Stackar

Programmering 2

vt 25

Outline

Stackar

En *stack* är en datastruktur där varje element refererar till ett annat element. De påminner lite granda om listor, med skillnaden att man brukar inte kunna komma åt valfria element, utan enbart det sista elementet i *stacken*.

Exempel

En vanlig *stack* IRL är en korthög. I flera kortspel så delar man ut ett antal kort till spelarna och sen lägger man resterande kort i en hög. Under spelets gång så drar man sedan det översta – och alltid det översta – kortet ur högen. När kortet är draget så försvinner det ur högen och nästa kort ligger överst.

Stackar

Last In First Out

En *stack* fungerar efter samma princip som anställningar på en större arbetsplats med ”Last In First Out”, LIFO. Alltså att det element, eller den anställd, som tillkom senast är det element som kommer plockas bort först när man plockar bort något — alltså att den senast anställda är den som sägs upp först.

Stackar

En *stack* har följande tre metoder:

1. push
2. peek
3. pop

Metoder

- Metoden `push` lägger till ett element sist (överst) i *stacken*. Du kan jämföra det med metoden `append` i listor.

Metoder

- Metoden `push` lägger till ett element sist (överst) i *stacken*. Du kan jämföra det med metoden `append` i listor.
- Metoden `peek` tittar på det sista (översta) elementet i *stacken* och ger värdet.

Metoder

- Metoden `push` lägger till ett element sist (överst) i *stacken*. Du kan jämföra det med metoden `append` i listor.
- Metoden `peek` tittar på det sista (översta) elementet i *stacken* och ger värdet.
- Metoden `pop` plockar bort det sista (översta), värdet i *stacken*. Du kan jämföra det med motsvarande metod i listor.

Outline

Listor som stackar

Listor

I Python kan du simulera en *stack* med en lista. Det är ganska ganska enkelt eftersom listan har alla metoder en *stack* har och lite till.

```
1 stack = []
2 stack.append(4) # Motsvarar push
3 stack[-1] # Motsvarar peek
4 stack.pop() # Motsvarar pop
```

Stackar och listor

Nackdelen med listor

Nackdelen med listor är att listor är en sammanhängande grupp i minnet på datorn. När man lägger till nya element i listan riskerar den att inte längre få plats på sin allokerade del i minnet, och då måste man flytta på HELA listan. Detta gör att operationer på listor kan ta olika lång tid att göra.

Stackar och listor

Fördelen med stackar

Stackar däremot lagrar varje element separat i minnet, det kan ge lite mer overhead i varje element. Men det gör att man inte riskerar att flytta på hela *stacken* när man lägger till element i den. Detta gör att operationer med *stackar* är konsekventa i hur lång tid de tar att genomföra.

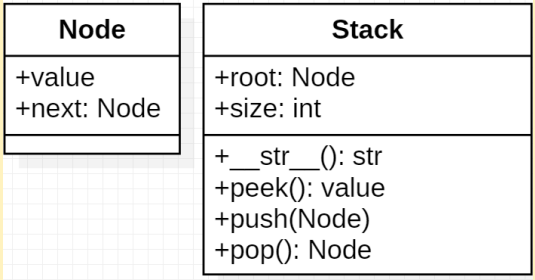
Outline

Övningar

Sida 1, Node

Följ pseudokod på följande slides (totalt 4) och skapa en `Stack` bestående av `Noder`. Skriv din kod i filen `stacks.py` (finns på Vklass) som har en *driver code* som testar dina klasser.

Här ser du klassdiagramen för dem två klasserna



Övningar

Sida 1, Node

```
1 CLASS Node:
2     METHOD __init__ takes in parameter value:
3         SET self.value to value
4         SET self.next to None
```

Övningar

Sida 2, Stack

```
1  CLASS Stack:
2      METHOD __init__:
3          SET self.root to Node('root')
4          SET self.size to 0
5
6      METHOD __str__:
7          SET out_string to ""
8          SET current to self.root.next
9          WHILE current:
10             SET out_string to out_string + str(current.value) +
                   '->'
11             SET current to current.next
12         RETRUN out_string[:-2]
```

Övningar

Sida 3, Stack

```
1     METHOD peek:
2         IF self.size is 0:
3             RAISE Exception
4         RETURN self.root.next.value
5
6     METHOD push takes in parameter value:
7         SET node to Node(value)
8         SET node.next to self.root.next
9         SET self.root.next to node
10        SET self.size to self.size + 1
```

Övningar

Sida 4, Stack

```
1    METHOD pop:
2        IF self.size is 0:
3            RAISE Exception
4        SET remove to self.root.next
5        SET self.root.next to remove.next
6        SET self.size to self.size - 1
7        RETURN remove.value
```