

Laboration 7: Hashtabeller

[Starta uppgift](#)

Inlämningsdatum Onsdag av 20:00 Poäng 1 Lämnar in en filuppladdning
Filtyper py

Mål	Läs i kursboken
Kunna använda och implementera en hashtabell. Kunna använda unittest för att testa program. Repetera Föreläsning 6: Hashning	Kapitel 6.5 (https://runestone.academy/runestone/books/published/pythonds/SortSearch/Hashing.htm)

□ Hashning med Pythons inbyggda dictionary

Minns du hur du i labb 2 gjorde en enkel kö med Pythons array? Nu ska du göra en egen hashtabell med Pythons dictionary!

Skriv en klass DictHash som använder Pythons inbyggda [dictionary](#) (<https://docs.python.org/3.5/tutorial/datastructures.html?highlight=dictionary#dictionaries>). Den måste ha metoderna

- **store**(nyckel, data) som lagrar data som value i din dictionary, med nyckel som key.
- **x = search**(nyckel) som slår upp nyckel i din dictionary.

Dessutom får du om du vill lägga till två extra metoder:

- Vill du kunna skriva `d[nyckel]` istället för `d.search(nyckel)`? Definiera då metoden `__getitem__(self, nyckel)` (https://docs.python.org/3/reference/datamodel.html#__getitem__) som anropar din search-metod.
- Vill du kunna skriva `if nyckel in d`? Definiera då metoden `__contains__(self, nyckel)` (https://docs.python.org/3/reference/datamodel.html#__contains__) som returnerar True om nyckel finns i d, False annars.

testkor till klass DictHash med datan från första labben. Attmanus [pokedex](#).

(<https://gist.github.com/armgilles/194bcff35001e7eb53a2a8b441e8b2c6>)

- (<https://gist.github.com/armgilles/194bcff35001e7eb53a2a8b441e8b2c6>) Skapa Pokémon-objekt på samma sätt som i labb 1. Lägg in objekten i hashtabellen, med namnet som nyckel.
- Sök efter pokémon i hashtabellen.
- Testa också att söka efter ett namn som inte finns med.

□ En egen implementation av hashtabellen

Nu ska du även göra hashningen själv, i din nya klass HashTable med samma gränssnitt och funktionalitet som DictHash. Krav:

- Definiera en klass HashNode för hashtabellens noder. Noderna måste innehålla både nyckel och värde.
- HashTabellen ska vara lagom stor.
- Du måste skriva en egen hashfunktion, som ger en *bra* fördelning över hela tabellen.
- Du ska kunna redogöra för hur bra fördelningen är. T.ex. med en teoretisk analys av din algoritm eller genom att mäta hur många krockar det är som mest vid insättning.
- Någon krockhantering måste ingå, t ex krocklistor eller probning.
- Du ska använda KeyError för att tala om att en nyckel inte finns.

```
class Node:
    """Noder till klassen Hashtable"""

    def __init__(self, key = "", data = None):
        """key: nyckeln som används vid hashningen
        data: det objekt som ska hashas in"""
        self.key = key
        self.data = data

class Hashtable:

    def __init__(self, size):
        """size: hashtabellens storlek"""
        #Fyll i kod här!

    def store(self, key, data):
        """key: nyckeln
        data: objektet som ska lagras
        Stoppar in "data" med nyckeln "key" i tabellen."""
        #Fyll i kod här!

    def search(self, key):
        """key: nyckeln

        Hamtar det objekt som finns lagrat med nyckeln "key" och returnerar det.
        Om "key" inte finns ska vi få en Exception, KeyError"""
        #Fyll i kod här!
        else:
            raise KeyError


    def hashfunction(self, key):
        """key: nyckeln
        Beräknar hashfunktionen för key"""
        #Fyll i kod här!
```

Testning del 1

Prova med datafilen: Armands [pokedex](#)

(<https://gist.github.com/armgilles/194bcff35001e7eb53a2a8b441e8b2c6>)

Testning del 2

Programmet [hashtest.py](#)  (https://canvas.kth.se/courses/21062/files/2990772/download?download_frd=1) innehåller data om alla atomer (namn och atomvikt). Läs om [unittest](#) (<https://docs.python.org/3/library/unittest.html>) så att du kan lista ut vad det gör och hur det anropar hashtabellen. Modifiera det sedan för att kontrollera om din hashtabell fungerar.

Testning del 3

Gå på [Kattis](#) (<https://kth.kattis.com/problems/kth.tilda.hashtabell>) (<https://kth.kattis.com/courses/DD1320/tildah20>) och testa att din hashtabell fungerar. Använd detta huvudprogram:

```
from hashtable import Hashtable
from sys import stdin

def main():
    hashtable = Hashtable(150001)

    for line in stdin:
        line = line.strip()
        key, *value = line.split()
        if key == '#':
            break
        elif len(value) != 0:
            hashtable.store(key, value[0])
        else:
            try:
                value = hashtable.search(key)
                print(value)
            except KeyError:
                print('None')

if __name__ == "__main__":
    main()
```

Betyg

Denna labb kan endast ge betyg E. Du måste lämna in den och redovisa den i tid för att få göra labbarna för högre betyg i period 4.

Redovisning

Labben lämnas in **individuellt** med "Lämna in uppgift"-knappen högst upp på sidan, och ska redovisas muntligt av **bägge** gruppmedlemmarna. **Skriv gruppmedlemmarnas namn i kommentarsfältet!**

Vid redovisningen ska du kunna

- motivera ditt val av hashfunktion, krockhantering och tabellstorlek,
- skissa hashtabellen,
- förklara varför hashning ger snabb sökning,
- berätta hur en unittest-fil är upplagd