



# Samsung Innovation Campus

| Coding, Programming & Data Science

Together for Tomorrow!  
**Enabling People**

Education for Future Generations

Chapter 4.

# Algorithm 1

## - Data Structures

Coding, Programming & Data Science

## ຈຸດປະສົງຂອງບົດ

- ✓ ນັກຮຽນຈະສາມາດເຂົ້າໃຈແນວຄວາມຄິດຂອງໂຄງສ້າງຂໍ້ມູນ, ກໍານີດແລະນຳໃຊ້ປະເພດຂໍ້ມູນນາມມະທຳ ລວມທັງ stack, queue, ແລະ hash table. ນອກຈາກນີ້, ນັກຮຽນຈະສາມາດເຂົ້າໃຈການຄົ້ນຫາແບບ linear search ແລະ binary search ເພື່ອແກ້ໄຂບັນຫາການຄົ້ນຫາ, ໃນຂະນະທີ່ສາມາດປຽບທຽບວິທີການຄົ້ນຫາຂໍ້ມູນໂດຍໃຊ້ hash table ແລະ ສະມັດຕະພາບຂອງ algorithm.

## ເນື້ອໃນບົດ

- ✓ ຫົວຂໍ້ທີ 22. Stack
- ✓ ຫົວຂໍ້ທີ 23. Queue
- ✓ ຫົວຂໍ້ທີ 24. Sequential Search
- ✓ ຫົວຂໍ້ທີ 25. Binary Search
- ✓ ຫົວຂໍ້ທີ 26. Hash Table

Unit 22.

# Stack

## Learning objectives

- ✓ ເຂົ້າໃຈແນວຄວາມຄິດຊອງ stack ແລະ ສາມາດກຳນົດ stack ເປັນປະເພດຂໍ້ມູນແບບນາມມະຫຸ້າ (abstract data type).
- ✓ ສາມາດສ້າງໂຄງສ້າງຂໍ້ມູນ stack ໃນຫ້ອງຮຽນໂດຍໃຊ້ພາສາ Python.
- ✓ ສາມາດປະຢູກໃຊ້ໂຄງສ້າງຂໍ້ມູນ stack ເພື່ອແກ້ໄຂບັນຫາການເປີດ-ປິດວົງເລັບໃຫ້ຄົບຖ້ວນ.

## Learning overview

- ✓ រវាននូវ methods ដើម្បីស្វែងស្រាវជ្រាវ stack ໂດຍໃຊ້ Python lists.
- ✓ រវាននូវ methods ដើម្បីនិយាយស្វែងស្រាវជ្រាវ stack បែនខ្លួនខែង Python's class.
- ✓ រវាននូវ methods ដើម្បីແກ້ໄຂបញ្ហាភាមបើក-បើក-លើក-លើក-តាមដឹកជញ្ជូនដោយໃຊ້ស្វែងស្រាវជ្រាវ stack.

## Concepts you will need to know from previous units

- ✓ បច្ចាស់លាយ list, ដំឡើង និង លើបិទបង្ហាញ.
- ✓ និយាយ class constructor និង និយាយពិនិត្យនូវ methods.
- ✓ ស្វែងស្រាវជ្រាវ class instance ដើម្បីបង្កើតពិនិត្យនូវ methods និង ផ្តល់ឱ្យការប្រើប្រាស់ methods.

## Keywords

Abstract  
Data Type

Stack

LIFO

push

pop

# Mission

# 1. Real world problem

## 1.1. ວິ່ງເລັບໃນ source codes

```
#include <stdio.h>

char *s[] = {"Hello", "World"};
int main() {
    printf("%s, %s!\n", s[0], s[1]);
}
```

- ▶ ມີຫຼາຍປະເພດຂອງວິ່ງເລັບຢູ່ໃນ ຊອດໂຄດໂປຣແກຣມ ຄອມພົວເຕີ ຂຶ້ງພວກມັນຈໍາເປັນຕ້ອງໄດ້ໃຊ້ຮ່ວມກັນເປັນຄຸ້ ຄື ເປີດ ແລະ ປິດ.
- ▶ ຮູບຢູ່ເບື້ອງຊ້າຍສະແດງໃຫ້ເຫັນຊອດໂຄດໂປຣແກຣມທີ່ຂຽນ ດ້ວຍພາສາ C ທີ່ສະແດງຄໍາວ່າ "Hello, Word!". ຮູບຂ້າງລຸ່ມ ນີ້ແມ່ນລຳດັບຂອງວິ່ງເລັບທີ່ລວມຢູ່ໃນຊອດໂຄດແຫ່ງນີ້ (ວິ່ງຂໍ, ວິ່ງປຶກາ ແລະ ວິ່ງເລັບ).

[]{}(){}([ ])] ) }

- ▶ ຖ້າວິ່ງເລັບຂ້າງເທິງບໍ່ຖືກຕ້ອງເນື່ອງຈາກການຈັບຄຸ້ທີ່ບໍ່ເຫັນມາສືມ, ຕົວແປພາສາ C ຈະພິມຂໍ້ຄວາມສະແດງການຜິດພາດ. ວິ່ງເລັບໃນພາສາ Python ກໍ່ຄວນຖືກຈັບຄຸ້ຢ່າງເໜາະສືມ.
- ▶ ພວກເຮົາຄວນກວດເບິ່ງແນວໄດ້ວ່າວິ່ງເລັບທີ່ກັບຄຸ້ຢ່າງເໜາະສືມໃນຊອດໂຄດທີ່ໃຫ້ມາ?

## 2.Mission

### 2.1. ຕົວກວດສອບການຈັບຄຸ້ວົງເລັບ

| ສ້າງໂປຣແກຣມເພື່ອກວດເບິ່ງວ່າການຈັບຄຸ້ວົງເລັບນີ້ນຖືກຕ້ອງ ຫຼື ບໍ່ຖືກຕ້ອງ ເນື່ອຂໍ້ຄວາມປະກອບດ້ວຍ ວົງຂໍ, ວົງປຶກກາ ແລະ ວົງເລັບທີ່ໃຫ້ມາດັ່ງລຸ່ມນີ້.

[ ] { } ( )

( ( (

( ( ( ) ) )

( ( ( ] ] ]

( { [ ] } )

) ( ] [ } {

ຖືກຕ້ອງ

ບໍ່ຖືກຕ້ອງ

### 3. Solution

#### 3.1. ການເຮັດວຽກຂອງຕົວກວດສອບ

| ວິ່ງເລັບທີ່ຖືກປ້ອນເຂົ້າໄປຖືກສິ່ງໄປໃຫ້ຕົວກວດສອບເພື່ອກວດສອບວ່າ ຄຸ້ໃດຖືກຕ້ອງ ຫຼື ບໍ່ຖືກຕ້ອງ

```
1 str = input("Input a string of parentheses: ")
2 print("Valid" if check(str) else "Invalid")
```

Input a string of parentheses: [{}()]
Valid

```
1 str = input("Input a string of parentheses: ")
2 print("Valid" if check(str) else "Invalid")
```

Input a string of parentheses: [{()])
Invalid

### 3. Solution

#### 3.2. ໄຄດສູດທ້າຍຂອງໂປຣແກຣມກວດສອບ

```
1 def check(expr):
2     opening = "[{("
3     closing = "])}"
4     stack = []
5     for char in expr:
6         if char in opening:
7             stack.append(char)
8         elif char in closing:
9             if len(stack) == 0:
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return len(stack) == 0
```

# | Key concept

# 1. Algorithms และ Data Structures

## 1.1. ປະເພດຂໍ້ມູນທີ່ໄດ້ກຳນົດໄວ້ແລ້ວ ແລະ ປະເພດຂໍ້ມູນທີ່ຜູ້ໃຊ້ກຳນົດເອງ

- | ພາສາການຂຽນໂປຣແກຣມທັງໝົດໃຫ້ປະເພດຂໍ້ມູນທີ່ກຳນົດໄວ້. ໃນ Python, ມີຫຼາຍຊະນິດຂອງຂໍ້ມູນທີ່ສ້າງຂຶ້ນໃນຕົວ ລວມທັງປະເພດ integer, real, ປະເພດ logic, string, list, dictionary, ແລະ ອື່ນໆ.
- | ນອກຈາກນີ້, ທຸກໆພາສາການຂຽນໂປຣແກຣມໄດ້ມີວິທີການກຳນົດປະເພດຂໍ້ມູນໃຫມ່.
- | ໃນ Python, ຜູ້ໃຊ້ສາມາດກຳນົດປະເພດຂໍ້ມູນໃຫມ່ເປັນ class.



Abstract data type ຫມາຍເຖິງປະເພດຂໍ້ມູນທີ່ກຳນົດໂດຍຜູ້ໃຊ້.

# 1. Algorithms และ Data Structures

## 1.2. Algorithms และ data structures

- | Algorithm หมายถึงขั้นตอนวิธีที่เพื่อแก้ไขปัญหาที่ได้กำหนดไว้ ย่างมีประสิทธิภาพ.  
Data structures แม่นโครงสร้างข้อมูล ซึ่งเป็นแนวความคิดที่บีบเน้นมาตามที่ทักษะของ  
และจัดระบบข้อมูลสำหรับงานเริ่มหาข้อมูลเข้าไปเก็บและงานเริ่มหาข้อมูลออกมานั้น  
ให้ได้ประสิทธิภาพ.
- | ดังนั้น, ต้องใช้ Algorithm ที่มีประสิทธิภาพเพื่อกำนิดโครงสร้างข้อมูล และ ความเลือกโครง  
สร้างข้อมูลที่เหมาะสมสมเพื่อออกรูปแบบ algorithm ที่มีประสิทธิภาพ.

### FOCUS

โปรแกรมคอมพิวเตอร์ เมื่อนำสู่�行 จะใช้ data structure และ algorithm.

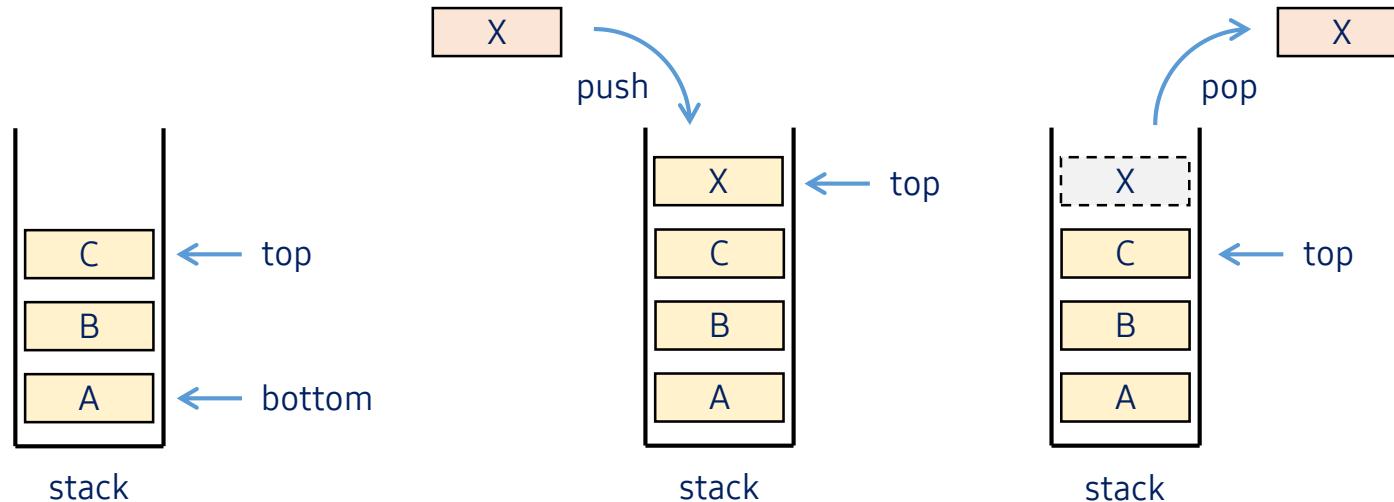
## 2. Stack ແມ່ນຫຍັງ?

### 2.1. Definition of a stack

| Stack ເປັນປະເພດຂໍ້ມູນແບບນາມມະທຳທີ່ເກັບຂໍ້ມູນລຽງຕິດຕໍ່ກັນໄປເປັນລໍາດັບ ທີ່ມີທາງເຂົ້າ ແລະ ທາງອອກຢູ່ເບື້ອງດຽວກັນ. ຂໍ້ມູນທີ່ເອົາເຂົ້າກ່ອນຈະຕ້ອງເອົາອອກນຳຫຼັງ ຂໍ້ມູນທີ່ເອົາເຂົ້ານຳຫຼັງຈະຕ້ອງເອົາອອກກ່ອນ (Last-In-First-Out).

 Focus Stack ປະກອບດ້ວຍສອງ Operation ດັ່ງນີ້.

- ▶ push: ເອົາຂໍ້ມູນເຂົ້າໄປເກັບຢູ່ຫຼັງສຸດ(ຕົວເທິງສຸດ).
- ▶ pop: ເອົາຂໍ້ມູນຕົວສຸດທ້າຍ (ຕົວເທິງສຸດ) ອອກມາ.



## 2. Stack ແມ່ນຫຍັງ?

### 2.2. ຕົວຢ່າງ stack

- ▶ Stack ແມ່ນພິບເຫັນທີ່ໄປໃນການດຳລົງຊີວິດປະຈຳວັນ.
- ▶ ຕົວຢ່າງ, ຖ້ວຍ/ຈານຖືກວາງຊ້ອນກັນໃນຂະນະທີ່ກຳລັງຈະລ້າງຖ້ວຍ/ຈານ ແລະ ຫຼັງຈາກນັ້ນຈະເອົາອັນຢູ່ເທິງສຸດອອກມາເທື່ອລະອັນເພື່ອລ້າງ.
- ▶ ຕົວຢ່າງອື່ນຂອງ stacking ໃນຊີວິດປະຈຳວັນປະກອບມີ ການວາງປຶ້ມ, ການວາງຫຼຽນ ແລະ ກະແຕ່ງໃສ່ໝາກໄມ້/ຜັກ, ຕຸ້ໄສ່ໄກ່ນ້ອຍ ແລະ Pringles.
- ▶ ຕົວຢ່າງອື່ນຂອງ stack ມີຫຍັງແດ່?



## 2. Stack ແມ່ນຫຍັງ?

### 2.2. ຕົວຢ່າງ stack

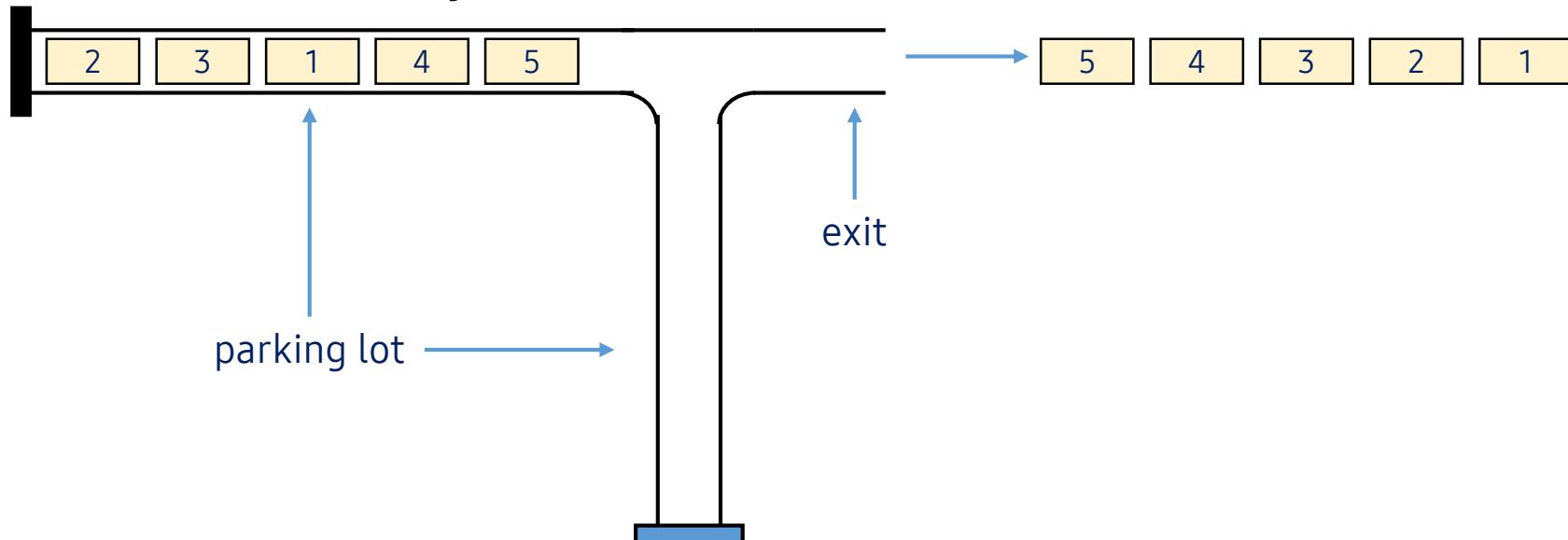


- ▶ Web browser ໄດ້ເກັບບັນທຶກທີ່ຢູ່ຂອງເວັບໄຊທີ່ເຂົ້າໄປໃຊ້ງານເຂົ້າໄປໄວ້ໃນ a stack. ເມື່ອເຂົ້າໄປໃຊ້ໜຳເວັບແພດໃໝ່ ມັນຈະເກັບສະສົມໜຳນັ້ນໄວ້ເປັນໂຕເທິງສຸດຂອງ stack.
- ▶ ຖ້າຜູ້ໃຊ້ຄລິກ ‘back,’ ທີ່ຢູ່ໂຕເທິງສຸດຂອງ stack ຈະຖືກເອີ້ນອອກມາໃຊ້ງານ ຂີ່ມັນຈະສະແດງເວັບໄຊທີ່ໄດ້ເຂົ້າໄປຢໍຽມຊົມກ່ອນໜຳນັ້ນ.
- ▶ ມີ undo function ໃນບັນດາ text editor ແລະ MS Word.
- ▶ Function ນີ້ກໍ່ເກັບບັນທຶກການແກ້ໄຂຕ່າງໆຂອງຜູ້ໃຊ້ເຂົ້າໄປໄວ້ໃນ stack, ແລະ ‘undo’ ຈະຍົກເລີກການແກ້ໄຂເຫຼົ່ານັ້ນຕາມລົວດ້າບຈາກເທິງລົງລຸ່ມຂອງ stack.

## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

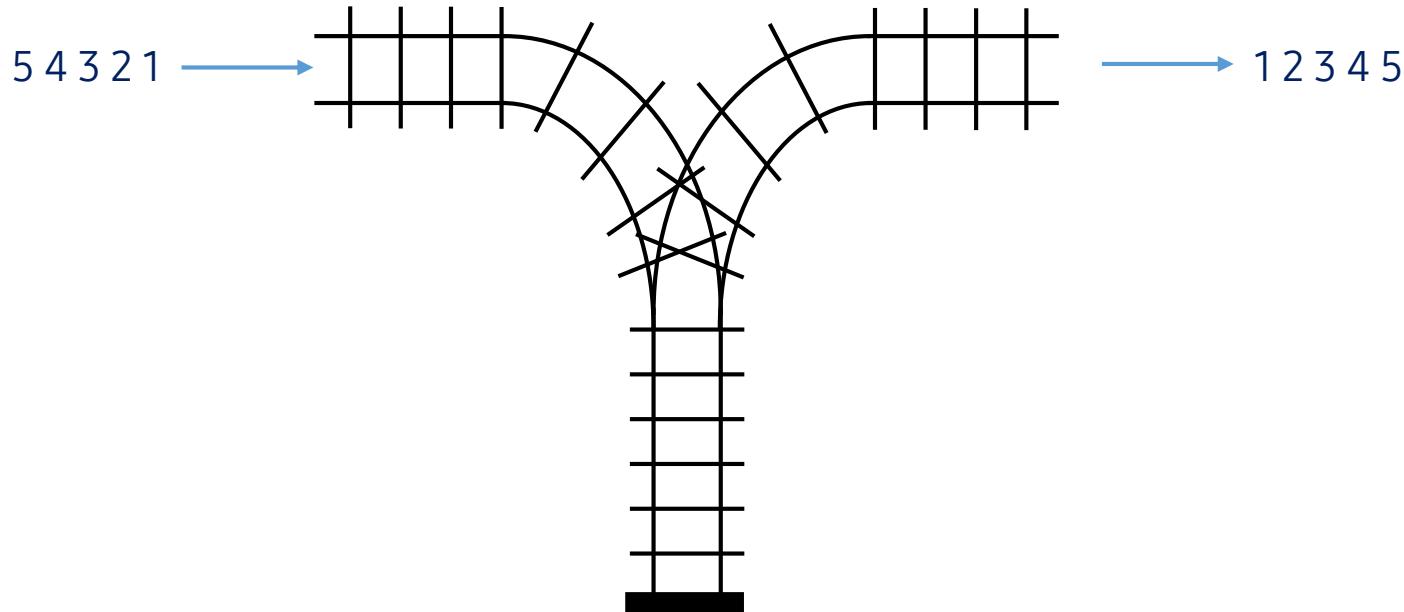
- | ຍົກຕົວຢ່າງ ການຈອດລິດຕາມຮູບຂ້າງລຸ່ມ.
- | ທາງເບື້ອງຊ້າຍ ມີລິດໃຫຍ່ 5 ຄົນຈອດຄາຢູ່ກາງຖະໜົນທີ່ມີສິ່ງກິດຂວາງຢູ່ດ້ານໜ້າ ແລະ ເສັ້ນທາງດ້ານລຸ່ມກໍ່ຕັນເຊັ່ນດຽວກັນ. ລິດ 5 ຄົນດັ່ງກ່າວໄດ້ມາຮອດຕາມລຳດັບ 2, 3, 1, 4, 5 ດ້ວຍຊ່ວງເວລາແບບສຸ່ມ. ລິດທັງໝົດຕ້ອງການອອກຈາກການກິດຂວາງດັ່ງກ່າວໄປທາງເບື້ອງຂວາຕາມລຳດັບ 1, 2, 3, 4, 5.
- | ຖ້າທ່ານເປັນຄົນຈັດການສະພາບການດັ່ງກ່າວ ທ່ານຈະແກ້ບັນຫານີ້ແນວໃດ?



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

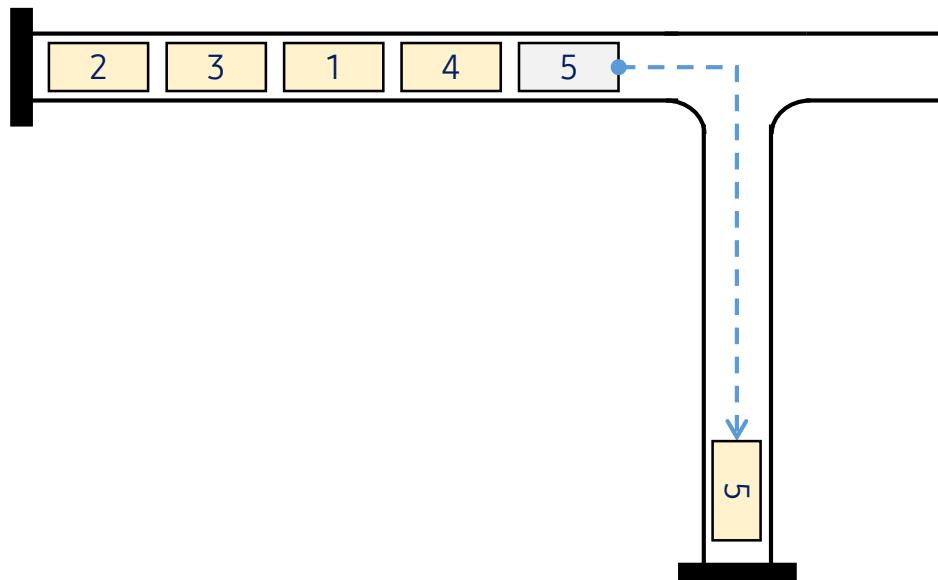
- | ກ່ອນທ່ານແກ້ໄຂບັນຫາບ່ອນຈອດລິດຂ້າງເທິງ ຈຶ່ງພິຈາລະນາສັນຫາລົດໄຟທີ່ມີລັກສະນະຄືຮູບຂ້າງລຸ່ມ.
- | ຖ້າວ່າລົດໄຟທີ່ມີໝາຍເລກຕູ້ຈາກ 1 ເຖິງ 5 ເຂົ້າມາຈາກດ້ານຊ້າຍ ແລະ ອອກໄປທາງດ້ານຂວາ ຈະເຫັນວ່າໝາຍເລກຕູ້ລົດໄຟທີ່ອອກໄປຈະບິນລວງໝາຍເລກຕູ້ທີ່ເຂົ້າມາ.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

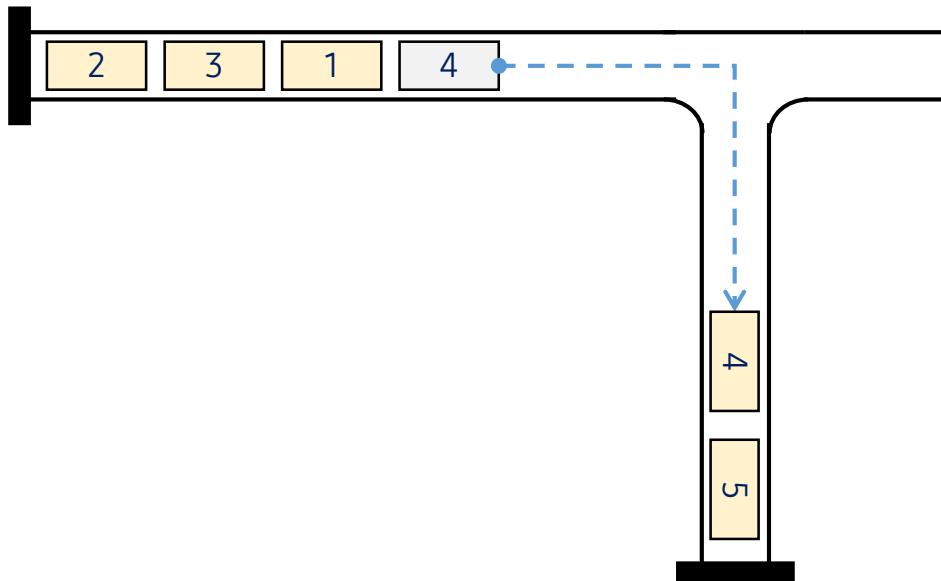
- | ແກ້ໄຂບັນຫາປ່ອນຈອດລົດດັ່ງນີ້.
- | ກ່ອນອື່ນ, ໃຫ້ຈອດລົດໝາຍເລກ 5 ຢູ່ໃນຊອຍທາງລຸ່ມຊື່ວຄາວ.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

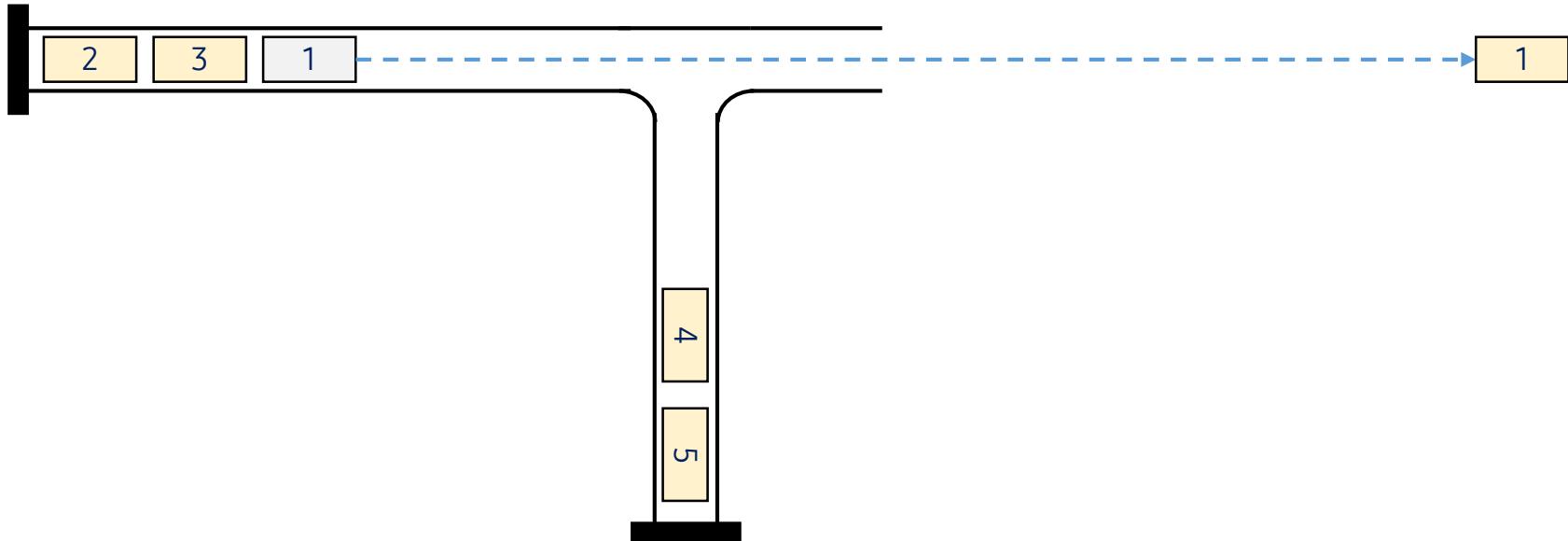
| ຕໍ່ໄປ, ໃຫ້ເອົາລົດໝາຍເລກ 4 ໄປຈອດຢູ່ບ່ອນຈອດລົດຊື່ວຄາງທາງດ້ານລຸ່ມຄືເກົ່າ.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

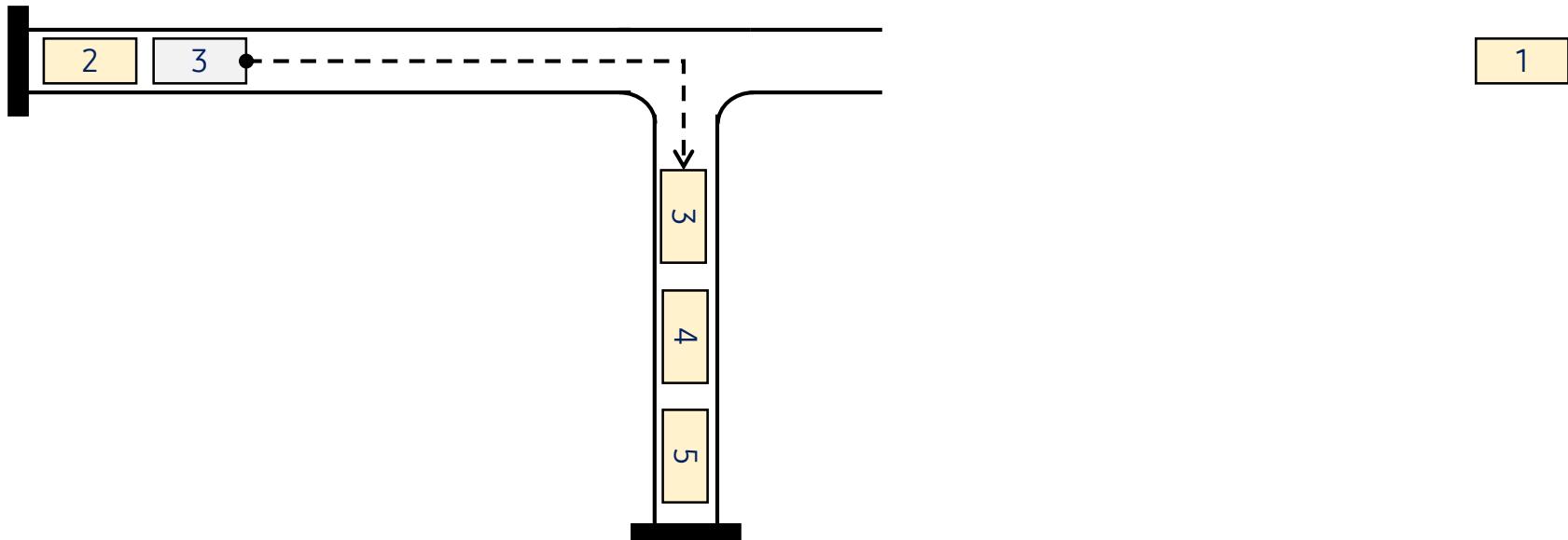
| ຕອນນີ້ ລົດເລກ 1 ສາມາດອອກຈາກບ່ອນຈອດລົດໄດ້ກ່ອນໜຸ່ແລ້ວ.



## 2. Stack ແມ່ນຫຍຸງ?

### 2.3. Applications ຂອງ stack

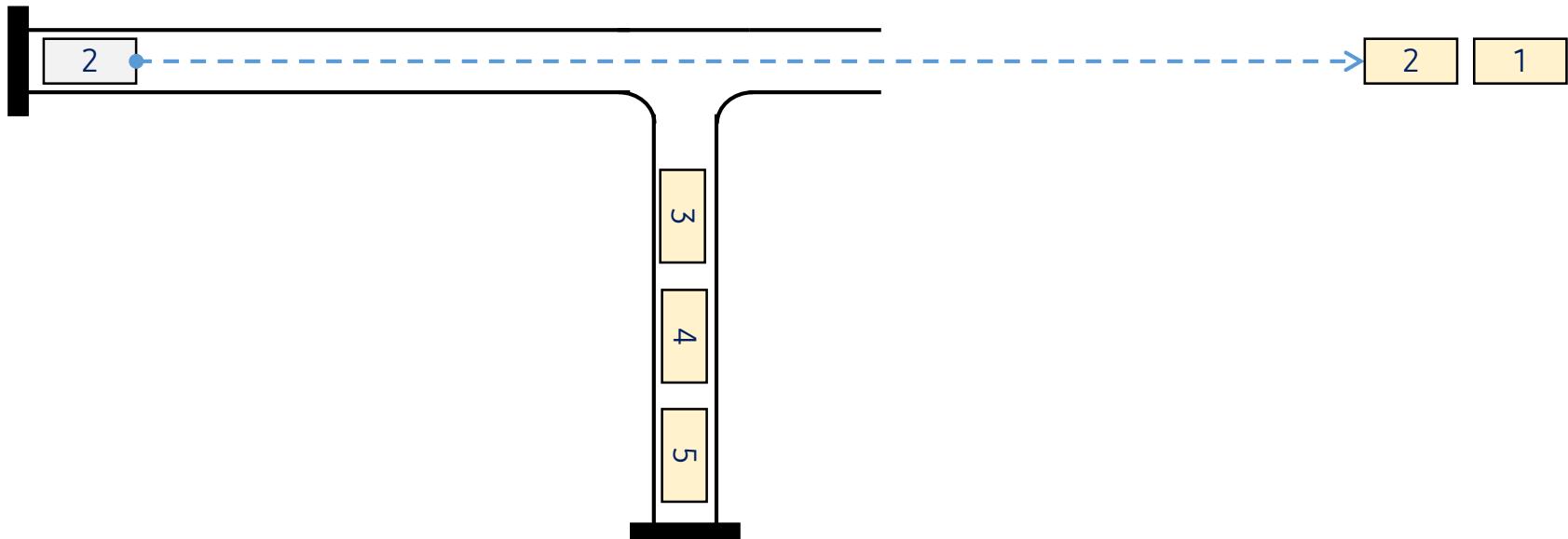
| ອີກເຫຼືອໜຶ່ງ, ໃຫ້ຈອດລົດໝາຍເລກ 3 ຢູ່ທາງເບື້ອງລຸ່ມນີ້ຊື່ວຄາວ.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

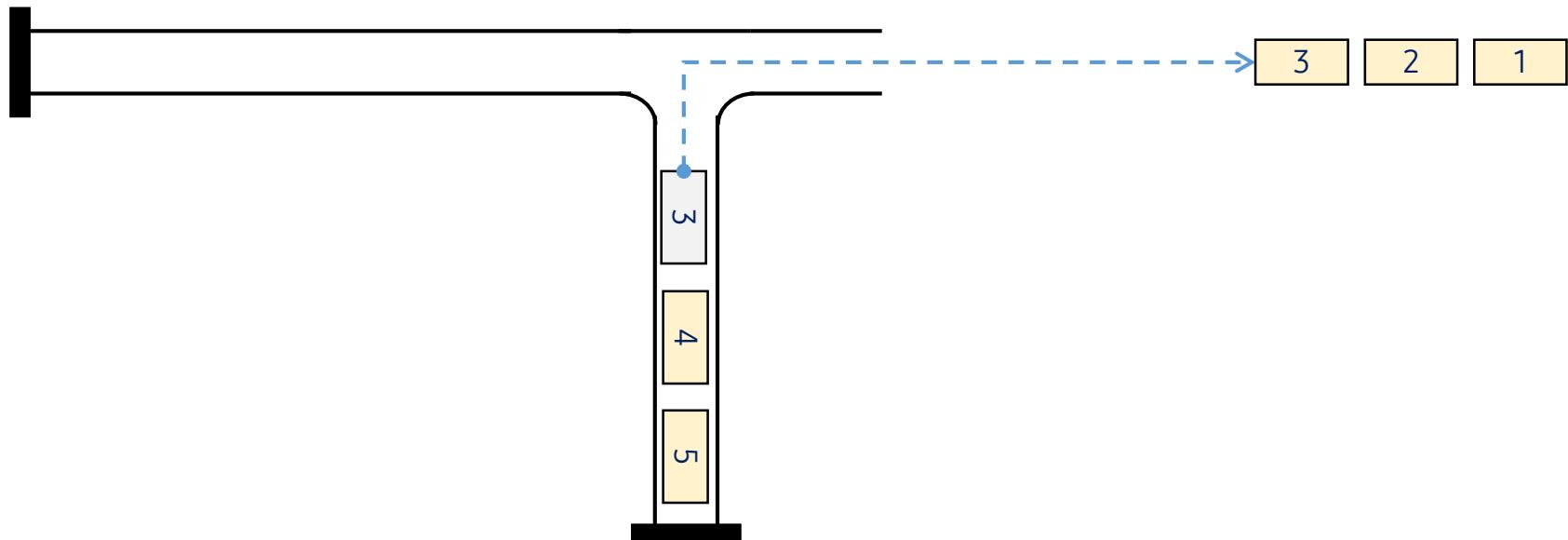
| ຈາກນັ້ນ, ລົດໝາຍເລກ 2 ສາມາດອອກຈາກບ່ອນຈອດລົດຕໍ່ຈາກລົດໝາຍເລກ 1.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

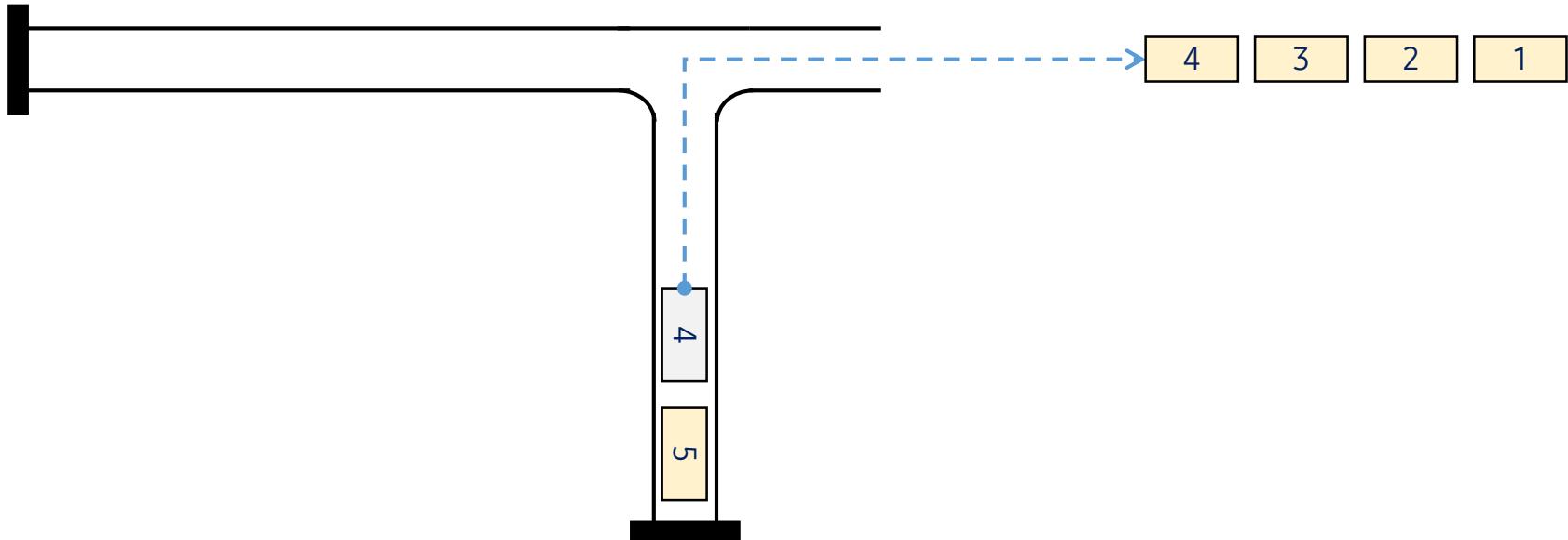
- | ເຫັນໄດ້ວ່າ ຫາງລຸ່ມ ລົດໄດ້ຈອດເປັນລະບຽບຕາມລຳດັບໄວ້ແລ້ວ, ລົດໝາຍເລກ 3 ສາມາດອອກໄດ້ ຕໍ່ຈາກລົດໝາຍເລກ 2.
- | ຕ້ອງເຂົ້າໃຈວ່າ ລົດໝາຍເລກ 3 ແມ່ນລົດຄົນສຸດທ້າຍທີ່ຈອດຢູ່ຫາງດ້ານລຸ່ມ.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

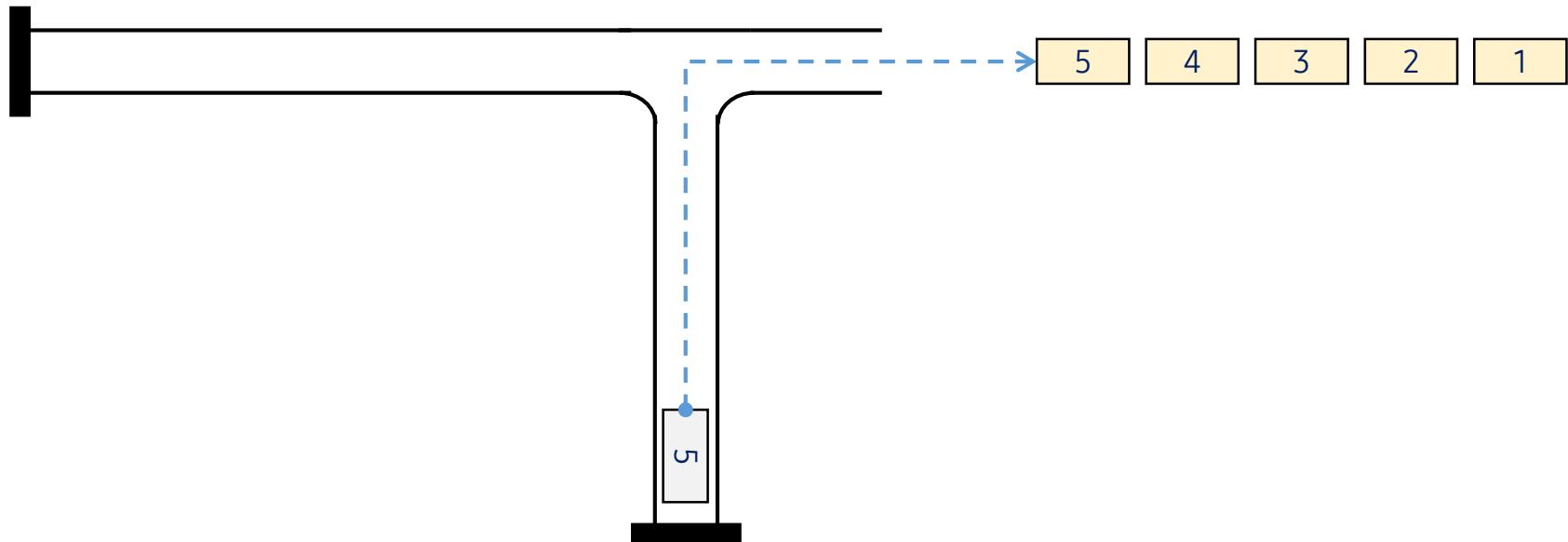
- | ຂະນະນີ້, ລົດຄັນສຸດທ້າຍທີ່ຈອດຢູ່ທາງລຸ່ມແມ່ນລົດໝາຍເລກ 4.
- | ລົດໝາຍເລກ 4 ສາມາດອອກຈາກບ່ອນຈອດລົດ ຕໍ່ຈາກລົດໝາຍເລກ 3.



## 2. Stack ແມ່ນຫຍັງ?

### 2.3. Applications ຂອງ stack

- | ຂະນະນີ້, ທ້າລົດເລກ 5 ອອກຈາກບ່ອນຈອດລົດ, ບັນຫາແມ່ນໄດ້ຮັບການແກ້ໄຂ.
- | ລົດໝາຍເລກ 5 ທີ່ອອກຈາກບ່ອນຈອດລົດສຸດທ້າຍແມ່ນລົດທີ່ຈອດຢູ່ທາງລຸ່ມກ່ອນໜຸ້ມືດ.



### 3. ການສ້າງ Stack

#### 3.1. ສ້າງໂດຍໃຊ້ປະເພດຂໍ້ມູນ list

| ໃນພາສາ Python, stack ສາມາດສ້າງໂດຍໃຊ້ປະເພດຂໍ້ມູນ list.

```
1 stack = []
2
3 def push(stack, item):
4     stack.append(item)
5
6 def pop(stack):
7     return stack.pop()
```

##### Line 1

- ກໍານົດໂຄງສ້າງ stack ໂດຍໃຊ້ຕົວປ່ຽນປະເພດຂໍ້ມູນ list.
- ການເພີ່ມຫຼືການລຶບອີງປະກອບແມ່ນເຮັດໄດ້ໃນທຸກຕຳແໜ່ງໃນປະເພດ phyton list, ດັ່ງນັ້ນ ການປະຕິບັດງານເພີ່ມແມ່ນການເອົາຂໍ້ມູນຂຶ້ນໃນ stack(push) ແລະ ເອົາຂໍ້ມູນອອກຈາກ stack (pop).

### 3. ການສ້າງ Stack

#### 3.1. ສ້າງໂດຍໃຊ້ປະເພດຂໍ້ມູນ list

| ເພື່ອຄວາມສະດວກ, ໃຫ້ສົມມຸດວ່າອີງປະກອບສຸດທ້າຍຂອງ stack ແມ່ນ 'top.' ຈາກນີ້ນ, ພັງຊັນ 'push' ຈະເພີ່ມອີງປະກອບເຂົ້າຂ້າງເທິງສຸດຂອງ stack.

```
1 stack = []
2
3 def push(stack, item):
4     stack.append(item)
5
6 def pop(stack):
7     return stack.pop()
```



#### Line 3-4

- ພັງຊັນ 'push' ຈະເປັນຕົວຮັບການປ້ອນຂໍ້ມູນຂອງ stack ປະເພດ list ແລະ ລາຍການຂໍ້ມູນທີ່ຈະເພີ່ມເຂົ້າ.
- ເນື່ອງຈາກ 'push' ຈະເພີ່ມລາຍການຂໍ້ມູນເຂົ້າດ້ານເທິງຂອງ stack, ໃຊ້ພັງຊັນ append() ຂອງ stack ປະເພດ list.

### 3. ການສ້າງ Stack

#### 3.1. ສ້າງໂດຍໃຊ້ປະເພດຂໍ້ມູນ list

| ພັງຊັນ 'pop' ຈະລຶບອີງປະກອບຕົວສຸດທ້າຍຂອງ stack ແລະ ບອກອີງປະກອບທີ່ຖືກລົບ.

```
1 stack = []
2
3 def push(stack, item):
4     stack.append(item)
5
6 def pop(stack):
7     return stack.pop()
```

#### Line 6-7

- Stack ປະເພດ list ແມ່ນການປ້ອນຂໍ້ມູນໃຫ້ແກ່ພັງຊັນ 'pop'.
- ເນື່ອງຈາກ 'pop' ລຶບ ແລະ ສິ່ງຄືນອີງປະກອບເທິງສຸດຂອງ stack ໂດຍໃຊ້ພັງຊັນ pop() ຂອງ stack ປະເພດ list.

### 3. ການສ້າງ Stack

#### 3.1. ສ້າງໂດຍໃຊ້ປະເພດຂຶ້ມູນ list

| Code ຕໍ່ໄປນີ້ໃຊ້ທິດສອບໂຄງສ້າງຂຶ້ມູນ stack ທີ່ຖືກສ້າງໄວ້ກ່ອນໜັນນີ້.

```
1 stack = []
2 print(stack)
3 push(stack, "A")
4 print(stack)
5 push(stack, "B")
6 print(stack)
7 pop(stack)
8 print(stack)
9 push(stack, "C")
10 print(stack)
11 pop(stack)
12 print(stack)
13 pop(stack)
14 print(stack)
```

```
[]  
['A']  
['A', 'B']  
['A']  
['A', 'C']  
['A']  
[]
```

### 3. ការស៉ាវ Stack

#### 3.1. ស៉ាវໄត្ថេប្រព័ន្ធមូលដ្ឋាន list

| เมื่ៅໃຊ៉ការ 'pop' រាយការណ៍ stack ទៅវាបីំផុត, មិនអាចបញ្ចប់លើកការណ៍ទាំងនេះបាន.

##### IndexError

```
1 stack = []
2 pop(stack)
```

```
-----
IndexError                                     Traceback (most recent call last)
<ipython-input-29-f40cd90df1a6> in <module>
      1 stack = []
----> 2 pop(stack)

<ipython-input-21-3270d1cc050e> in pop(stack)
      5
      6     def pop(stack):
----> 7         return stack.pop()

IndexError: pop from empty list
```

### 3. กานส້າງ Stack

### 3.1. ສ້າງໂດຍໃຊ້ປະເພດຂໍມູນແບບ list

| เพื่อแก้ไขข้อบกพร่อง IndexError ใน stack หัวข้อเบื้องต้น ให้ส้างฟังก์ชัน 'is\_empty()' เพื่อ  
ກວດເບີ່ງວ່າ stack ແມ່ນຫວັງບໍ່.

```
1 def is_empty(stack):  
2     return True if len(stack) == 0 else False  
3  
4 stack = []  
5 print(is_empty(stack))
```

True

Line 1-2

- ฟังชัน `is_empty()` สิ่งกืนค่า `False` ถ้าความຍາວຂອງ `stack` ທີ່ເປັນພາຣາມີເຕີຕົວໜຶ່ງມີຄ່າເປັນ `0` ແລະ ສິ່ງກິນຄ່າ `True` ທັງບໍ່ເປັນດັ່ງນັ້ນ.
  - ເມື່ອເລີ່ມຕົ້ນ `stack`, ความຍາວ `stack` ແມ່ນ `0`. ດັ່ງນັ້ນ, `is_empty()` ສິ່ງກິນຄ່າ `True`.

### 3. ການສ້າງ Stack

#### 3.1. ສ້າງໂດຍໃຊ້ປະເພດຂຶ້ມູນແບບ list

| ເພື່ອແກ້ໄຂບັນຫາ IndexError ໃນ stack ຫວ່າງເປົ່າ, ທໍາອິດ ໃຫ້ສ້າງຝັງຊັນ 'is\_empty()' ເພື່ອກວດເບິ່ງວ່າ stack ແມ່ນຫວ່າງຫຼືບໍ່.

```
1 def pop(stack):
2     return None if is_empty(stack) else stack.pop()
3
4 stack = []
5 print(pop(stack))
```

None



#### Line 1-2

- ຝັງຊັນ pop() ຈະລຶບ ແລະ ສິ່ງຄືນ None object ຖ້າ stack ຫວ່າງເປົ່າ, ແລະ ຖ້າ stack ບໍ່ຫວ່າງ, ມັນຈະລຶບແລະສິ່ງຄືນອີງປະກອບເທິງສຸດຂອງ stack list.
- Stack ເບື້ອງຕື່ນແມ່ນຫວ່າງເປົ່າ, ສະນັ້ນ pop() ສິ່ງຄືນ None object ແລະ ບໍ່ມີ IndexError ເກີດຂຶ້ນ.

### 3. ການສ້າງ Stack

#### 3.2. ສ້າງ Stack ໃນຮູບແບບ class

- | ໃນພາສາ Python, ປະເພດຂໍ້ມູນນາມມະທຳສາມາດຖືກກຳນົດເປັນ class.
- | ປະເພດຂໍ້ມູນ stack ແບບນາມມະທຳ ທີ່ຖືກສ້າງຈາກ list ສາມາດຖືກກຳນົດເປັນ class ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້.

```
1 class Stack:  
2  
3     def __init__(self):  
4         self.stack = []  
5  
6     def is_empty(self):  
7         return True if len(self.stack) == 0 else False  
8  
9     def push(self, item):  
10        self.stack.append(item)  
11  
12    def pop(self):  
13        return None if self.is_empty() else self.stack.pop()
```



#### ແຖວທີ 1-4

- ກຳນົດໂຄງສ້າງຂໍ້ມູນ stack ເປັນ stack class.
- `__init__( self)` constructor ເລີ່ມຕົ້ນບັນດາ field ສະມາຊິກ stack ເປັນ list ເປົ້າຫວ່າງ.

### 3. ການສ້າງ Stack

#### 3.2. ສ້າງ Stack ໃນຮູບແບບ class

| 'push' ສາມາດຖືກສ້າງເປັນ method ຂອງ stack class ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້.

```
1 class Stack:  
2  
3     def __init__(self):  
4         self.stack = []  
5  
6     def is_empty():  
7         return True if len(self.stack) == 0 else False  
8  
9     def push(self, item):  
10        self.stack.append(item)  
11  
12    def pop(self):  
13        return None if self.is_empty() else self.stack.pop()
```

#### Line 9-10

- push() method ໄດ້ຮັບ 'self' ແລະ 'item' ທີ່ຈະເພີ່ມເຂົ້າໃນ stack ເປັນພາລາມີຕີ.
- 'push' ຈະເພີ່ມຂໍ້ມູນເຂົ້າໃສ່ຕໍາແໜ່ງເທິງສຸດຂອງ self.stack list ເຊິ່ງເປັນ field ສະມາຊິກຂອງ stack class.

### 3. ການສ້າງ Stack

#### 3.2. ສ້າງ Stack ໃນຮູບແບບ class

| 'pop' ສາມາດຖືກສ້າງເປັນ method ຂອງ stack class ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້.

```
1 class Stack:  
2  
3     def __init__(self):  
4         self.stack = []  
5  
6     def is_empty():  
7         return True if len(self.stack) == 0 else False  
8  
9     def push(self, item):  
10        self.stack.append(item)  
11  
12    def pop(self):  
13        return None if self.is_empty() else self.stack.pop()
```



Line 12-13

- method pop() ໄດ້ຮັບ 'self' ເປັນພາລາມີເຕີ.
- 'pop' ຈະສົ່ງຄືນ None ຖ້າ self.is\_empty() ເປັນ True ແລະ ຖ້າບໍ່ແມ່ນ, ມັນຈະລຶບ ແລະ ສົ່ງຄືນອີງປະກອບຕົວ  
ເທິງສຸດຂອງ self.stack.

### 3. ການສ້າງ Stack

#### 3.2. ສ້າງ Stack ໃນຮູບແບບ class

- | Method `is_empty()` ທີ່ກວດສອບວ່າ stack ຫວ່າງເປົ້າຄວນທີ່ກໍານົດເປັນ method ຂອງ stack class.

```
1 class Stack:  
2  
3     def __init__(self):  
4         self.stack = []  
5  
6     def is_empty(self):  
7         return True if len(self.stack) == 0 else False  
8  
9     def push(self, item):  
10        self.stack.append(item)  
11  
12    def pop(self):  
13        return None if self.is_empty() else self.stack.pop()
```



Line 6-7

- Method `is_empty()` ສົ່ງຄືນຄ່າ `True` ທ້າຄວາມຍາວຂອງ `self.stack` ມີຄ່າເປັນ 0 ແລະ ທ້າ ບໍ່ແມ່ນ, ມັນຈະສົ່ງຄືນຄ່າ `False`.

### 3. ການສ້າງ Stack

#### 3.2. ສ້າງ Stack ໃນຮູບແບບ class

| Code ຕໍ່ໄປນີ້ທີດສອບໂຄງສ້າງຂໍ້ມູນ stack ທີ່ຖືກກຳນົດເປັນ class ກ່ອນໜັນນີ້.

```
1 stack = Stack()
2 stack.push("A")
3 stack.push("B")
4 print(stack.pop())
5 stack.push("C")
6 print(stack.pop())
7 print(stack.pop())
8 print(stack.pop())
```

B  
C  
A  
None

#### ແຖວທີ 1-8

- ຕົວປ່ຽນ stack ຈະບັນທຶກ object instance ຂອງ stack class ທີ່ຖືກສ້າງຂຶ້ນໂດຍການເອີ້ນໃຊ້ constructor Stack().
- Method push() ແລະ pop() ຂອງພັງຊັນ stack object ສາມາດຖືກເອີ້ນໃຊ້ໂດຍໃຊ້ຕົວດຳເນີນການອ້າງອີງ ''
- ພິມຄ່າທີ່ສົ່ງກັບຄືນຂອງພັງຊັນ stack.pop() ໂດຍໃຊ້ພັງຊັນ print() ເພື່ອກຳນົດລຳດັບການເອົາຄ່າອອກຈາກ stack.



## One More Step

- | ຝັງຊັນ pop() ໃຊ້ method pop() ຂອງ list ເພື່ອລຶບອົງປະກອບເທິງສຸດຂອງ stack ແລະ ສິ່ງຄ່າຂອງມັນຄືນກັບໄປ. ເຮົາຈະສ້າງຝັງຊັນ peek() ທີ່ຈະເຮັດໃຫ້ເຮົາຮູ້ວ່າອົງປະກອບໄດ້ຢູ່ເທິງສຸດຂອງ stack ໄດ້ແນວໃດ?
- | ສໍາລັບ method peek() ພຽງແຕ່ສິ່ງຄ່າອົງປະກອບຕົວເທິງສຸດອອກມາສະແດງໂດຍບໍ່ມີການລຶບອົງປະກອບດັ່ງກ່າວ.

```
1 def peek(stack):  
2     return None if is_empty(stack) else stack[-1]
```

- | ເຮົາຍັງສາມາດກຳນົດ method peek() ເປັນ method ທີ່ເປັນສະມາຊິກຂອງ stack class ທີ່ຖືກກຳນົດໄວ້ກ່ອນໜີ້ດັ່ງຕໍ່ໄປນີ້.

```
15 def peek(self):  
16     return None if is_empty(self.stack) else self.stack[-1]
```



## TIP

- | ເມື່ອກຳນົດຄ່າດັດສະນີເປັນ -1 ໃນບັນ list ຂອງ Python, ເຮົາສາມາດເຂົ້າຫາອົງປະກອບເທິງຂອງ list.
- |  $\text{stack}[-1] == \text{stack}[\text{len(stack)} - 1]$

| Let's code

# 1. បំណុលទារាងសិមតុនខ្លះវិញលើប៊ូលីម

## 1.1. និយាយខ្លះបំណុល

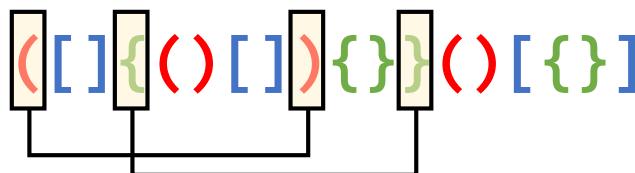
- | ត្រូវការពិនិត្យ algorithm ថា បំណុលទារាងខ្លះវិញលើប៊ូលីម ត្រូវបានដោះស្រាយ។
- | ត្រូវពិនិត្យថា Valid តាមខ្លះទារាងខ្លះវិញលើប៊ូលីម និង Invalid.
- | សិមតុនថា មិនមែនសាមសួលប៊ូលីម និងត្រូវបានដោះស្រាយ។
  - ▶ វិញខ្លះ: [], វិញបិកការ: {}, វិញលើប៊ូលីម: ()
- | វិញលើប៊ូលីមត្រូវបានដោះស្រាយ និងត្រូវបានដោះស្រាយ នៅពេលបង្កើតឡើង។
  1. បំណុលទារាងខ្លះវិញលើប៊ូលីម ត្រូវបានដោះស្រាយ និងត្រូវបានដោះស្រាយ នៅពេលបង្កើតឡើង។
  2. បំណុលទារាងខ្លះវិញលើប៊ូលីម ត្រូវបានដោះស្រាយ និងត្រូវបានដោះស្រាយ នៅពេលបង្កើតឡើង។
  3. បំណុលទារាងខ្លះវិញលើប៊ូលីម ត្រូវបានដោះស្រាយ និងត្រូវបានដោះស្រាយ នៅពេលបង្កើតឡើង។

[ ] { } ( )

# 1. ប៊ាន្ហាកវាមសិមតុនខោងវីរ៉ែល

## 1.2. ពិវិះយោង Input & output

| តែងបែន្នូលសម្រាប់ការបង្កើតកម្មវិធី។

Valid	Invalid
<code>[]{}()</code>	(
<code>((()))</code>	(]
<code>({[]})</code>	)()
<code>[][]{}()({[]})((())())</code>	([()])
<code>([]{}()[]{}{})()[]{}[]</code>	

# 1. បំណុលការសមតុល្យរបៀប

## 1.3. Codes ហើយដោះស្រាយការណ៍

- | ផ្សាយជាមួយនឹង check\_parentheses() រកបខ្ល័ត្តការណ៍ថា ចំណាំនេះត្រូវបានដោះស្រាយឡើង ឬត្រូវបានបញ្ជាក់ថា មិនត្រូវបានដោះស្រាយឡើង។
- | ឱ្យបង្ហាញពីលទ្ធផលនៃការដោះស្រាយនេះ។

```
1 str = input("Input a string of parentheses: ")
2 print("Valid" if check_parentheses(str) else "Invalid")
```

```
Input a string of parentheses: ([]{}())
Valid
```

```
1 str = input("Input a string of parentheses: ")
2 print("Valid" if check_parentheses(str) else "Invalid")
```

```
Input a string of parentheses: ([]{}())
Invalid
```

# 1. ບັນຫາຄວາມສົມດຸນຂອງວິງເລັບ

## 1.3. Codes ທີ່ແກ້ໄຂບັນຫາດັ່ງກ່າວ

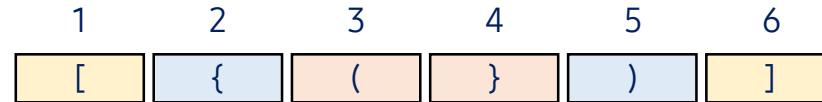
| ຕໍ່ໄປນີ້ສະແດງໃຫ້ເຫັນຜົນໄດ້ຮັບສຸດທ້າຍຂອງພິຈາລະນາ check\_parentheses().

```
1 def check_parentheses(expr):
2     opening = "[{("
3     closing = "])})"
4     stack = Stack()
5     for char in expr:
6         if char in opening:
7             stack.push(char)
8         elif char in closing:
9             if stack.is_empty():
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return stack.is_empty()
```

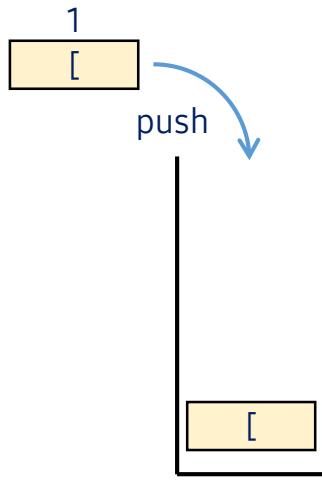
# 1. ប៊ូនហាកវាមសិមតុនខទ្ទុវីងលេប

## 1.4. រោងចាយប៊ូនហាកវាមសិមតុនខទ្ទុវីងលេបដើម្បីការបញ្ចប់ក្នុងក្រុងក្រោម

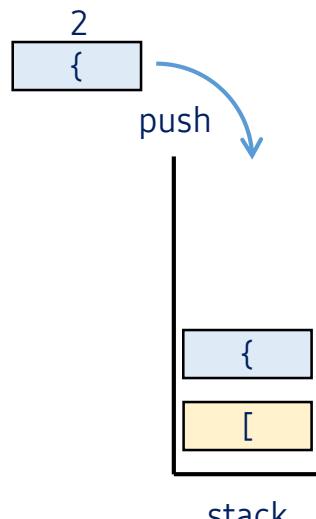
| ឲ្យក្នុងស្ថាយខ្លួន stack ដើម្បីការបញ្ចប់ក្នុងក្រុងក្រោម។ តើយាំងដឹងថាគ្នុងក្រុងក្រោមនេះមានអ្វីដឹង? តើយាំងដឹងថាគ្នុងក្រុងក្រោមនេះមានអ្វីដឹង?



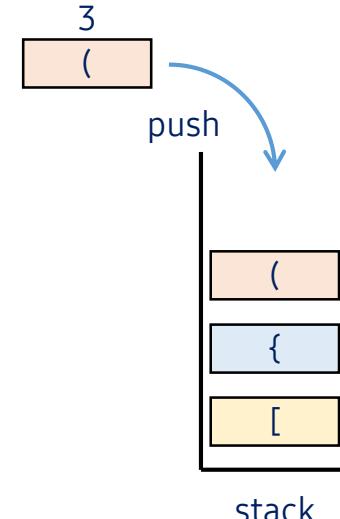
1. Push វីងខ្លួនដែលបានបើកប្រឈម  
stack.



2. Push វីងបីកភាពដែលបានបើកប្រឈម  
stack.

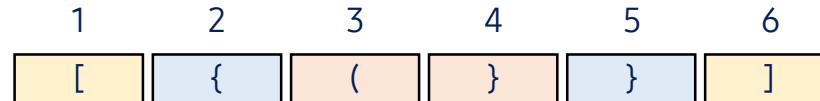


3. Push វីងលេបដែលបានបើកប្រឈម  
stack.

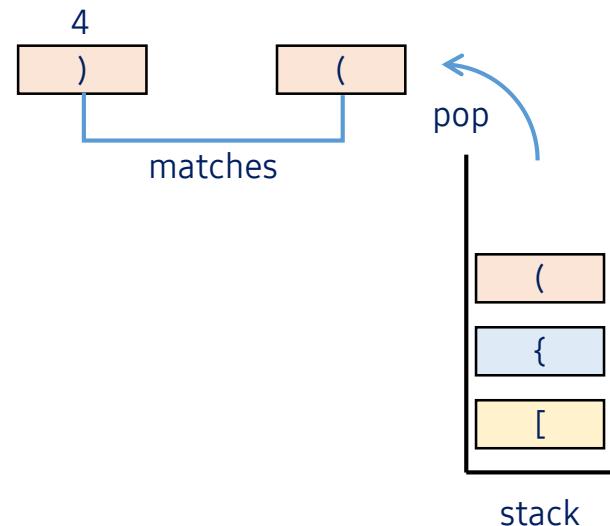


# 1. บันทึกความสัมฤทธิ์ของวิ่งเล็บ

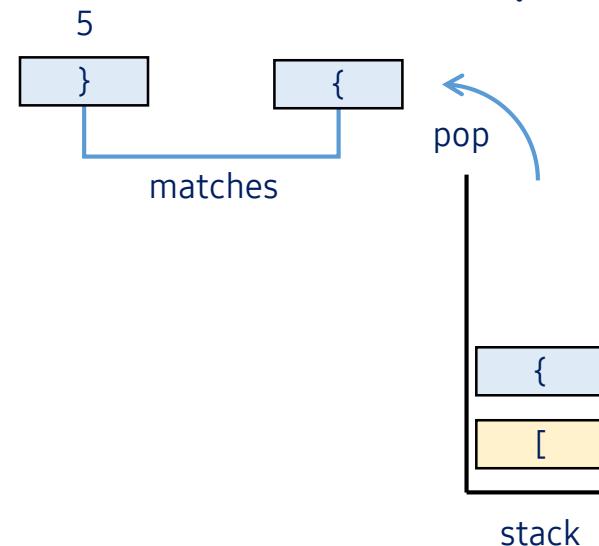
## 1.4. แก้ไขบันทึกด้วย stack



4. เมื่อมีวิ่งเล็บเปิด, pop เอิวิ่งเล็บเปิดออกจาก stack เพื่อเอามาจับคู่กัน.

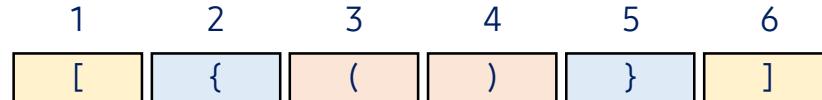


5. เมื่อมีวิ่งปีกกาเปิด, pop เอิวิ่งปีกกาเปิดออกจาก stack เพื่อเอามาจับคู่กัน.

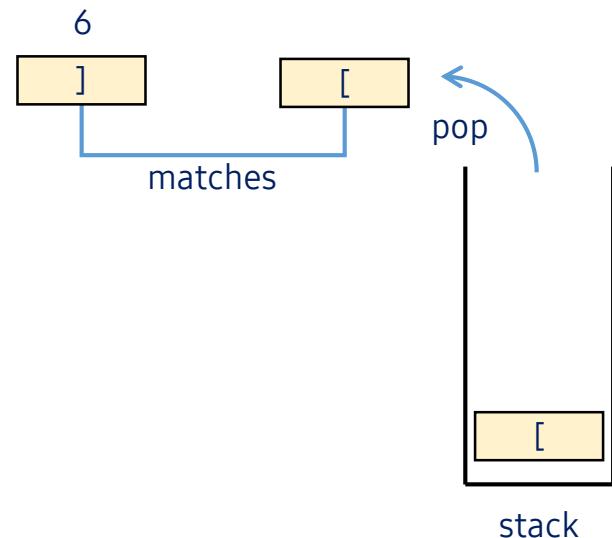


# 1. ບັນຫາຄວາມສົມດຸນຂອງວິ່ງເລັບ

## 1.4. ແກ້ໄຂບັນຫາດັ່ງກ່າວໂດຍໃຊ້ stack



6. ເມື່ອມີວິ່ງຂຶ້ນ, pop ເອົາວິ່ງຂຶ້ນເປີດອອກຈາກ stack  
ເພື່ອເອົາມາຈັບຄຸ້ກັ້ນ.



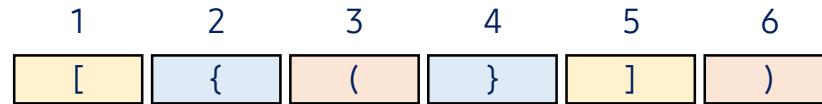
7. ຖ້າ stack ຫວ່າງເປົ່າ ຫຼັງຈາກກວດສອບທຸກປະເພດ  
ວິ່ງເລັບໃນຂໍ້ຄວາມ ແລ້ວຖືວ່າວິ່ງເລັບທັງໝົດແມ່ນສົມ  
ດຸນກັນແລ້ວ.



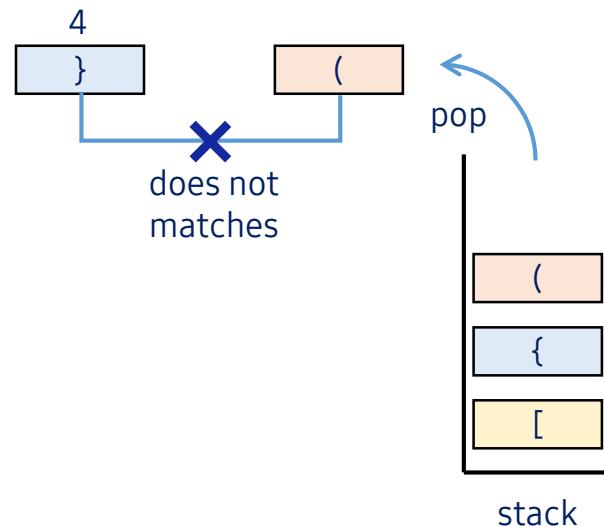
# 1. ບັນຫາຄວາມສົມດຸນຂອງວິ່ງເລັບ

## 1.4. ແກ້ໄຂບັນຫາດັ່ງກ່າວໂດຍໃຊ້ stack

| ຕໍ່ໄປນີ້ສະແດງໃຫ້ເຫັນວິ່ງເລັບທີ່ບໍ່ກົງກັນ.



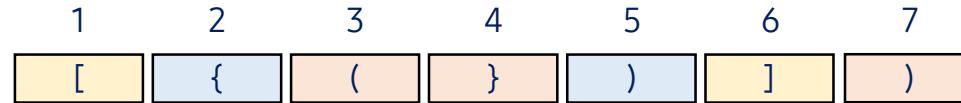
ກໍລະນີ 1. ວິ່ງເລັບປິດບໍ່ກົງກັບວິ່ງເລັບທີ່ pop ອອກມາຈາກ stack.



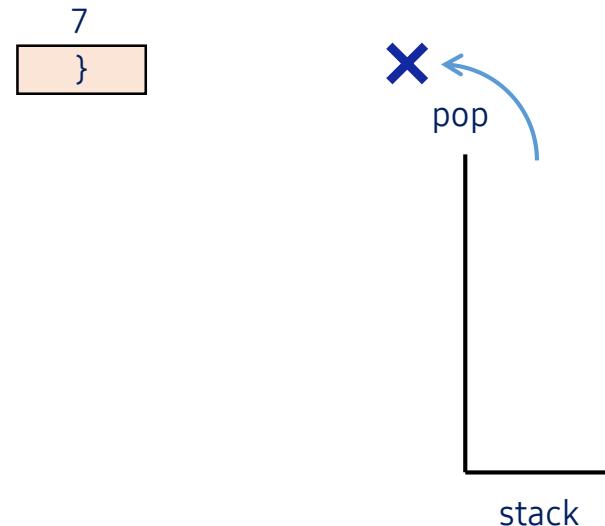
# 1. ບັນຫາຄວາມສົມດຸນຂອງວົງເລັບ

## 1.4. ແກ້ໄຂບັນຫາດັ່ງກ່າວໂດຍໃຊ້ stack

| ຕໍ່ໄປນີ້ສະແດງໃຫ້ເຫັນວົງເລັບທີ່ບໍ່ກົງກັນ.



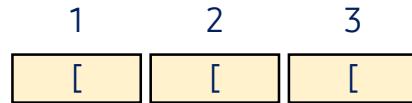
ກໍລະນີ 2. ຖ້າມີວົງເລັບປິດແຕ່ stack ຫວ່າງເປົ່າ.



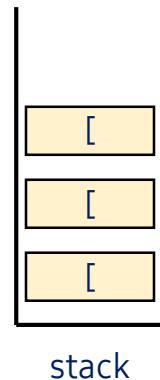
# 1. ບັນຫາຄວາມສົມດຸນຂອງວິ່ງເລັບ

## 1.4. ແກ້ໄຂບັນຫາດັ່ງກ່າວໂດຍໃຊ້ stack

| ຕໍ່ໄປນີ້ສະແດງໃຫ້ເຫັນວິ່ງເລັບທີ່ບໍ່ກົງກັນ.



ກໍລະນີ 3. ຖ້າ stack ບໍ່ແມ່ນຫວ່າງເປົ້າ ຫຼັງຈາກການກວດສອບ  
ຂໍຄວາມທັງຫມີດມີແຕ່ວິ່ງເລັບເປີດ.



# 1. ບັນຫາຄວາມສົມດຸນຂອງວິ່ງເລັບ

## 1.5. ລົງມືປະຕິບັດ ແລະ ຂຽນໂປຣແກຣມ

| ກຳນົດ function ດັ່ງນີ້ພື້ນກວດສອບວ່າວິ່ງເລັບສົມດຸນກັນຫຼືບໍ່.

```
1 def check_parentheses(expr):
2     opening = "[{("
3     closing = "])})"
4     stack = Stack()
5     for char in expr:
6         if char in opening:
7             stack.push(char)
8         elif char in closing:
9             if stack.is_empty():
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return stack.is_empty()
```

### Line 1

- ຟັງຊັ້ນ chek\_parentheses() ໄດ້ຮັບຂໍ້ຄວາມ 'expr' ເປັນຕົວກຳນົດຂໍ້ມູນປ້ອນເຂົ້າ.
- ຟັງຊັ້ນນີ້ສິ່ງຄ່າ True ຖ້າມີວິ່ງເລັບທີ່ກົງກັນຢູ່ໃນຂໍ້ຄວາມ expr ແລະ ຖ້າບໍ່ແມ່ນດັ່ງນັ້ນ, ມັນຈະສິ່ງຄືນ False.

## 1. ប័ណ្ណាកវាមសិមតុនខេទ្យវិងលេប

## 1.5. ລົງມືປະຕິບັດ ແລະ ຂຽນໂປຣແກຣມ

| เลี้มตົ່ມໂຄງສ້າງຂໍ້ມູນທີ່ຕ້ອງການສໍາລັບການຂຽນໂປຣແກຣມຕາມ algorithm ດັ່ງຕໍ່ໄປນີ້.

```
1 def check_parentheses(expr):
2     opening = "[{("
3     closing = "])})"
4     stack = Stack()
5     for char in expr:
6         if char in opening:
7             stack.push(char)
8         elif char in closing:
9             if stack.is_empty():
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return stack.is_empty()
```

## Line 2-4

- 'ການເປີດ' ແລະ 'ການປິດ' ເລີ່ມຕົ້ນວົງເລັບເປົ້າແລະປິດຕາມລໍາດັບ. ສັງເກດເຫັນວ່າວົງເລັບທີ່ກົງກັນແມ່ນຕັ້ງຢູ່ໃນຕາແຫັນນັ່ງດຽວກັນຂອງຂໍ້ຄວາມ. ດັດຊະນີຂອງຂໍ້ຄວາມຈະກວດສອບການເພື່ອທຽບຄຸ້ງກັນ.
  - ຕົວປ່ຽນ stack ແມ່ນເລີ່ມຕົ້ນກັບ object instance ຂອງ stack class ທີ່ຖືກກຳນົດໄວ້ກ່ອນໜີ້ນີ້.

# 1. ບັນຫາຄວາມສົມດຸນຂອງວິງເລັບ

## 1.5. ລົງມືປະຕິບັດ ແລະ ຂຽນໂປຣແກຣມ

| ໃຊ້ຊຸດຄຳສັ່ງທີ່ປະມວນຜົນວິງເລັບເປີດຂອງແຕ່ລະຕົວອັກສອນໃນຂໍຄວາມທີ່ໃຫ້.

```
1 def check_parentheses(expr):
2     opening = "[{("
3     closing = "])})"
4     stack = Stack()
5     for char in expr:
6         if char in opening:
7             stack.push(char)
8         elif char in closing:
9             if stack.is_empty():
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return stack.is_empty()
```

### Line 5-7

- ກວດເບິ່ງທຸກຕົວອັກສອນ (char) ໃນຂໍຄວາມ expr.
- ຖ້າ char ແມ່ນວິງເລັບເປີດ, ໃຫ້ push ມັນເຂົ້າເກັບໃນ stack.

# 1. ບັນຫາຄວາມສົມດຸນຂອງວິ່ງເລັບ

## 1.5. ລົງມືປະຕິບັດ ແລະ ຂຽນໂປຣແກຣມ

| ໃຊ້ຊຸດຄຳສັ່ງທີ່ປະມວນຜົນວິ່ງເລັບປິດຂອງແຕ່ລະຕົວອັກສອນໃນຂໍ້ຄວາມທີ່ໃຫ້.

```

1 def check_parentheses(expr):
2     opening = "[{("
3     closing = "])})"
4     stack = Stack()
5     for char in expr:
6         if char in opening:
7             stack.push(char)
8         elif char in closing:
9             if stack.is_empty():
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return stack.is_empty()
```

### Line 8-12

- ຖ້າ 'char' ແມ່ນວິ່ງເລັບປິດແທນທີ່ຈະເປັນວິ່ງເລັບເປີດ, ກວດເບິ່ງວ່າມັນກິງກັບວິ່ງເລັບອື່ນໜີ້ບໍ່.
- ຖ້າມີວິ່ງເລັບປິດ ແລະ stack ຫວ່າງເປົ່າ, ສະແດງວ່າວິ່ງເລັບບໍ່ກິງກັນ (ກໍລະນີທີ 2).
- ຖ້າມີວິ່ງເລັບປິດ ແລະ ຕຳແໜ່ງຕົວອັກສອນທີ່ເອີມຈາກ stack ແລະ ຕຳແໜ່ງຂໍ້ຄວາມຂອງ char ແຕກຕ່າງກັນ, ສະແດງວ່າວິ່ງເລັບບໍ່ກິງກັນ (ກໍລະນີທີ 1)

# 1. ບັນຫາຄວາມສົມດຸນຂອງວິ່ງເລັບ

## 1.5. ລົງມືປະຕິບັດ ແລະ ຂຽນໂປຣແກຣມ

| ຫຼັງຈາກການກວດສອບຂໍ້ຄວາມຫັງໜີດ, ໃຫ້ສົ່ງຄືນຜົນໄດ້ຮັບດັ່ງຕໍ່ໄປນີ້.

```
1 def check_parentheses(expr):
2     opening = "[{("
3     closing = "])})"
4     stack = Stack()
5     for char in expr:
6         if char in opening:
7             stack.push(char)
8         elif char in closing:
9             if stack.is_empty():
10                 return False
11             if opening.index(stack.pop()) != closing.index(char):
12                 return False
13     return stack.is_empty()
```

Line 13

- ຖ້າ stack ບໍ່ຫວ່າງຫຼັງຈາກກວດເບີ່ງຕົວອັກສອນຫັງໜີດຂອງຂໍ້ຄວາມ expr ທີ່ປະກິດມີແຕ່ວິ່ງເລັບເປີດ ກໍ່ສະແດງວ່າວິ່ງເລັບບໍ່ກົງກັນ (ກໍລະນີທີ 3).
- ຖ້າ stack ຫວ່າງເປົ່າ, ມີວິ່ງເລັບທີ່ກົງກັນ, ຈະສົ່ງຜົນໄດ້ຮັບຂອງ stack ທີ່ຫວ່າງເປົ່າ.

# 1. ບັນຫາຄວາມສົມດຸນຂອງວິງເລັບ

## 1.5. ລົງມືປະຕິບັດ ແລະ ຂຽນໂປຣແກຣມ

| ກວດສອບ ຂໍ້ມູນທີ່ບໍອນເຂົ້າຕ່າງໆຂອງພັງຊັ້ນ check\_parentheses() ຜ່ານ code ຕໍ່ໄປນີ້.

```
1 str = input("Input a string of parentheses: ")  
2 print("Valid" if check_parentheses(str) else "Invalid")
```

Input a string of parentheses:

# | Pop quiz

## Quiz. #1

| ຂຽນຜົນໄດ້ຂອບຂອງ codes ຈາກຕົວຢ່າງທີ່ໃຊ້ stack.

```
1 stack = Stack()  
2 stack.push("Banana")  
3 stack.push("Apple")  
4 stack.push("Tomato")  
5 stack.pop()  
6 stack.push("Strawberry")  
7 stack.push("Grapes")  
8 stack.pop()  
9 print(stack.stack)
```

## Quiz. #2

| ຂຽນຜົນໄດ້ຮັບຂອງ codes ຈາກຕົວຢ່າງທີ່ໃຊ້ stack.

```
1 stack = Stack()
2 items = [10 * i for i in range(1, 10)]
3 for item in items:
4     stack.push(item)
5     if (item // 10) % 2 == 0:
6         stack.pop()
7 print(stack.stack)
```

# | Pair programing



## Pair Programming Practice

### ແນວທາງ, ກິນໄກ ແລະ ແຜນສຸກເສີນ

ການກະກຽມການສ້າງໂປຣແກຣມຮ່ວມກັນເປັນຄູ່ກ່ຽວຂ້ອງກັບການໃຫ້ຄໍາແນະນຳແລະກິນໄກເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄ່າຢ່າງຖືກຕ້ອງແລະ ໃຫ້ເຊົາເຈົ້າເຮັດວຽກເປັນຄູ່. ຕົວຢ່າງ, ນັກຮຽນຄວນປ່ຽນ "ເຮັດ." ການກະກຽມທີມປະສິດທິຜົນຕ້ອງໃຫ້ມີແຜນການສຸກເສີນໃນກໍລະນີທີ່ຄ່າຮ່ວມງານທັນນີ້ບໍ່ຢູ່ທີ່ຕັດສິນໃຈທີ່ຈະບໍ່ເຊົາຮ່ວມດ້ວຍເຫດຜົນໃດທີ່ນີ້ ຫຼືດ້ວຍເຫດຜົນອື່ນ. ໃນກໍລະນີເຫຼົ້ານີ້, ມັນເປັນສິ່ງສຳຄັນທີ່ຈະເຮັດໃຫ້ມັນຊັດເຈນວ່ານັກຮຽນທີ່ມີປະຕິບັດໜ້າທີ່ຢ່າງຫ້າວຫ້ານຈະບໍ່ຖືກລົງໂທດຍ້ອນວ່າການຈັບຄູ່ບໍ່ໄດ້ຜົນດີ.

### ການຈັບຄູ່ທີ່ສ້າຍຄືກັນ, ບໍ່ຈໍາເປັນຕ້ອງເຫັນກັນ, ຄວາມສາມາດເປັນຄູ່ຮ່ວມງານ

ການຂຽນໂປຣແກຣມຄູ່ຈະມີປະສິດທິພາບເມື່ອນັກຮຽນຕັ້ງໃຈຮ່ວມກັນເຮັດວຽກ, ຊຶ່ງວ່ານີ້ຈໍາເປັນຕ້ອງມີຄວາມຮັບເຫັນກັນ, ແຕ່ຕ້ອງມີຄວາມສາມາດເຮັດວຽກເປັນຄູ່ຮ່ວມງານ. ການຈັບຄູ່ນັກຮຽນທີ່ບໍ່ສາມາດເຂົາກັນໄດ້ມັກຈະເຮັດໃຫ້ການມີສ່ວນຮ່ວມທີ່ບໍ່ສົມດຸນກັນ. ຄສອນຕ້ອງເນັນຫຼັກວ່າການຂຽນໂປຣແກຣມຄູ່ບໍ່ແມ່ນຍັດທະສາດ "divide-and-conquer", ແຕ່ຈະເປັນຄວາມພະຍາຍາມຮ່ວມມືເຮັດວຽກທີ່ເທົ່າຈີງໃນທຸກໆດ້ານສໍາລັບໂຄງການທັງໝາດ. ອຸດວນຫຼືກເວັ້ນການຈັບຄູ່ນັກຮຽນທີ່ອຳນ້າຍກັບນັກຮຽນທີ່ເກົ່າໜ້າຍ.

### ກະຕຸນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈຸງໃຈພິເສດ

ການສະໜັບແນງຈຸງໃຈພິເສດສາມາດຊ່ວຍກະຕຸນນັກຮຽນໃຫ້ຈັບຄູ່, ໂດຍສະເພາະກັບນັກຮຽນທີ່ມີຄວາມສາມາດໜ້າຍ. ຈະເຫັນວ່າມັນເປັນປະໂຫຍດທີ່ຈະໃຫ້ນັກຮຽນຈັບຄູ່ເຮັດວຽກຮ່ວມກັນພຽງແຕ່ທັນນີ້ຫຼືສອງວຽກເທົ່ານັ້ນ.



## ການປະຕິບັດງານຕົວຈິງໃນການຊຽນໂປຣແກຣມເປັນຄຸ້ມ



| ป้องกันภัยในภาคเรียนรู้ทั่วไปร่วมกัน

ສິ່ງທີ່ຫາຍສໍາລັບຄູ່ແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງບກຄົນ, ໃນຂະນະທີ່ນຳໃຊ້ຜົນປະໂຫຍດຂອງການຮ່ວມມື. ຈະຮຸ້ໄດ້ແນວໃດວ່ານີ້ກາຮຽນຕັ້ງໃຈເຮັດວຽກ ຫຼື ກົງແຮງງານຜ່ຽວມາງານ? ຜູ້ຊ່ວງຊານແນະນຳໃຫ້ທີ່ບໍ່ທວນຄົນການອອກແບບຫຼັກສູດ ແລະ ການປະເມີນ ພ້ອມທັງປຶກສາຫາລືຢ່າງຈະແຈ້ງ ແລະ ຊັດເຈນກຽວກັບພື້ນຖານທີ່ຈະຖືກຕິຄວາມວ່າຂີ້ຕົວ. ຜູ້ຊ່ວງຊານເນັ້ນໜັກໃຫ້ຄູ່ເຮັດການມອບໝາຍໃຫ້ມີຄວາມໝາຍຕົນກາຮຽນ ແລະ ອະທິບາຍຄຸນຄ່າຂອງສິ່ງທີ່ນີ້ກາຮຽນຈະຮຽນຮູ້ໂດຍການເຮັດສໍາເລັດ.

ສະພາບແວດລ້ອມການຮຽນຮູ້ໃນການຮ່ວມມື

สะพานแ渭ล้อมกวนหຽนหັງໃນຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຫຽວກ້ອງໃຫ້ນັກຫຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳກັນຫຽນຮັງ. ສະພາບແວດລ້ອມກັນຫຽນຮັງຮ່ວມກັນສາມາດມີສ່ວນຮ່ວມທັງກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນໂດຍກົງ. ຕົວຢ່າງ, ນັກສຶກສາຄູ່ເຮັດວຽກມອບທາມຍັກຮ່ວມກັນໃນການຂຽນໂປຣແກຣມ; ນັກສຶກສາກ່າມນ້ອຍງ່າສືນທະນາຄໍາຕອບທີ່ເປັນໄປໄດ້ຕໍ່ກັບຄໍາຖາມຂອງອາຈານໃນລະຫວ່າງການບັນຍາຍ; ແລະ ນັກຫຽນເຮັດວຽກຮ່ວມກັນນອກຫ້ອງຫຽນເພື່ອຫຽນຮັງແວດຄວາມຄົດໃຫ້. ການຫຽນຮັງການຮ່ວມມືແມ່ນແຕກຕ່າງຈູາກໂຄງການທີ່ນັກຫຽນ “divide and conquer.” ເມື່ອນັກຫຽນແບ່ງວຽກກັນ, ແຕ່ລະຄົນຮັບຜິດຊອບພຽງແຕ່ສ່ວນຫນຶ່ງຂອງການແກ້ໄຂບັນຫາ ແລະ ຈະບໍ່ຄ່ອຍມີບັນຫາຫຍັງໃນການເຮັດວຽກຮ່ວມກັບຄົນອື່ນໃນທີ່ມ. ໃນສະພາບແວດລ້ອມກັນເຮັດວຽກຮ່ວມກັຍ, ນັກຫຽນມີສ່ວນຮ່ວມໃນການສືນທະນາປົກສາຫາລືເຊິ່ງກັນແລະວັນ.

**Q1.** เอกสารนี้ HTML ประกอบด้วยหลายแท็กตั้งที่สะແດງข้างลุ่มนี้. ให้ช่วยโปรแกรมที่กิจ  
กับแท็กเอกสารนี้ HTML.

งานจับคู่แท็ก HTML

```
<html>
<body>
<h1>Hello, World!</h1>
<p> We are learning the art of coding
with Python programming language.
Here we are learning ... </p>
<ul>
<li> Data Structures, </li>
<li> Algorithms, </li>
<li> and Computational Thinking,
    eventually. </li>
</ul>
</body>
</html>
```

## Hello, World!

We are learning the art of coding  
with Python programming  
language. Here we are learning ...

- Data Structures,
- Algorithms,
- and Computational Thinking,  
eventually.

## ໂອກະສານອ້າງອີງ

An HTML element is defined by a start tag, some content, and an end tag.

### HTML Elements

The HTML **element** is everything from the start tag to the end tag:

`<tagname>Content goes here...</tagname>`

Examples of some HTML elements:

`<h1>My First Heading</h1>`

`<p>My first paragraph.</p>`

[https://www.w3schools.com/html/html\\_elements.asp](https://www.w3schools.com/html/html_elements.asp)



TIP

- | ในภาษา Python, method find() ที่ก็มำใช้เพื่อຊອກຫາตำาແທນมັງເລີ່ມຕົ້ນຂອງຂໍ້ຄວາມຍ່ອຍ. ຊອກຫາ '<' ແລະ '>' ເພື່ອຈຳແນກແທັກເອກະສານ HTML.
- | ຖ້າແທັກທີ່ຖືກຈຳແນກປະກອບມີ '/' ນັ້ນແມ່ນແທັກເປີດ ແລະ ຖ້າບໍ່ແມ່ນດັ່ງນັ້ນ, ຈະແມ່ນແທັກປິດ. ກວດເບິ່ງການຈັບຄຸ້ແທັກ HTML ໃນວິທີທີ່ຄ້າຍຄືກັນໂດຍການໃຊ້ stack ເພື່ອຈັບຄຸ້ວິງເລັບ.

```
1 text = input("Input the string of HTML document: ")
```

Input the string of HTML document: <html> <body> <h1>Hello, World!</h1> <p> We are learning the art of coding with Python programming language. Here we are learning ... </p> <ul> <li> Data Structures, </li> <li> Algorithms, </li> <li> and Computational Thinking, eventually. </li> </ul> </body> </html>

```
1 start = text.find('<')
2 while start != -1:
3     end = text.find('>', start + 1)
4     tag = text[start:end+1]
5     print(tag, end = ' ')
6     start = text.find('<', end + 1)
```

<html> <body> <h1> </h1> <p> </p> <ul> <li> </li> <li> </li> <li> </li> </ul> </body> </html>