Seminarium 05

Experimentering

Gustav Sörnäs

5 oktober 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 :>

Nästa seminarium

Repetitionstillfälle! Säg till om det är något ni vill repetera via mail till `gusso230@student.liu.se`.

Den här presentationen

```
https://github.com/sornas/tdde23-seminars.git
```

Dagens seminarium

- Funktionell programmering
- Uppgift: högre ordningens funktioner
- Uppgift: lambdauttryck
- Uppgift: iteratorfunktioner

Funktionell programmering

Vad är funktionell programmering?

Funktionell programmering — vanliga funktioner

```
def is_number(x):
    return isinstance(x, int) or isinstance(x, float)

>>> is_number(1)
True
>>> is_number(1.1)
True
>>> is_number("5")
False
>>> is_number([3.14])
False
```

Funktionell programmering — högre ordningens funktioner

```
def is_number(x):
    return isinstance(x, int) or isinstance(x, float)

def test(first, pred):
    return pred(first)

>>> test(1, is_number)
True
>>> test(1.1, is_number)
True
>>> test("5", is_number)
False
>>> test([3.14], is_number)
False
```

Funktionell programmering — högre ordningens funktioner

```
def is_number(x):
    return isinstance(x, int) or isinstance(x, float)

def both(first, second, pred):
    return pred(first) and pred(second)

>>> both(1, 2, is_number)
True
>>> both(1.1, 5, is_number)
True
>>> both("5", 10, is_number)
False
>>> both(1, [3.14], is_number)
False
```

Funktionell programmering — uppgift

Skriv en funktion `count(values, pred)` som går igenom en lista och räknar hur många av elementen som uppfyller ett predikat.

Kom ihåg att ett predikat är en funktion som testar någonting och returnerar `True` eller `False`.

```
>>> count([1, 2, "3", 4, "5"], is_number)
3
>>> count([1, 2, [3]], is_number)
2
```

Funktionell programmering — rena funktioner

```
def f(x):
    return 2 * x

>>> f(1)
2
>>> f(2)
4
f(1)
>>> 2
```

Funktionell programmering — sidoeffekter

```
increment = 0
def g():
    increment += 1
    return increment

>>> g()
1
>>> g()
2
>>> g()
3
```

Funktionell programmering — funktioner som värden

```
def f(x):
    return x

>>> print(f)
<function f at 0x7f429bac7b80>
>>> g = f
>>> g(1)
1
```

Funktionell programmering — funktioner som värden

```
def create_f():
    def f(x):
        return x
    return f

>>> create_f
<function create_f at 0x7f429bac7c10>
>>> g = create_f()
>>> g
<function create_f.<locals>.f at 0x7f429bac7ca0>
>>> g(1)
1
>>> create_f()(2)
2
```

Lambda-uttryck

Vi har sett att funktioner går att använda som värden.

```
def double(x):
    return x * 2


>>> f = double
>>> f(5)
10
>>> f
<function double at 0x7f0ff6cfbc10>
```

Lambda-uttryck

```
def double_f(x):
    return x * 2

double_l = lambda x: x * 2

>>> double_f(5)
10
>>> double_l(5)
10
```

Lambda-uttryck

```
def test(first, pred):
    return pred(first)

>>> test(1, lambda x: isinstance(x, int))
True
>>> test("2", lambda x: isinstance(x, int))
False
```

Lambda-uttryck — uppgift

Med hjälp av `count` från den tidigare uppgiften, skriv predikatfunktioner som lambdan för att räkna följande:

1. Antalet förekomster av strängen ”a”.
2. Antalet listor som är två element långa.
3. Antalet tal som är delbara med tre.

```
>>> count(["a", "B", "c", "a", "d"], ?)
2
>>> count([["a"], [1, 2], ["b", "c"]], ?)
2
>>> count([1, 2, 3, 4, 5, 6, 9], ?)
3
```

```
lambda x: isinstance(x, int) or isinstance(x, float)
```

Iteratorfunktioner

Funktioner som hanterar iteratorer (t.ex. listor) på olika sätt.

Vissa tar in funktioner som argument.

Några exempel som kan vara bra att känna till:

- `list`: gör om en iterator till en lista. Väldigt vanlig.
- `map`: Applicera en funktion på varje element och ge tillbaka en iterator över retur-värdena.

Iteratorfunktioner — andra bra grejer

Kolla upp vad dom gör genom att söka i Python-dokumentationen!

- `filter`
- `len`
- `min`
- `max`
- `enumerate`

Iteratorfunktioner — count

Skriv `count` från tidigare, fast den här gången genom att kombinera inbyggda iteratorfunktioner!

```
def count(values, pred):
    return ?
```

Tips: ni kommer använda 3 av funktionerna som precis nämndes. Vilka? Och hur?