tipos:
  - Si es **entero** puede tomar valores del 1 al 13. Si no es así debes elevar una excepción de tipo `InvalidCardError` con el mensaje:
    `f'Invalid card: {repr(value)} is not a supported value'`
  - Si es **string** puede tomar los siguientes valores: `'A'`, `'1'`, `'2'`, `'3'`, `'4'`, `'5'`, `'6'`, `'7'`, `'8'`, `'9'`, `'10'`, `'J'`, `'Q'` y `'K'`. Si no es así debes elevar una excepción de tipo `InvalidCardError` con el mensaje:
    `f'Invalid card: {repr(value)} is not a supported value'`
- Crea un atributo `suit` que almacene el "palo" (*string*) de la carta:

| *Carta* | Tréboles | Diamantes | Corazones | Picas |
|---|---|---|---|---|
| *Palo* | ♣ | ♦ | ♠ | ♥ |

- El argumento `suit` puede tomar los valores `'♣'`, `'♦'`, `'♠'` y `'♥'`. Si no es así debes elevar una excepción de tipo `InvalidCardError` con el mensaje:
  `f'Invalid card: {repr(suit)} is not a supported suit'`

```
def __repr__(self):
```

- Devuelve el **glifo** de la carta.
- Debes hacer uso del fichero `data/glyphs.dat` para cargar – en algún momento – los glifos de las cartas.

```
def __eq__(self, other: Card | object) -> bool:
```

- Indica si las cartas `self` y `other` son **iguales**.
- Ten en cuenta que `other` puede ser también cualquier otro objeto.

```
def __lt__(self, other: Card) -> bool:
```

- Indica si la carta `self` es **menor** que `other`.

```
def __gt__(self, other: Card) -> bool:
```

- Indica si la carta `self` es **mayor** que `other`.

```
def __add__(self, other: Card) -> Card:
```

- Devuelve una nueva carta como suma de la carta `self` y la carta `other`.
- Para sumar dos cartas ten en cuenta lo siguiente:
  - El **nuevo palo** será el de la carta más alta.
  - El **nuevo valor** será la suma de los valores de ambas cartas. Si el valor pasa de 13, se convertirá en un ***as***.

```
def get_available_suits(cls) -> str:
```

- Es un **método de clase**.
- Devuelve todos los posibles palos como *cadena de texto*.

```
def get_cards_by_suit(cls, suit: str):
```

- Es un **método de clase**.
- Es una **función generadora**.
- Devuelve los **glifos** de las cartas del palo `suit`.

---

* Escribe una clase `InvalidCardError` que represente un error de carta inválida *

---

Implementa, *al menos*, los siguientes **métodos**:

```
def __init__(self, message: str = ''):
```

- El mensaje *por defecto* de esta excepción debe ser:
  `'Invalid card'`
- Si se asigna algún valor al argumento `message`, el mensaje aparecerá como:
  `'Invalid card: El mensaje que sea'`

```
def __str__(self):
```