1. Какво е това стек - LIFO?

* Стекът е линейна структура от данни. Стекът служи за съхранение на елементи
* Добавянето на данни в стека се нарича „push“
* Процесът на извличане на данни се нарича „pop“

1. Методи

* Методите на списъка list() улесняват използването на стека
* За добавяне на елемент се използва append()
* За извличане на елемент се използва pop()
* Пример:
* **stack = [3, 4, 5]**
* **stack.append(8)**
* **stack.append(1)**
* **print(stack) *# [3, 4, 5, 8, 1]***
* **stack.pop() *# 1***
* **print(stack) *# [3, 4, 5, 8]***
* **stack.pop() *# 8***
* **stack.pop() *# 5***
* **print(stack) *# [3, 4]***

1. Пример:

My\_stack = []

My\_stack.append(1)

My\_stack.append(2)

My\_stack.append(3)

My\_stack.append(4)

My\_stack.append(5)

My\_stack.append(6)

While my\_stack:

Print(my\_stack.pop())

1. Пример – последна стойност:
2. My\_stack = [1, 2, 3, 4, 5, 6]

* Print(my\_stack[-1])
* A = my\_stack.pop()

Print(A)

my\_stack.append(A)

1. Нека да напишем програма която ревърсва стринга използвайки – стек

* I love Python - > nohtyP evol I
* I study at UKTC - > CTKU ta yduts I

**text = list(input())**

**stack = []**

**for i in range(len(text)):**

**stack.append(text.pop())**

**print("".join(stack))**

* **Или направо с принт**

1. Напишете програма която печата текста във всяка скоба на отделен ред използвайки стек

**Вход:**

**1 + (2 - (2 + 3) \* 4 / (3 + 1)) \* 5**

Изход:

(2 + 3)

(3 + 1)

(2 - (2 + 3) \* 4 / (3 + 1))

**text = input()**

**parentheses = []**

**for i in range(len(text)):**

**if text[i] == "(":**

**parentheses.append(i)**

**elif text[i] == ")":**

**start\_index = parentheses.pop()**

**print(text[start\_index:i + 1])**

1. Опашка – FIFO

* Опашката е абстрактен тип данни „първи влязъл – първи излязъл“.
* Използваме ги, когато искаме елементите да се използват в реда, в който са били подадени
* Възможно е да използвате списък като опашка, но те са много по-бавни
* Правене, вмъкване или взимане от началото на списък е бавно
* from datetime import datetime as dt  
  from collections import deque  
  my\_list = []  
  my\_deque = deque()  
    
  for i in range(1\_000\_000):  
   my\_list.append(i)  
   my\_deque.append(i)  
    
  start = dt.now()  
  for seconds in range(10\_000):  
   my\_list.pop(0)  
  end = dt.now()  
  print(end - start)  
    
  start = dt.now()  
  for seconds in range(10\_000):  
   my\_deque.popleft()  
  end = dt.now()  
  print(end - start)
* Ето защо използваме библиотеката collections – from collections import deque
* Използваме append() за добавяне към опашката и popleft() за премахване от опашката

**stack = [3, 4, 5]**

**stack.append(8)**

**stack.append(1)**

**print(stack) *# [3, 4, 5, 8, 1]***

**stack.popleft() *# 1***

**print(stack) *# [3, 4, 5, 8]***

**stack.popleft() *# 8***

**stack.popleft() *# 5***

**print(stack) *# [3, 4]***

1. \* Гошо е мениджър в Кауфланд, и се нуждае от вашата помощ за да види добре ли се справия новият касиер Стамат на касата. Напишете програма която чете име на клиент и го добавя в края на опашката, докато не получите команда “END”. Но ако получите команда „PAID“, трябва да изпечатате името на всеки клиент на опашката от първия до последния (всеки на нов ред) и да изпразните опашката. Като получите команда „END“ трябва да изпечатате следния текст:

„{останали хора на опашката} people remaining.“

Входни данни 1:

Maria

Neven

Ognqn

PAID

Gosho

Kaloyan

Silvia

Linda

END

Изходни данни 1:

Maria

Neven

Ognqn

4 people remaining.

Входни данни 2:

Ivan

Pesho

Alexander

END

Изходни данни 2:

1. people remaining.
2. \* Домакин на парти

Напишете програма, която следи хората, които си взимат хапки от плато и тяхното количеството, останало накрая. На първия ред ще получите началното количество хапки (цяло число) в платото. След това на следващите редове ще ви бъдат дадени имената на някои хора, които искат да си вземат хапки(всеки на отделен ред), докато не получите командата "Start", добавяйте тези хора в опашка. Ще получавате и други две команди. Накрая ще получите командата "End":

- "{хапка}" - (цяло число), които текущият човек в опашката иска да получи. Проверете дали има достатъчно хапки в платото за този човек.

o Ако има достатъчно хапки, отпечатайте „{person\_name} take bites“ и го/я премахнете от опашката.

o В противен случай отпечатайте „{person\_name} must wait“ и премахнете лицето от опашката, без да намалявате хапките в платото.

- "refill {beets}" - добавете дадените хапки в платото.

Накрая отпечатайте колко хапки остават във формат: "остават {left\_liters} bites".

Входни данни 1:

2

Peter

Any

Start

2

refill 1

1

End

Изходни данни 1:

Peter take bites

Ay take bites

0 bites left

Входни данни 2:

10

Peter

George

Any

Stamat

Start

2

3

3

3

End

Изходни данни 2:

Peter take bites

George take bites

Any take bites

Stamat must wait

2 bites left

1. Горещ картоф е игра, в която децата образуват кръг и хвърлят горещ картоф. Броенето започва с първото дете. При всяко n-то хвърляне детето, което държи картофа, напуска играта. Когато едно дете напусне играта, то предава картофа на следващото дете(от където започва новото броене). Продължава, докато остане само едно дете.

Вашата задача е да напишете програма, която симулира играта Горещ картоф. На първия ред ще получите имената на децата, разделени с един интервал. На втория ред ще получите n-то хвърляне (цяло число), детето което „изгаря“.Отпечатайте всяко дете, което е премахнато от кръга, във формат „Removed {дете}“. Накрая отпечатайте единственото останало дете във формат „Winner is {дете}“.

Входни данни 1:

Tracy Emilia Daniel

2

Изходни данни 1:

Removed Emilia

Removed Tracy

Last is Daniel

Входни данни 2:

George Peter Michail Stamat Toni

10

Изходни данни 2:

Removed Toni

Removed Peter

Removed Michail

Removed George

Last is Stamat

Входни данни 3:

George Peter Michail Stamat Toni

1

Изходни данни 3:

Removed George

Removed Peter

Removed Michail

Removed Stamat

Last is Toni