Seminarium 03

Experimentering

Gustav Sörnäs

21 september 2021

Seminarieformen

Ibland läsa kod och diskutera frågor, ibland skriva kod i mindre grupper. Sedan diskussion i helklass.

Innan seminariet: läs förberedelsematerialet och försök er på uppgifterna.

Skicka in lösningar så vi kan diskutera i helklass: `seminarium.sörnäs.se`. Anonymt, sålänge du inte skriver ditt namn i koden :)

Dagens seminarium

- Läsbar kod
- Uppgift: metoder
- Uppgift: rekursiva listor
- Uppgift: dictionary

Läsbar kod

Korten i en kortlek är numrerade mellan 1 och 52. Skriv en loop som går igenom alla kort i kortleken.

Läsbar kod

```
for i in range(1, 53):
    ...
```

Läsbar kod

```
for card_nr in range(1, 53):
    ...
```

Läsbar kod

```
deck_size = 52
for card_nr in range(1, deck_size + 1):
    ...
```

Läsbar kod

```
deck_size = 52
for card_nr in range(1, deck_size + 1):
    ...

for card_nr in range(1, deck_size + 1):
    ...

for card_nr in range(1, deck_size + 1):
    ...
```

Läsbar kod

```
DECK_SIZE = 52
for card_nr in range(1, DECK_SIZE + 1):
    ...

for card_nr in range(1, DECK_SIZE + 1):
    ...

for card_nr in range(1, DECK_SIZE + 1):
    ...
```

Metoder

Skriv funktioner som manipulerar strängar på olika sätt. Välj själva! Några förslag:

Indata	Utdata
Hello noob	h3ll0 n00b
Super sale	!!!SUPER SALE!!!
snake_to_camel	snakeToCamel
/linux/to/windows	C:\linux\to\windows
Por que no los dos?	¿Por que no los dos?
Jag är vilse	Jojagog äror vovilolsose
Best idea ever	Best. Idea. Ever.

```
>>> leetspeak("Hello noob")
"h3ll0 n00b"
```

Rekursiva listor

Iteration för att gå igenom en lista:

```
def print_values(values):
    for value in values:
        print(value)

>>> numbers = [1, 2, 5]
```

Rekursiva listor

Rekursion för att gå igenom en lista:

```
def print_values(values):
    if not values:
        return
    else:
        print(values[0])
        print_values(values[1:])

>>> numbers = [1, 2, 5]
```

Rekursiva listor

Ibland har vi t.ex. listor i listor, en typ av *rekursiv struktur*.

for-loop för att gå igenom den yttre listan och rekursion för att gå igenom de inre listorna:

```
def print_values(values):
    for value in values:
        if isinstance(value, list):
            print_values(value)
        else:
            print(value)

>>> numbers = [1, [2], [[[5]], 6]]
```

Rekursiva listor

Rekursion för att gå igenom både den yttre listan och de inre listorna:

```
def print_values(values):
    if not values:
        return
    elif isinstance(values[0], list):
        print_values(values[0])
    else:
        print(values[0])
    print_values(values[1:])
```

```
>>> numbers = [1, [2], [[[5]], 6]]
```

Rekursiva listor — uppgift

Skriv en funktion `find_length(length, values)` som letar efter ord av längd `length` i en lista av strängar (`values`) och dess underlistor (eventuellt med ytterligare listor).

Tips:

- Rekursionen kommer alltid bara kolla på första elementet (`values[0]`) och resten av listan (`values[1:]`).
- Fundera på vad som händer om:
  1. Den inskickade listan är tom.
  2. Det första elementet är en lista.
  3. Det första elementet inte är en lista.

     ...

  Kan vissa av de här ske samtidigt? Vad händer då?

Dictionary — uppgift

Skriv en funktion `flatten_dict` som givet ett dictionary av dictionaries returnerar ett nytt ”platt” dictionary.

```
>>> data = {
        "a": {
            "a": {
                "python": 1,
                "java": 2
            },
            "b": {}
        },
        "b": {
            "c": 3
        },
    }
>>> flatten_dict(data)
{'python': 1, 'java': 2, 'c': 3}
```

- `a | b` kombinerar två dictionaries. Testa!