Unit 24.

ການຄົ້ນຫາແບບເຮັດຕາມລຳດັບ (Sequential Search)

Learning objectives

- ✓ ໃຫ້ເຂົ້າໃຈບັນຫາການຄົ້ນຫາ ແລະ ສາມາດແກ້ໄຂບັນຫາດັ່ງກ່າວ ໂດຍໃຊ້ການຄົ້ນຫາແບບຕາມລຳດັບ.
- ✓ ສາມາດສ້າງວິທີການຄົ້ນຫາແບບຕາມລຳດັບໂດຍໃຊ້ພາສາ Python.
- ✓ ສາມາດນຳໃຊ້ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບຕາມລຳດັບເພື່ອແກ້ໄຂບັນຫາໃນການຊອກຫາຄ່າສູງສຸດທີ່ມີຫຼາຍຄ່າ.

Learning overview

- ✓ ເຂົ້າໃຈບັນຫາການຄົ້ນຫາທີ່ຊອກຫາອົງປະກອບທີ່ລະບຸໄວ້ຈາກຂໍ້ມູນທີ່ຈັດລຽງໄວ້ແລ້ວແບບສຸ່ມ.
- ✓ ຮຽນຮູ້ວິທີການແກ້ໄຂບັນຫາການຄົ້ນຫາໂດຍໃຊ້ການຄົ້ນຫາແບບຕາມລໍາດັບ.
- ✓ ຮຽນຮູ້ວິທີການຂຽນຂັ້ນຕອນວິທີໃນການຄົ້ນຫາແບບຕາມລໍາດັບດ້ວຍພາສາ Python ແລະ ວິເຄາະເວລາທີ່ໃຊ້ໃນການຄົ້ນຫາກໍລະນີທີ່ດີທີ່ສຸດ, ບໍ່ດີທີ່ສຸດ ແລະ ໂດຍສະເລ່ຍ.

Concepts you will need to know from previous units

- ✓ ວິທີການເຂົ້າໄປຫາທຸກອົງປະກອບໃນລາຍການໃດໜຶ່ງໂດຍໃຊ້ iterator
- ✓ ວິທີການປຽບທຽບຂະໜາດຂອງອົງປະກອບລາຍການ
- ✓ ວິທີການໃຊ້ແນວຄວາມຄິດຂອງ Big-O ເພື່ອປະເມີນເວລາທີ່ໃຊ້ຫຼາຍສຸດໃນການເຮັດວຽກຂອງຂັ້ນຕອນວິທີ

Keywords

Searching Problem

Sequential Search

Linear Time

Mission

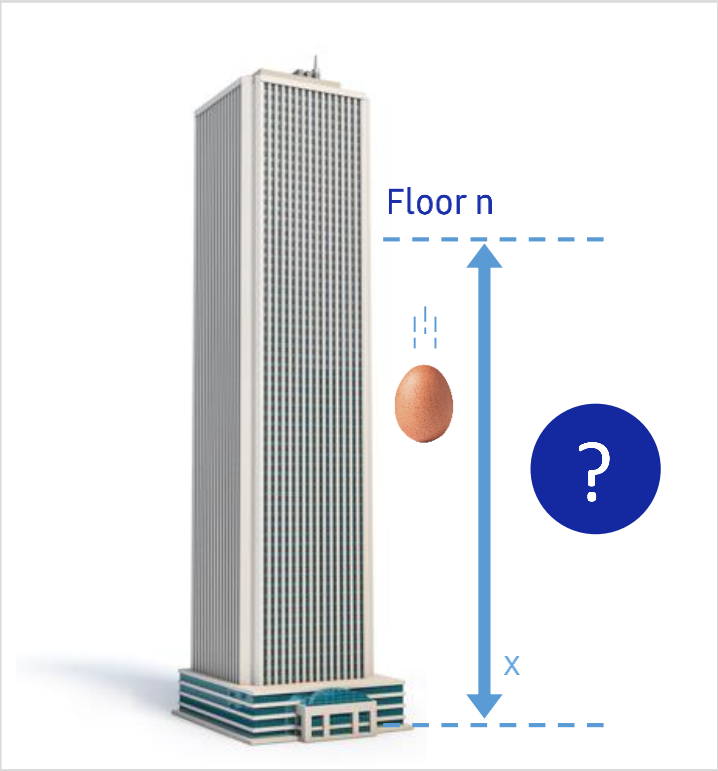
1. Real world problem

1.1. ການຕົກຂອງໄຂ່

- ສິ່ງທ້າທາຍໃນການຕົກຂອງໄຂ່ໄດ້ຖືກອອກແບບສຳລັບ ນັກຮຽນທີ່ມັກກ່ຽວກັບວິທະຍາສາດ.
- ນັກຮຽນຄວນເຮັດອຸປະກອນປ້ອງກັນໄຂ່ ເພື່ອບໍ່ໃຫ້ໄຂ່ແຕກ ເຖິງແມ່ນວ່າຈະຕົກຈາກບ່ອນສູງກໍຕາມ.
- ຈິນຕະນາການວ່າເຈົ້າໄດ້ເຂົ້າຮ່ວມໃນສິ່ງທ້າທາຍຂອງການຖິ້ມ ໄຂ່ ແລະ ເຮັດອຸປະກອນປ້ອງກັນໄຂ່ດ້ວຍຄວາມຄິດທີ່ດີ.
- ເຈົ້າຈະເຮັດແນວໃດຖ້າເຈົ້າຕ້ອງການຄິດໄລ່ຫາຊັ້ນທີ່ສູງທີ່ສຸດ ໃນອາຄານ 100 ຊັ້ນຫຼັງໜຶ່ງທີ່ປ່ອຍໄຂ່ລົງແລ້ວຍັງບໍ່ແຕກ?
- ລອງມາສ້າງຂັ້ນຕອນວິທີເພື່ອແກ້ໄຂບັນຫາການຕົກຂອງໄຂ່.

2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່

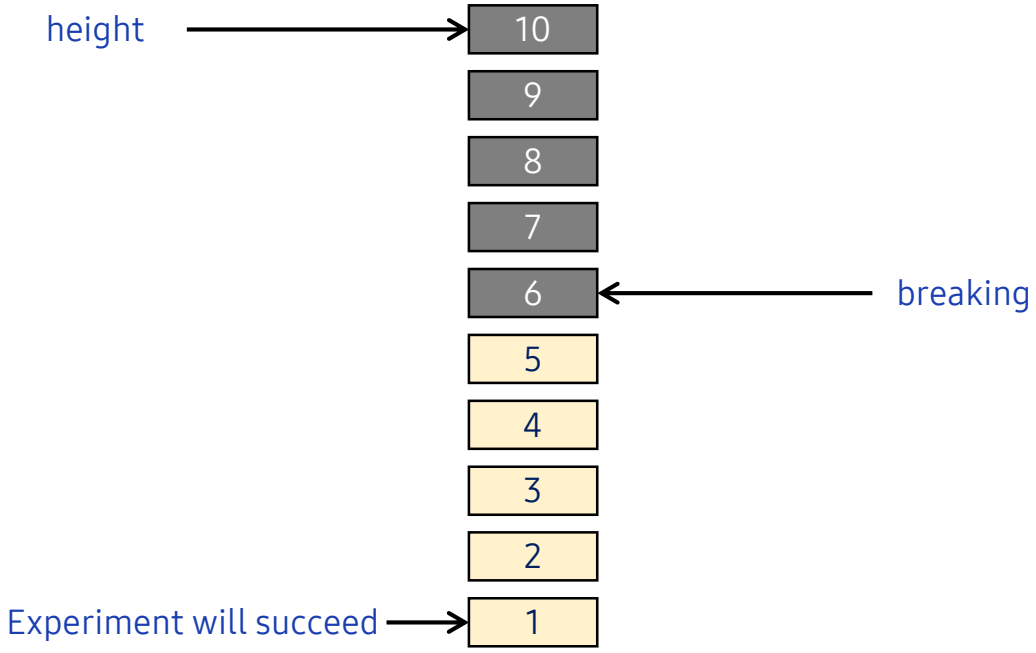


- ຊອກຫາຊັ້ນສູງທີ່ສຸດທີ່ເປັນໄປໄດ້ ສໍາລັບການປ່ອຍໄຂ່ລົງແລ້ວບໍ່ແຕກ ຕາມທີ່ໄດ້ກໍານົດຈໍານວນຊັ້ນຂອງອາຄານທັງໝົດ.
- ຊັ້ນສູງທີ່ສຸດສໍາລັບການປ່ອຍໄຂ່ລົງແລ້ວຍັງບໍ່ແຕກແມ່ນຖືກກໍານົດ ໂດຍຕົວເລກສຸ່ມລະຫວ່າງ 1 ຫາຄ່າສູງສຸດຂອງຊັ້ນອາຄານ ໂດຍຜ່ານ ຟັງຊັນສຸ່ມ. ເນື່ອງຈາກວ່າມີໄຂ່ໜ່ວຍດຽວ, ສົມມຸດວ່າທ່ານບໍ່ ສາມາດດໍາເນີນການທົດລອງຕໍ່ໄປໄດ້ຖ້າມັນແຕກ.
- ຖ້າໄຂ່ບໍ່ແຕກ, ທ່ານສາມາດເຮັດການທົດລອງອີກຄັ້ງ. ດັ່ງນັ້ນ, ທົດ ສອບຈາກຊັ້ນທີ 1 ເຖິງຊັ້ນສູງສຸດ. ຊອກຫາຊັ້ນທີ່ນ້ອຍກວ່າຊັ້ນທີ່ເຮັດ ໃຫ້ໄຂ່ແຕກຄັ້ງທໍາອິດ.
- ຢ່າງໃດກໍຕາມ, ຖ້າຮອດຊັ້ນສູງສຸດແລ້ວໄຂ່ຍັງບໍ່ແຕກ, ໃຫ້ຖືວ່າຊັ້ນທີ່ ຍັງບໍ່ແຕກຄືຊັ້ນສູງສຸດຂອງອາຄານ ແລະ ຖ້າໄຂ່ແຕກຢູ່ຊັ້ນທີ 1 ໃຫ້ຖື ວ່າຊັ້ນທີ່ຍັງບໍ່ແຕກຄືຊັ້ນທີ 0.

2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່

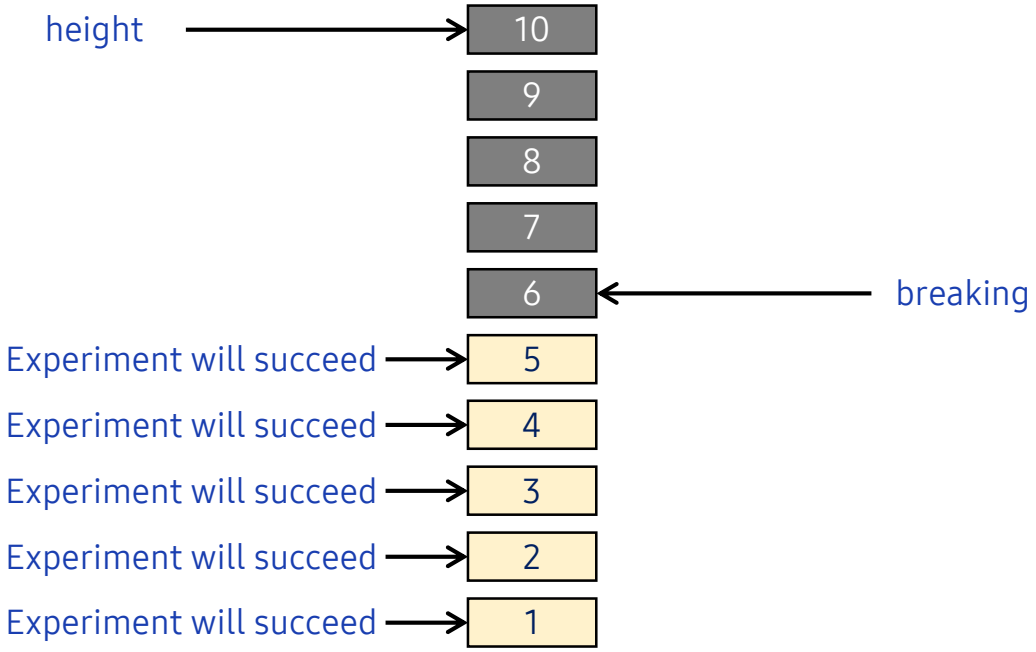
- ທົດລອງອາຄານທີ່ມີຄ່າ N=10.
- ສົມມຸດວ່າໄຂ່ບໍ່ແຕກເມື່ອຕົກຈາກຊັ້ນທີ 5 ຫຼື ຕ່ຳກວ່າ ແລະ ມັນແຕກເມື່ອຕົກຈາກຊັ້ນທີ 6 ຫຼື ສູງກວ່າ. ດັ່ງນັ້ນ, ໄຂ່ຈະບໍ່ແຕກເມື່ອເຮັດການທົດລອງຢູ່ຊັ້ນ 1.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່

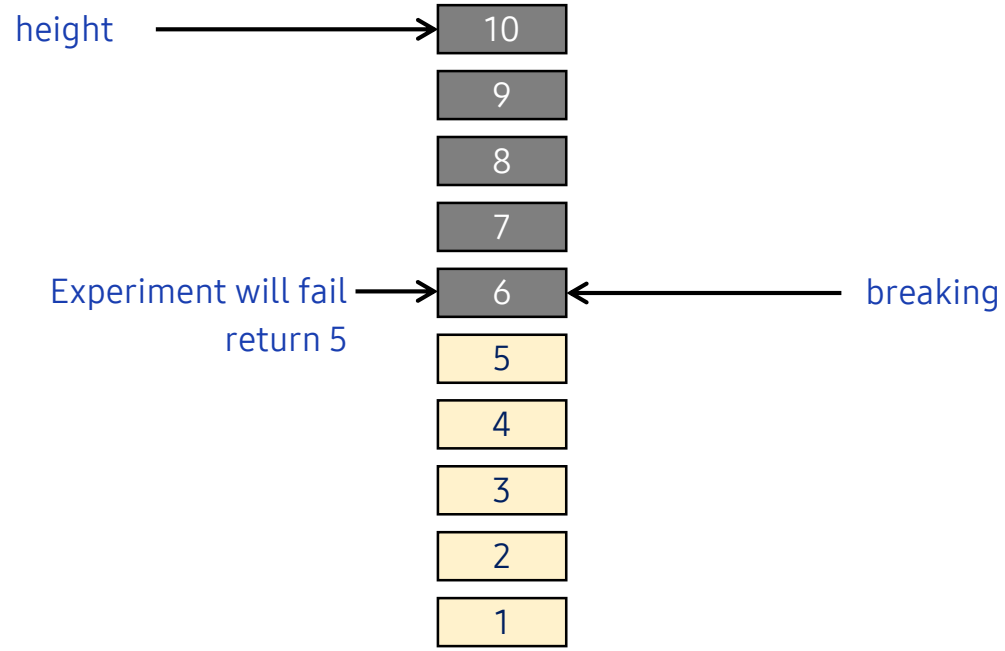
- ສືບຕໍ່ດຳເນີນການທົດລອງໂດຍການຂຶ້ນແຕ່ລະຊັ້ນ ແລະ ການທົດລອງຈະປະສົບຜົນສຳເລັດໂດຍທີ່ໄຂ່ບໍ່ແຕກ.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່

- ການທົດລອງຢູ່ໃນຊັ້ນທີ 6 ຈະເຮັດໃຫ້ໄຂ່ແຕກ.
- ດັ່ງນັ້ນ, ຊັ້ນທີ 5 ແມ່ນຊັ້ນສູງສຸດທີ່ໄຂ່ຍັງບໍ່ແຕກ.



3. Solution

3.1. ວິທີເຮັດວຽກຂອງການແກ້ໄຂບັນຫາ

ບັນຫາການປ່ອຍໄຂ່ເພື່ອຊອກຫາຄວາມສູງທີ່ໄຂ່ແຕກໂດຍການຮັບຄ່າຄວາມສຸງຂອງອາຄານ ' height' ເປັນຂໍ້ມູນປ້ອນເຂົ້າຈາກຜູ້ໃຊ້.

```
1 from random import randint
2
3 height = int(input("Input the number of floors: "))
4 breaking = randint(1, height)
5 floor = find_highest_safe_floor(height, breaking)
6 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Your egg will safe till the 94-th floor.
```

3. Solution

3.2. Final Code ຂອງບັນຫາການຕົກຂອງໄຂ່

```python
def do_experiment(floor, breaking):
    return floor >= breaking

def find_highest_safe_floor(height, breaking):
    for n in range(1, height + 1):
        if do_experiment(n, breaking):
            return n - 1
    return height
```

Key concept

1. ບັນຫາກ່ຽວກັບການຄົ້ນຫາ (Searching problem)

1.1. ປະເພດການຄົ້ນຫາມີສອງ

- ໃນການຂຽນໂປຣແກຣມຄອມພິວເຕີ, ຫນຶ່ງໃນຂະບວນການທີ່ເຮັດເປັນປະຈໍາທີ່ສຸດແມ່ນການຊອກຫາຂໍ້ມູນໃດໜຶ່ງ ທີ່ມີຢູ່ໃນໂຄງສ້າງຂໍ້ມູນໃດໜຶ່ງ.
- ຖ້າໂຄງສ້າງຂໍ້ມູນທີ່ກໍານົດບໍ່ມີກົດລະບຽບໃດໆ, ຈໍາເປັນຕ້ອງກວດເບິ່ງທັງຫມົດ. ສໍາລັບຕົວຢ່າງ, ພິຈາລະນາບັນຫາ ເພື່ອຊອກຫາທີ່ຢູ່ຂອງຄ່າສຸ່ມ x ໃດໜຶ່ງ ຈາກ list S ໃນຂະນະທີ່ N ເປັນຈໍານວນຖ້ວນຖືກກໍານົດໄວ້ແບບສຸ່ມ. ຈະເປັນແນວໃດຖ້າ x ບໍ່ມີຢູ່ໃນ list S? ທ່ານຈະບໍ່ສາມາດຊອກຫາທີ່ຢູ່ຂອງ x ໄດ້ຈົນກ່ວາການກວດສອບອົງປະກອບທັງຫມົດໃນ S.
- ຖ້າຂໍ້ມູນທີ່ມີກົດລະບຽບ, ບັນຫາການຄົ້ນຫາສາມາດແກ້ໄຂໄດ້ຢ່າງມີປະສິດທິພາບ. ຕົວຢ່າງ, ຖ້າໂຄງສ້າງຂໍ້ມູນທີ່ໄດ້ຮັບແມ່ນຂໍ້ມູນທີ່ມີບໍາດັບ, ແມ່ນນໍາໃຊ້ການຄົ້ນຫາແບບ binary search ເຊິ່ງຈະອະທິບາຍໃນພາຍຫຼັງ.

Sequential Search	vs.	Binary Search

1. ບັນຫາກ່ຽວກັບການຄົ້ນຫາ (Searching problem)

1.2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

- ການຄົ້ນຫາແບບຕາມລຳດັບ ໝາຍເຖິງຂັ້ນຕອນວິທີທີ່ເຮັດການສືບຄົ້ນໂດຍການກວດສອບທຸກລາຍການຂໍ້ມູນໄປຕາມລຳດັບ.
- ການຄົ້ນຫາແບບຕາມລຳດັບແມ່ນໃຊ້ສຳລັບຂໍ້ມູນແບບສຸ່ມ. ດັ່ງນັ້ນ, ເນື່ອງຈາກວ່າການຄົ້ນຫາແບບດັ່ງກ່າວແມ່ນເຮັດໄດ້ໂດຍບໍ່ມີຂໍ້ກຳນົດໃດໆ, ມັນຍັງຖືກເອີ້ນວ່າ 'ຊອກຫາເຂັມໃນກອງຫຍ້າ(‘finding a needle in a haystack’).

2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

ໃຫ້ພິຈາລະນາຂັ້ນຕອນວິທີການຄົ້ນຫາແບບຕາມລຳດັບທີ່ຊອກຫາຈຳນວນຖ້ວນໃດໜຶ່ງຢູ່ໃນ list ຂອງຈຳນວນຖ້ວນທີ່ກຳນົດໃຫ້.

Focus ຕໍ່ໄປນີ້ສະແດງໃຫ້ເຫັນບັນຫາຂັ້ນຕອນວິທີຄົ້ນຫາແບບຕາມລຳດັບ ແລະ ຂໍ້ມູນທີ່ປ້ອນເຂົ້າ ແລະ ຜົນໄດ້ຮັບຂອງມັນ.

- ບັນຫາ: ມີຈຳນວນຖ້ວນ x ຫຼືບໍ ຢູ່ໃນກຸ່ມຈຳນວນຖ້ວນ S ທີ່ໃຫ້ມາ?
- ຂໍ້ມູນປ້ອນເຂົ້າ: S ເປັນ list ຂອງຈຳນວນຖ້ວນ, x ເປັນຈຳນວນຖ້ວນແບບສຸ່ມທີ່ຕ້ອງການຊອກຫາ.
- ຜົນໄດ້ຮັບ: ທີ່ຢູ່ດັດສະນີຖ້າ x ມີຢູ່ໃນ list S ແລະ -1 ຖ້າມັນບໍ່ມີ.

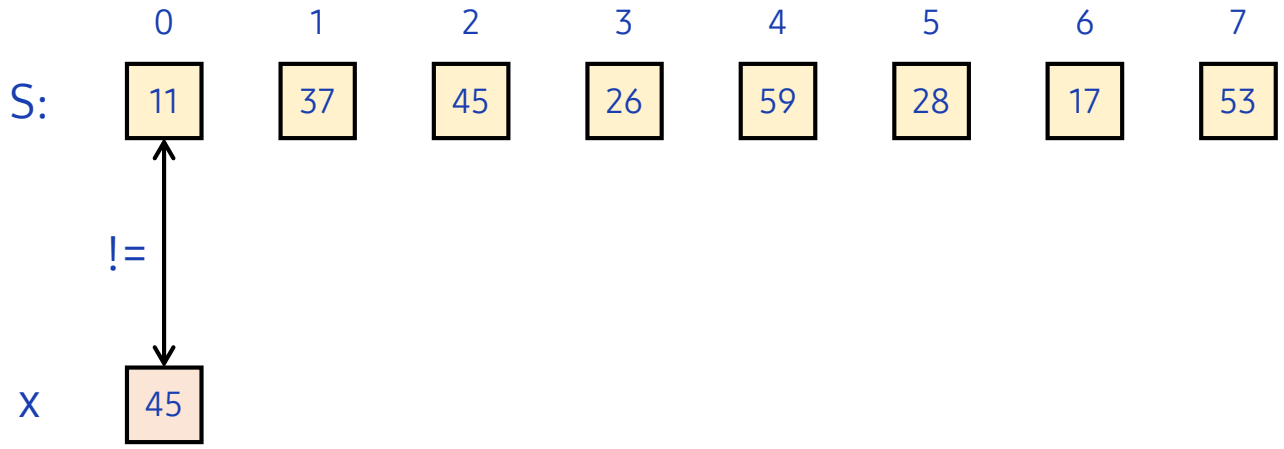
```
1 S = [11, 37, 45, 26, 59, 28, 17, 53]
2 x = 53
3 pos = seq_search(S, x)
4 print(f"In S, {x} is at position {pos}.")
```

```
In S, 53 is at position 7.
```

2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

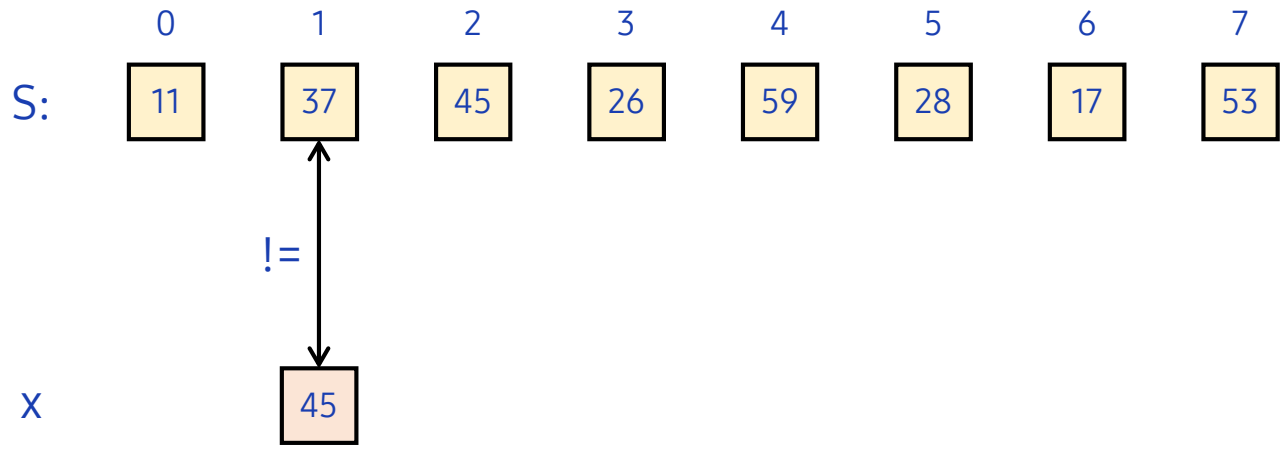
- ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບຕາມລຳດັບຈະປຽບທຽບຄ່າ x ທີ່ຕ້ອງການຊອກຫາກັບແຕ່ລະອົງປະກອບ ໃນ list S ທີ່ໃຫ້ມາ.
- ເມື່ອເລີ່ມຕົ້ນການຄົ້ນຫາແບບຕາມລຳດັບ, ມັນຈະສົ່ງຄືນດັດສະນີ 0 ຖ້າຄ່າເທົ່າກັນກັບອົງປະກອບທຳອິດ ແລະ ຖ້າບໍ່ແມ່ນ, ມັນຈະປຽບທຽບກັບອົງປະກອບຕໍ່ໄປ.



2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

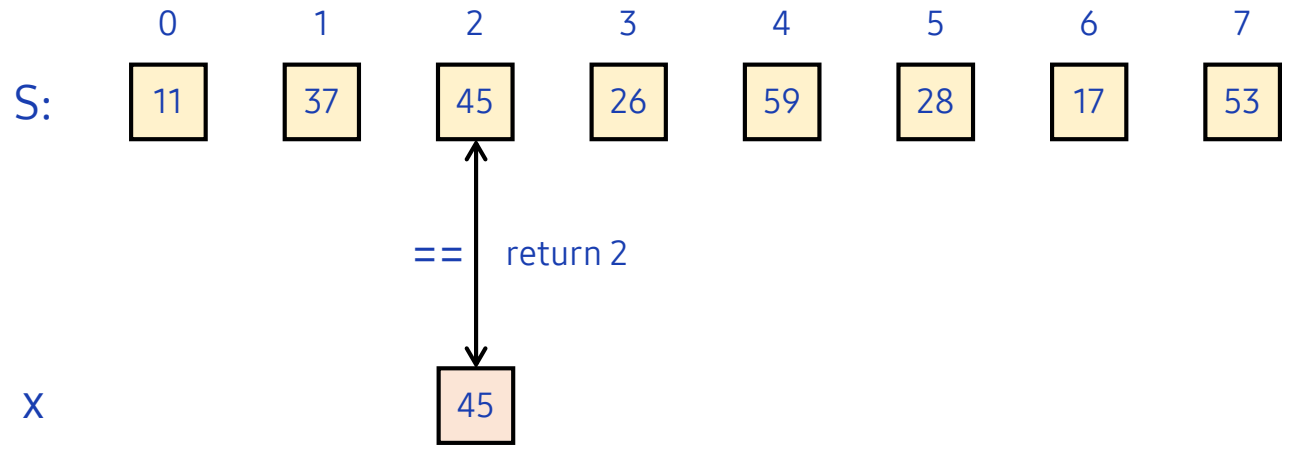
ຖ້າຄ່າບໍ່ເທົ່າກັບອົງປະກອບຕໍ່ໄປ S[1], ມັນຈະໄປປຽບທຽບກັບອັນຕໍ່ໄປອີກ.



2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

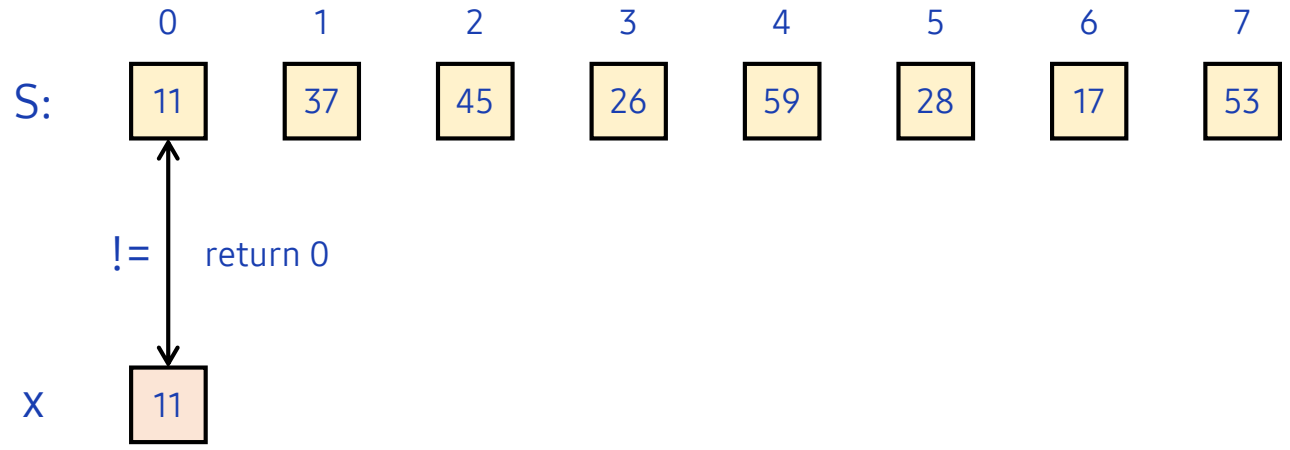
ຄ່າ x ແມ່ນເທົ່າກັນກັບອົງປະກອບ S[2], ດັ່ງນັ້ນມັນສົ່ງຄືນຄ່າດັດສະນີ 2 ແລ້ວສຳເລັດການຄົ້ນຫາ



2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

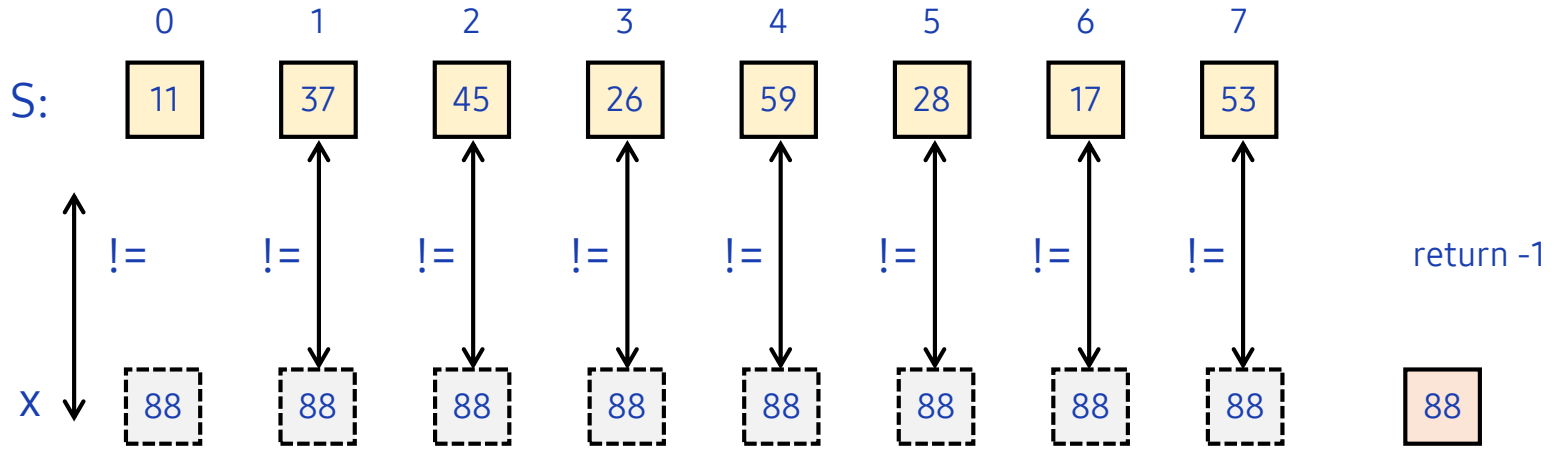
ໃນການຄົ້ນຫາແບບຕາມລຳດັບ, ຖ້າອົງປະກອບທີ່ຈະຊອກຫາແມ່ນ S[0] ເຊິ່ງເປັນອົງປະກອບທຳອິດ, ມັນຈະດຳເນີນການປຽບທຽບພຽງແຕ່ຄັ້ງດຽວກ່ຳສຳເລັດການຄົ້ນຫາແລ້ວ, ເຊິ່ງຫມາຍເຖິງ 'ກໍລະນີທີ່ດີທີ່ສຸດ(Best-case)'.



2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມອັນດັບ

- ໃນການຄົ້ນຫາແບບຕາມລຳດັບ, ຖ້າອົງປະກອບທີ່ຈະຊອກຫາບໍ່ມີຢູ່ໃນ S, ການປຽບທຽບຈະຕ້ອງເຮັດກັບອົງປະກອບທັງຫມົດຂອງ S.
- ຖ້າຫາກວ່າມີ n ອົງປະກອບໃນ S, ການປຽບທຽບຈະເຮັດທັງຫມົດ n ຄັ້ງ. ໃນກໍລະນີດັ່ງກ່າວນັ້ນ, ມັນຖືກເອີ້ນວ່າ 'ກໍລະນີທີ່ບໍ່ດີທີ່ສຸດ(Worst-case).'



2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

- ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບຕາມລຳດັບສາມາດຖືກປະຕິບັດດັ່ງຕໍ່ໄປນີ້.
- ຮັບ ລາຍການຂໍ້ມູນ ' nums' ແລະ ຄ່າທີ່ຕ້ອງການ x ເປັນຂໍ້ມູນທີ່ປ້ອນເຂົ້າເພື່ອສືບຕໍ່ການຄົ້ນຫາຕາມລຳດັບຂອງອົງປະກອບໃນ nums.

```
1  def seq_search(nums, x):
2      for i in range(len(nums)):
3          if x == nums[i]:
4              return i
5      return -1
```

Line 2-5

- ແຖວທີ 2 ແມ່ນ for-loop ສຳລັບການຄົ້ນຫາແບບຕາມລຳດັບຂອງອົງປະກອບໃນ nums.
- ແຖວທີ 3 ແລະ 4 ສົ່ງຄືນຄ່າ 'i' ຖ້າ nums[i] ແລະ x ຄືກັນ.
- ແຖວທີ 5 ສົ່ງຄືນຄ່າ -1 ກໍລະນີຢູ່ໃນ nums ບໍ່ມີ x ທີ່ເຮົາຕ້ອງການ.

2. ການຄົ້ນຫາແບບຕາມລຳດັບ(Sequential search)

2.1. ຂັ້ນຕອນວິທີສຳຫຼັບການຄົ້ນຫາແບບຕາມລຳດັບ

ຟັງຊັນ seq_search() ທີ່ຖືກສ້າງກ່ອນໜ້ານີ້ແມ່ນຖືກນຳໃຊ້ເພື່ອຊອກຫາອົງປະກອບ ຂອງ list ດັ່ງຕໍ່ໄປນີ້.

```
1 S = [11, 37, 45, 26, 59, 28, 17, 53]
2 x = 11
3 pos = seq_search(S, x)
4 print(f"In S, {x} is at position {pos}.")
```

```
In S, 11 is at position 0.
```

```
1 S = [11, 37, 45, 26, 59, 28, 17, 53]
2 x = 77
3 pos = seq_search(S, x)
4 print(f"In S, {x} is at position {pos}.")
```

```
In S, 77 is at position -1.
```

One More Step

- ເວລາຫຼາຍສຸດທີ່ໃຊ້ຄົ້ນຫາຂອງການຄົ້ນຫາແບບຕາມລຳດັບເທົ່າໃດ? ອິເຄາະດ້ວຍຕຳລາ Big-O.
- ໃນເບື້ອງຕົ້ນ, ກໍລະນີທີ່ດີທີ່ສຸດ, ການຄົ້ນຫາສຳເລັດໂດຍການປຽບທຽບພຽງຄັ້ງດຽວ, ດັ່ງນັ້ນເວລາທີ່ໃຊ້ແມ່ນ O(1). ນອກຈາກນີ້, ຖ້າມີການປຽບທຽບ n ຄັ້ງ ໃນກໍລະນີທີ່ບໍ່ດີທີ່ສຸດ ມັນຈະໃຊ້ເວລາ O(n).
- ເວລາໂດຍສະເລ່ຍທີ່ໃຊ້ໃນການຄົ້ນຫາແບບຕາມລຳດັບຈະອິເຄາະແນວໃດ? ສົມມຸດວ່າ S ມີອົງປະກອບທັງໝົດ N ຕົວ ເມື່ອຄົ້ນຫາທຸກໆອົງປະກອບຂອງ S ອັນຕໍ່ອັນ, ຄຳນວນເບິ່ງຈຳນວນການປຽບທຽບ (==) ທີ່ໄດ້ເຮັດ.
- ກຳນົດເປັນ 1 ສຳລັບອົງປະກອບທຳອິດ ແລະ 2 ສຳລັບອົງປະກອບທີສອງ, ແລະ ອື່ນໆ. ສຳລັບອົງປະກອບ N ຕົວ, ການປຽບທຽບຈະເຮັດທັງໝົດ N ຄັ້ງ. ດັ່ງນັ້ນ, ຈຳນວນການປຽບທຽບທັງໝົດທີ່ຕ້ອງການເພື່ອຊອກຫາທຸກອົງປະກອບແຕ່ລະອັນແມ່ນ N(N+1)/2.

$$1 + 2 + \cdots + N = \frac{N(N+1)}{2}$$

- ດັ່ງນັ້ນ, ເວລາຄົ້ນຫາໂດຍສະເລ່ຍ T(n) ກັບອົງປະກອບທັງຫມົດແມ່ນ O(n), ເຊິ່ງເປັນເວລາຄົ້ນຫາແບບເສັ້ນຊື່ (linear search).

$$T(N) = \frac{N(N+1)}{2} \div N = \frac{N+1}{2} \in O(n)$$

3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

ພິຈາລະນາຂັ້ນຕອນວິທີທີ່ຊອກຫາລະຫັດທີ່ໃຫຍ່ທີ່ສຸດໃນ list ຂອງຈຳນວນຖ້ວນທີ່ໃຫ້ມາ.

Focus ຕໍ່ໄປນີ້ແມ່ນ ຂໍ້ມູນປ້ອນເຂົ້າ ແລະ ຜົນໄດ້ຮັບຂອງບັນຫາກ່ຽວກັບຂັ້ນຕອນວິທີເພື່ອຊອກຫາລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ.

- ບັນຫາ: ລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຢູ່ໃນ S ຊຶ່ງເປັນກຸ່ມຂອງຈຳນວນຖ້ວນທີ່ໃຫ້ມາຢູ່ບ່ອນໃດ?
- ຂໍ້ມູນປ້ອນເຂົ້າ: S ເປັນລາຍການຈຳນວນຖ້ວນ
- ຜົນໄດ້ຮັບ: ດັດຊະນີທີ່ໃຫຍ່ທີ່ສຸດໃນບັນດາອົງປະກອບພາຍໃນ S

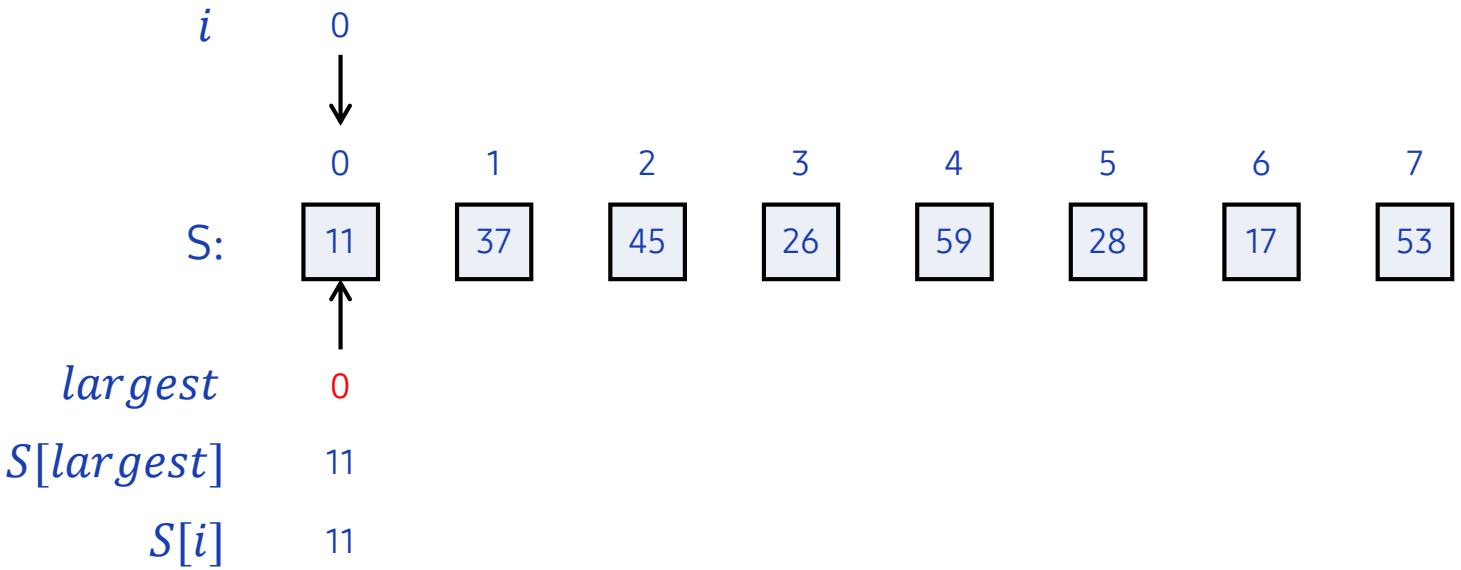
```
1  nums = [11, 37, 45, 26, 59, 28, 17, 53]
2  pos = find_largest(nums)
3  print(f"The largest is {nums[pos]} at {pos}")
```

```
[11, 37, 45, 26, 59, 28, 17, 53]
The largest is 59 at 4
```

3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

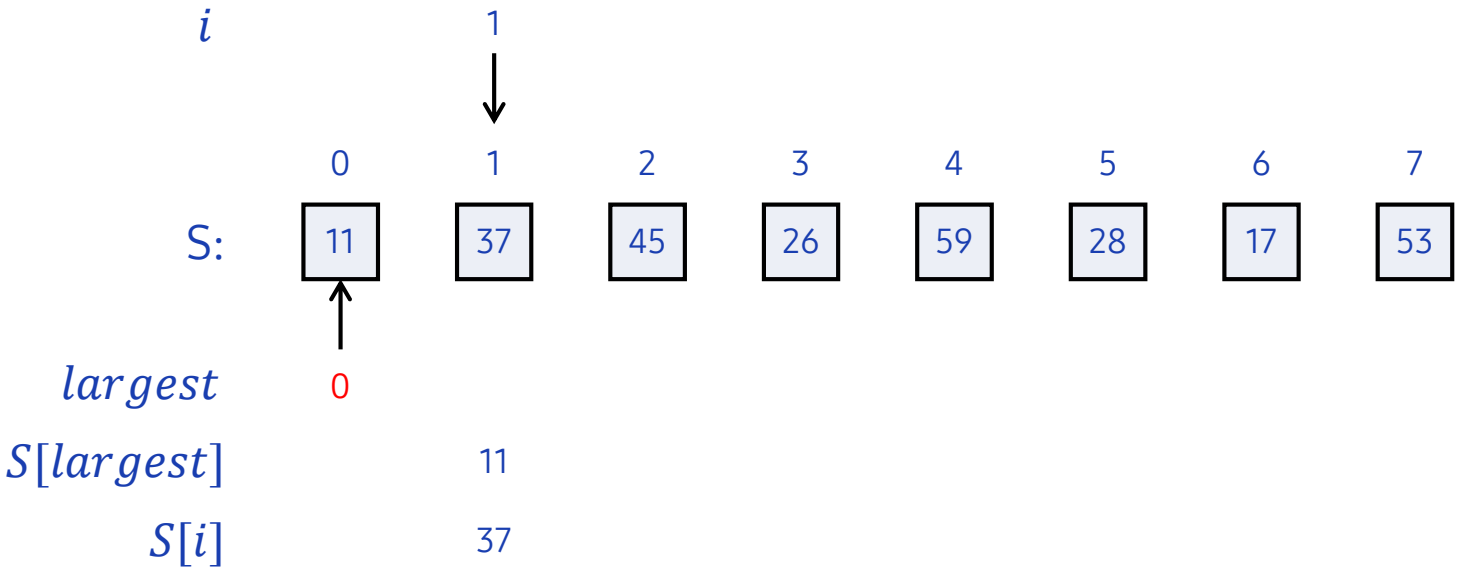
ຂັ້ນຕອນວິທີການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດສາມາດຖືກສ້າງໃນລັກສະນະທີ່ຄ້າຍຄືກັນກັບວິທີການຄົ້ນຫາແບບຕາມລຳດັບ. ທຳອິດ, ສົມມຸດວ່າທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດແມ່ນອົງປະກອບທຳອິດຂອງ S.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

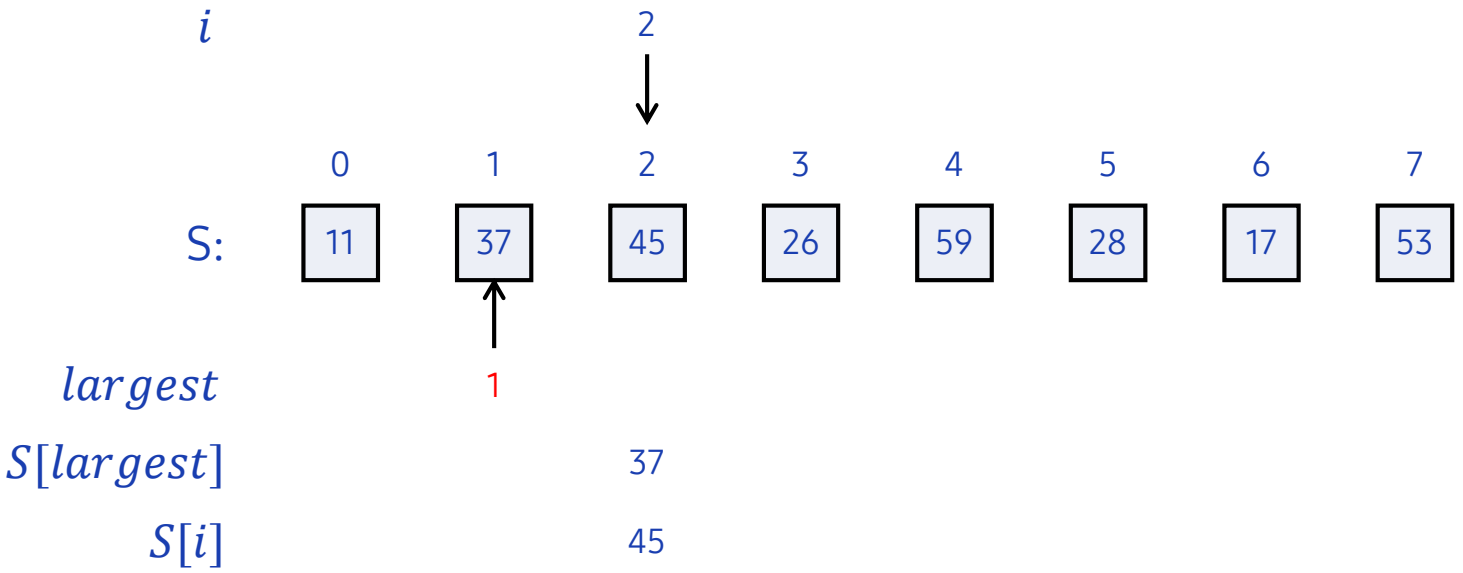
- ເມື່ອປຽບທຽບ S[1] ກັບ S[largest] ໃນປັດຈຸບັນ, ເພາະວ່າ S[1] ໃຫຍ່ກວ່າ S[0], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຄວນຈະຖືກປ່ຽນເປັນ 1.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

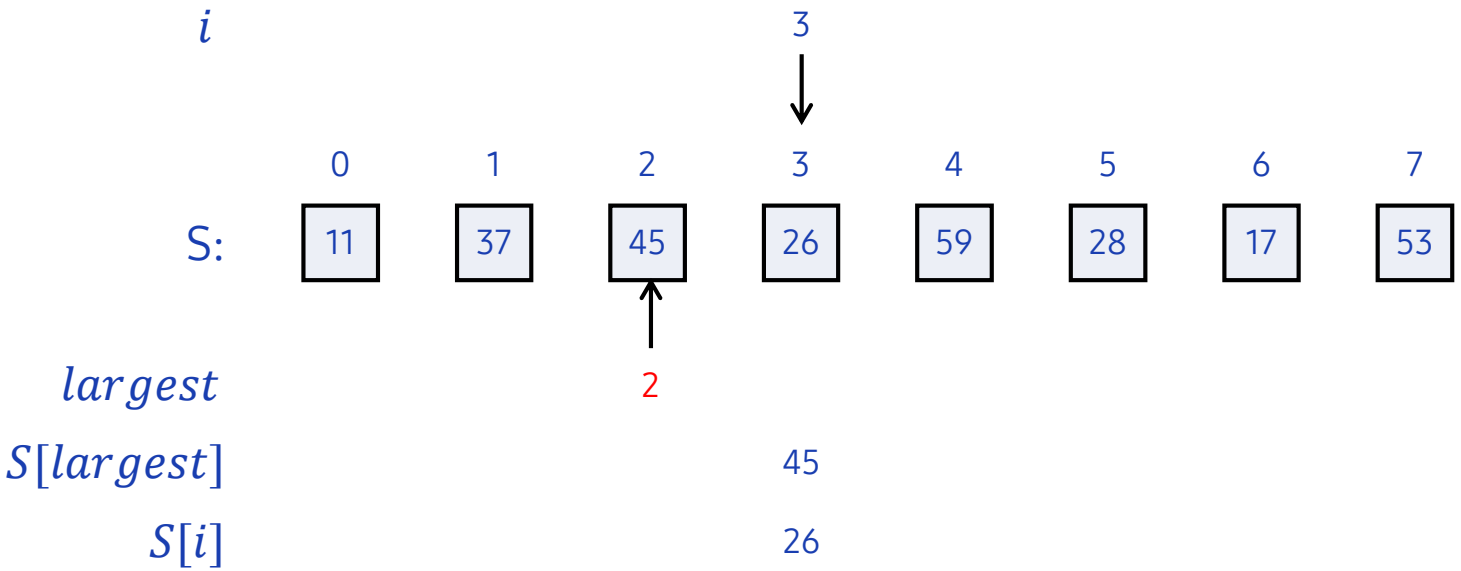
- ເມື່ອປຽບທຽບ S[2] ກັບ S[largest] ໃນປັດຈຸບັນ, ເພາະວ່າ S[2] ໃຫຍ່ກວ່າ S[1], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຄວນຈະຖືກປ່ຽນເປັນ 2.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

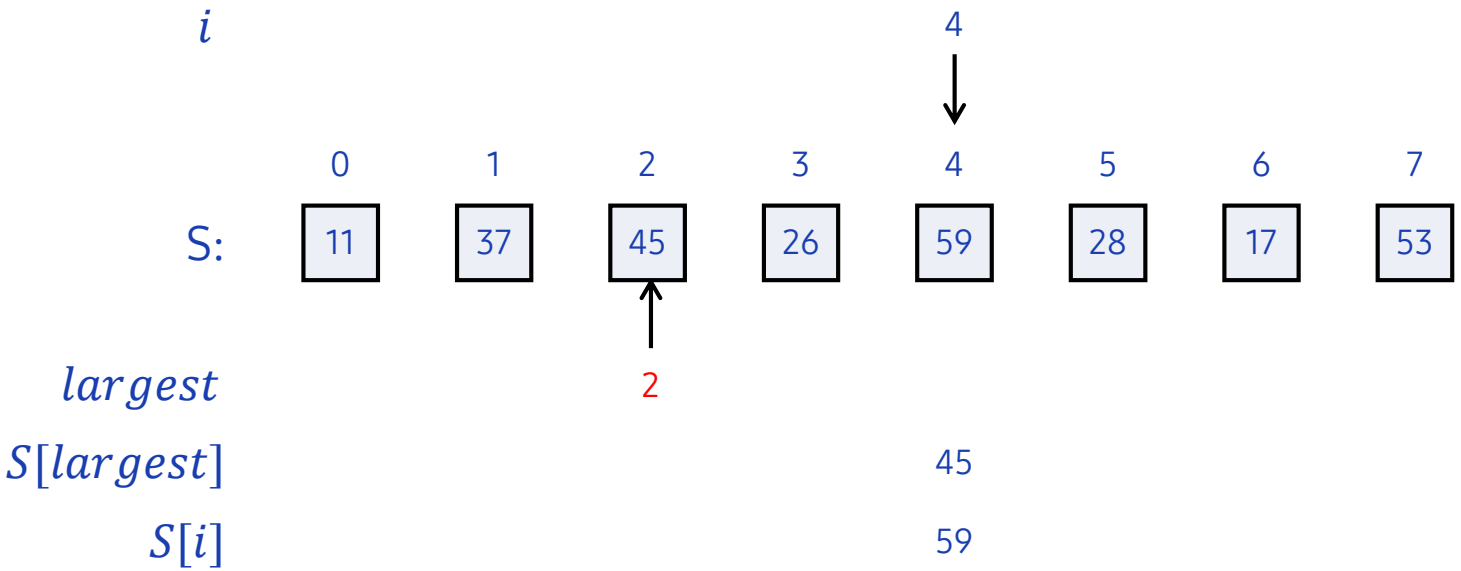
- ເມື່ອປຽບທຽບ S[3] ກັບ S[largest] ໃນປັດຈຸບັນ, ເພາະວ່າ S[3] ມີຂະໜາດນ້ອຍກວ່າ S[2], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດບໍ່ປ່ຽນແປງ.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

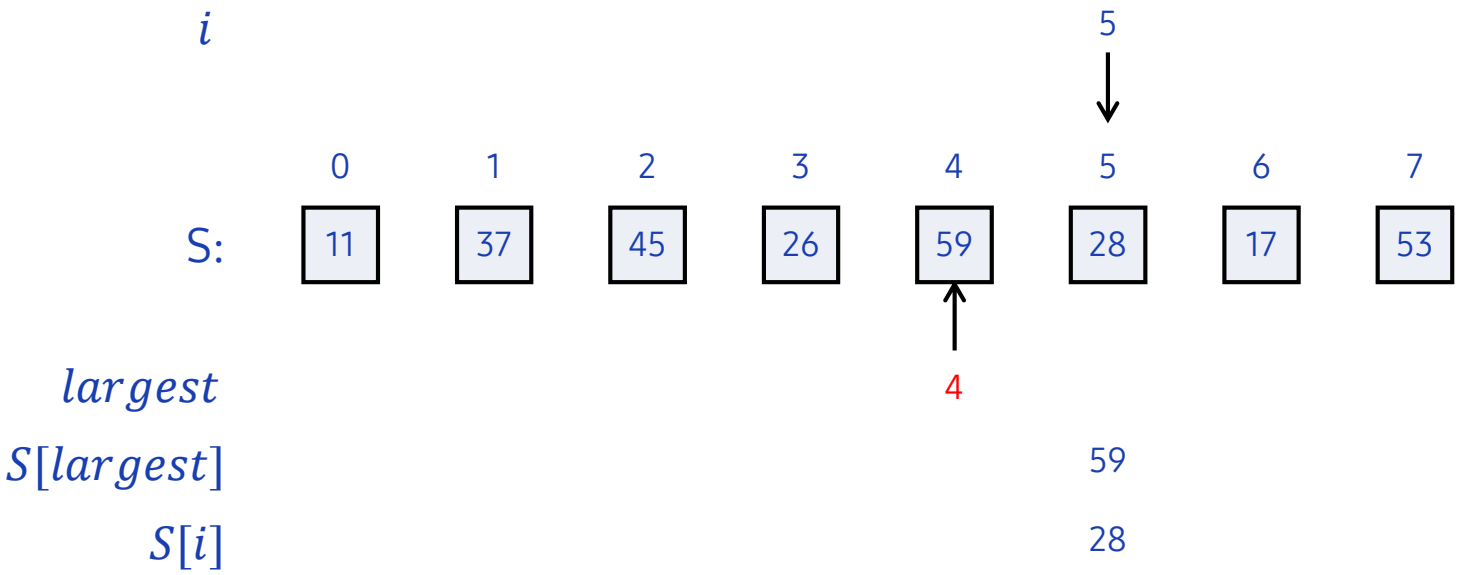
- ເມື່ອປຽບທຽບ S[4] ກັບ S[largest] ໃນປັດຈຸບັນ, ເພາະວ່າ S[4] ໃຫຍ່ກວ່າ S[2], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຄວນຈະຖືກປ່ຽນເປັນ 4.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

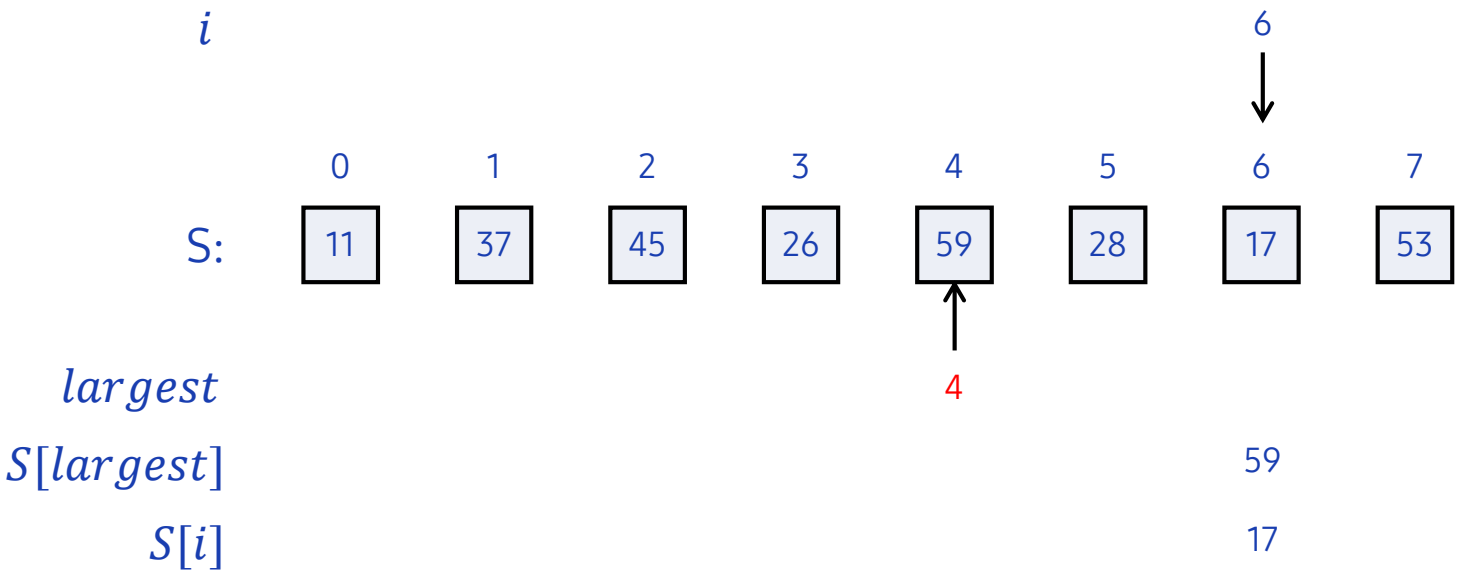
ຖ້າບໍ່ມີຄ່າທີ່ໃຫຍ່ກວ່າ S [largest], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຍັງຄົງຢູ່ຄືກັນ.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

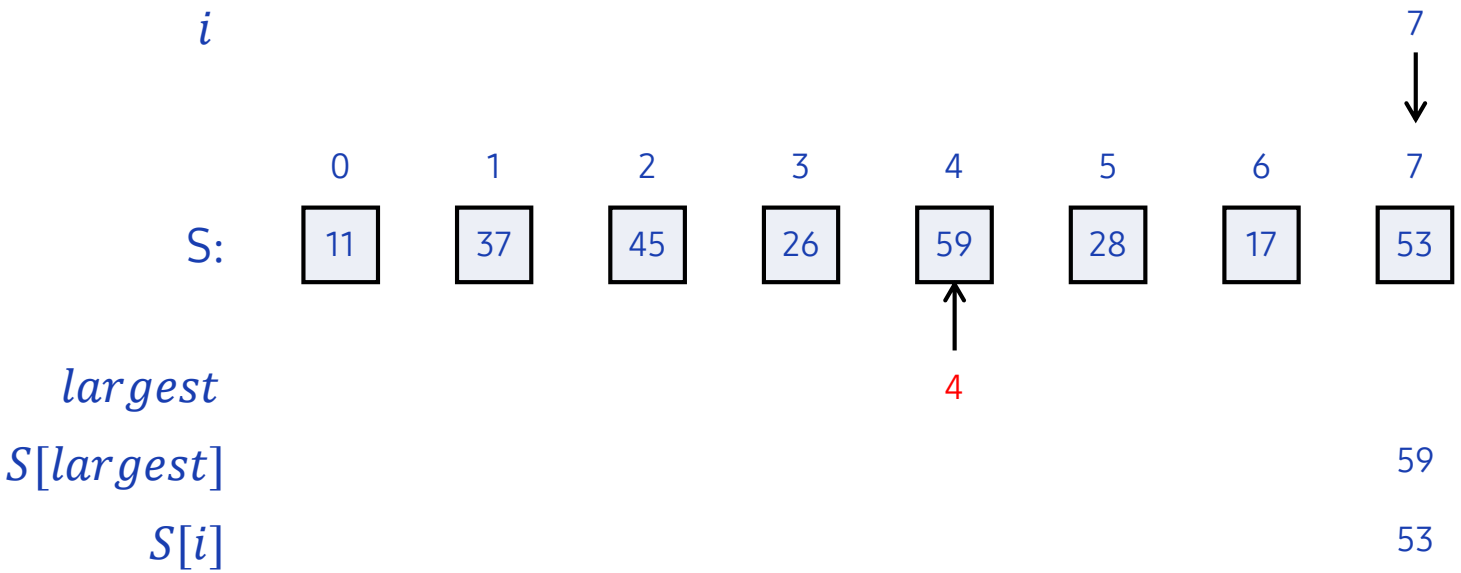
ຖ້າບໍ່ມີຄ່າທີ່ໃຫຍ່ກວ່າ S [largest], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຍັງຄົງຢູ່ຄືກັນ.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

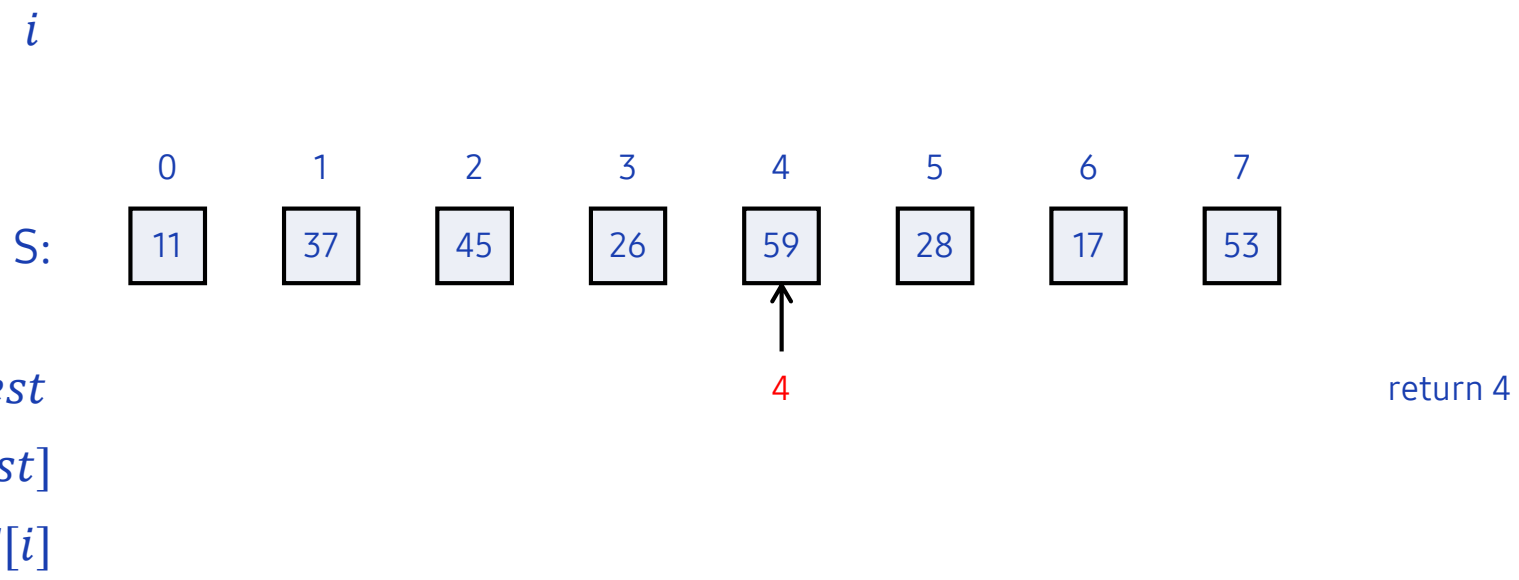
ຖ້າບໍ່ມີຄ່າທີ່ໃຫຍ່ກວ່າ S [largest], ທີ່ຢູ່ຂອງລະຫັດທີ່ໃຫຍ່ທີ່ສຸດຍັງຄົງຢູ່ຄືກັນ.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.1. ຂັ້ນຕອນວິທີສຳຫຼັງການຄົ້ນຫາຄ່າລະຫັດທີ່ໃຫຍ່ທີ່ສຸດ

- ຄ່າທີ່ໃຫຍ່ທີ່ສຸດແມ່ນ max ຊຶ່ງເປັນດັດຊະນີຫຼັງຈາກສຳເລັດການຄົ້ນຫາອົງປະກອບທັງຫມົດ, ສະນັ້ນມັນສົ່ງຄືນຄ່າທີ່ໃຫຍ່ທີ່ສຸດ.



3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.2. ສ້າງ ແລະ ຂຽນຊອດໂຄດ

ນຳໃຊ້ຍຸດທະສາດທີ່ໃຊ້ໃນເມື່ອກ່ອນເພື່ອສ້າງຟັງຊັນ find_largest() ດັ່ງຕໍ່ໄປນີ້.

```
1  def find_largest(nums):
2      largest = 0
3      for i in range(1, len(nums)):
4          if nums[largest] < nums[i]:
5              largest = i
6      return largest
```

Line 2-5

- ເລີ່ມຕົ້ນດັດສະນີຂອງອົງປະກອບທຳອິດໃນ nums list ໃຫ້ເປັນຄ່າທີ່ໃຫຍ່ທີ່ສຸດ.
- ປຽບທຽບຄ່າໃນ nums[largest] ປັດຈຸບັນ ແລະ ຄ່າ nums[i] ຈາກອົງປະກອບທີ 2 ຫາອົງປະກອບສຸດທ້າຍຂອງ nums list.
- ຖ້າຄ່າ 'i' ຫຼາຍກວ່າຄ່າທີ່ໃຫຍ່ທີ່ສຸດ, ປັບປຸງຄ່າທີ່ໃຫຍ່ທີ່ສຸດເປັນ 'i.'

3. ການຄົ້ນຫາຄ່າທີ່ໃຫຍ່ທີ່ສຸດ

3.2. ສ້າງ ແລະ ຂຽນຊອດໂຄດ

```
1  def find_largest(nums):
2      largest = 0
3      for i in range(1, len(nums)):
4          if nums[largest] < nums[i]:
5              largest = i
6      return largest
```

Line 6

- ຄ່າທີ່ໃຫຍ່ທີ່ສຸດຫຼັງຈາກສຳເລັດການຄົ້ນຫາແມ່ນຄ່າດັດສະນີຂອງອົງປະກອບທີ່ໃຫຍ່ທີ່ສຸດໃນ nums.

Let’s code

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

- ຂຽນ algorithm ເພື່ອຊອກຫາຊັ້ນສູງທີ່ສຸດທີ່ໄຂ່ຍັງບໍ່ແຕກໃນເວລາທີ່ປ່ອຍລົງ ໃນຄວາມສູງຂອງອາຄານ ທັງໝົດທີ່ໃຫ້.
- ເງື່ອນໄຂການສ້າງ algorithm ມີດັ່ງນີ້.

1. 'ການແຕກ,' ເຊິ່ງເປັນຊັ້ນຕ່ຳສຸດທີ່ໄຂ່ແຕກ, ຖືກເລືອກແບບສຸ່ມລະຫວ່າງ 1 ແລະຄວາມສູງສຸງສຸດ.
2. ການທົດລອງປ່ອຍໄຂ່ແມ່ນສະແດງອອກເປັນ do_experiment(floor).
   - ສົ່ງຄືນຄ່າ True ຖ້າໄຂ່ແຕກເມື່ອປ່ອຍລົງຈາກພື້ນ.
   - ຖ້າບໍ່ແມ່ນດັ່ງນັ້ນ, ສົ່ງຄືນຄ່າ False.
3. ແນວໃດກໍ່ຕາມ, ໂປຣແກຣມຕ້ອງຖືກປິດເມື່ອຟັງຊັນ do_experiment() ເປັນຄ່າ True.

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

ຮັບຂໍ້ມູນປ້ອນເຂົ້າຈາກຜູ້ໃຊ້ກ່ຽວກັບຄວາມສູງອາຄານ 'height' ໂດຍຜ່ານຟັງຊັນ input().

```
1 from random import randint
2
3 height = int(input("Input the number of floors: "))
4 breaking = randint(1, height)
5 floor = find_highest_safe_floor(height, breaking)
6 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Your egg will safe till the 94-th floor.
```

Line 3

- ປ່ຽນປະເພດການປ້ອນຂໍ້ມູນທີ່ໄດ້ຮັບໂດຍຟັງຊັນ input() ເປັນຟັງຊັນ int() ແລະເອົາໄປເກັບໄວ້ໃນຕົວປ່ຽນ ' height.'

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

ສ້າງຕົວເລກສຸ່ມລະຫວ່າງ 1 ແລະ ຄວາມສູງ ແລະ ເອົາຄ່າໄປເກັບໄວ້ໃນຕົວປ່ຽນ 'breaking.'

```
1 from random import randint
2
3 height = int(input("Input the number of floors: "))
4 breaking = randint(1, height)
5 floor = find_highest_safe_floor(height, breaking)
6 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Your egg will safe till the 94-th floor.
```

Line 1, 4

- ສຳລັບການສ້າງຈຳນວນຖ້ວນແບບສຸ່ມໃນຂອບເຂດໃດໜຶ່ງ, ໃຫ້ໃຊ້ຟັງຊັນ randint() ໃນໂມດູນແບບສຸ່ມ.
- ຟັງຊັນ randint(a, b) ສົ່ງຄືນຄ່າຈຳນວນຖ້ວນສຸ່ມໃນຊ່ວງຈາກ a ແລະ b (ເທົ່າກັບ ຫຼືໃຫຍ່ກວ່າ 'a' ແລະ ເທົ່າກັບ ຫຼື ນ້ອຍກວ່າ 'b').

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

ສ້າງຟັງຊັນສົ່ງຄືນຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກ.

```
1  from random import randint
2
3  height = int(input("Input the number of floors: "))
4  breaking = randint(1, height)
5  floor = find_highest_safe_floor(height, breaking)
6  print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Your egg will safe till the 94-th floor.
```

Line 5-6

- ຟັງຊັນ find_highest_safe_floor() ສົ່ງຄືນຄ່າຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກ.
- ພິມອອກໃນຮູບແບບຂໍ້ຄວາມໂດຍໃຊ້ f-string. f-string ເພີ່ມ 'f' ຢູ່ທາງໜ້າຂອງຂໍ້ຄວາມ, ແລະ ໃຊ້ {} ໃສ່ກວາມຕົວປ່ຽນຢູ່ພາຍໃນຂໍ້ຄວາມ.

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

ສ້າງຟັງຊັນ do_experiment() ສຳລັບການທົດລອງການປ່ອຍໄຂ່.

```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor(height, breaking):
5      for n in range(1, height + 1):
6          if do_experiment(n, breaking):
7              return n - 1
8      return height
```

Line 1-2

- ຕົວປ່ຽນ ' breaking' ແມ່ນຊັ້ນຕ່ຳສຸດທີ່ໄຂ່ແຕກ.
- ດັ່ງນັ້ນ, ຖ້າຄ່າຄວາມສູງພື້ນເປັນພາລາມິເຕີ ຂໍ້ມູນປ້ອນເຂົ້າມີຄ່າເທົ່າກັນ ຫຼື ໃຫຍ່ກວ່າຄ່າຊັ້ນທີ່ໄຂ່ແຕກ, ມັນສົ່ງຄືນຄ່າເປັນ True ແລະ ຖ້າບໍ່ແມ່ນດັ່ງນັ້ນ, ມັນຈະສົ່ງຄືນຄ່າ False.

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

ຊອກຫາຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກໃນການທົດລອງປ່ອຍໄຂ່ລົງໂດຍໃຊ້ຂໍ້ມູນ ' height' ແລະ ' breaking'.

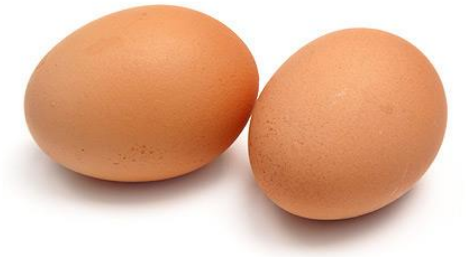
```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor(height, breaking):
5      for n in range(1, height + 1):
6          if do_experiment(n, breaking):
7              return n - 1
8      return height
```

Line 4-8

- ເຮັດການທົດລອງປ່ອຍໄຂ່ຄືນໃໝ່ຈາກຊັ້ນທີ 1 ຫາຊັ້ນ ' height' ແລະ ຖ້າໄຂ່ແຕກ, ໃຫ້ສົ່ງຄືນ ຄ່າຊັ້ນ n-1.
- ຖ້າທຳການທົດລອງທຸກຊັ້ນແລ້ວໄຂ່ຍັງບໍ່ແຕກ, ໃຫ້ສົ່ງຄືນຄ່າຊັ້ນສູງສຸດ ' height'.

One More Step

- ເວລາສຸງສຸດທີ່ໃຊ້ໃນການແກ້ໄຂບັນຫາການຕົກໄຂ່ແມ່ນເທົ່າໃດ?
- ຖ້າມີໄຂ່ໜ່ວຍດຽວ, ຄ້າຍຄືກັນກັບການຄົ້ນຫາແບບຕາມລຳດັບ, ກໍລະນີທີ່ດີທີ່ສຸດແມ່ນ O(1) ໃນຂະນະທີ່ກໍລະນີທີ່ບໍ່ດີທີ່ສຸດແມ່ນ O(n). ເວລາກໍລະນີໂດຍສະເລ່ຍຈະເປັນ O(n).
- ຈະເປັນແນວໃດ ຖ້າມີຈຳນວນໄຂ່ທີ່ບໍ່ຈຳກັດ ຖືວ່າການແຕກໄຂ່ໃນລະຫວ່າງການທົດລອງແມ່ນບໍ່ສຳຄັນ?
- ເມື່ອມີຈຳນວນໄຂ່ທີ່ບໍ່ຈຳກັດ, ມັນເປັນໄປໄດ້ທີ່ຈະໃຊ້ການຄົ້ນຫາແບບ binary search ເຊິ່ງຈະກ່າວເຖິງໃນລາຍລະອຽດຕໍ່ໄປ. ເວລາກໍລະນີໂດຍສະເລ່ຍຈະເປັນ O(logn).
- ຖ້າມີໄຂ່ 2 ໜ່ວຍ ຈະຕ້ອງເຮັດການທົດລອງຈັກເທື່ອຢ່າງໜ້ອຍ?
- ບັນຫານີ້ເອີ້ນວ່າບັນຫາ two-eggs ແລະ ຢ່າງໜ້ອຍຈະໄດ້ເຮັດການທົດລອງ 14 ຄັ້ງໃນອາຄານຊັ້ນ 100 ເປັນກໍລະນີທີ່ບໍ່ດີທີ່ສຸດ.



Pop quiz

Quiz. #1

ຄາດຄະເນຜົນໄດ້ຮັບຂອງ algorithm ຕໍ່ໄປນີ້. ວິເຄາະ code ແລະ ອະທິບາຍພຶດຕິກຳຂອງ algorithm ນີ້.

```python
def find_two(nums):
    x = y = 0
    for i in range(1, len(nums)):
        if nums[x] < nums[i]:
            x = i
        elif nums[y] > nums[i]:
            y = i
    return x, y
```

```python
nums = [11, 37, 45, 26, 59, 28, 17, 53]
i, j = find_two(nums)
print(nums[i], nums[j])
```

Quiz. #2

ຈຳນວນການປຽບທຽບທີ່ໄດ້ເຮັດໃນຟັງຊັນ find_two() ທີ່ຖືກສ້າງໃນ quiz #1 ມີຫຼາຍເທົ່າໃດ?

Pair programming

Pair Programming Practice

ແນວທາງ, ກົນໄກ ແລະ ແຜນສຸກເສີນ

ການກະກຽມການສ້າງໂປຣແກຣມຮ່ວມກັນເປັນຄູ່ກ່ຽວຂ້ອງກັບການໃຫ້ຄຳແນະນຳແລະກົນໄກເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄູ່ຢ່າງຖືກຕ້ອງແລະ ໃຫ້ເຂົາເຈົ້າເຮັດວຽກເປັນຄູ່. ຕົວຢ່າງ, ນັກຮຽນຄວນປ່ຽນ "ເຮັດ." ການກະກຽມທີ່ມີປະສິດຕິຜົນຕ້ອງໃຫ້ມີແຜນການສຸກເສີນໃນກໍລະນີທີ່ຄູ່ຮ່ວມງານຫນຶ່ງບໍ່ຢູ່ຫຼືຕັດສິນໃຈທີ່ຈະບໍ່ເຂົ້າຮ່ວມດ້ວຍເຫດຜົນໃດຫນຶ່ງ ຫຼືດ້ວຍເຫດຜົນອື່ນ. ໃນກໍລະນີເຫຼົ່ານີ້, ມັນເປັນສິ່ງສໍາຄັນທີ່ຈະເຮັດໃຫ້ມັນຊັດເຈນວ່ານັກຮຽນທີ່ມີປະຕິບັດໜ້າທີ່ຢ່າງຫ້າວຫັນຈະບໍ່ຖືກລົງໂທດ ຍ້ອນວ່າການຈັບຄູ່ບໍ່ໄດ້ຜົນດີ.

ການຈັບຄູ່ທີ່ຄ້າຍຄືກັນ, ບໍ່ຈຳເປັນຕ້ອງເທົ່າທຽມກັນ, ຄວາມສາມາດເປັນຄູ່ຮ່ວມງານ

ການຂຽນໂປຣແກຣມຄູ່ຈະມີປະສິດທິພາບເມື່ອນັກຮຽນຕັ້ງໃຈຮ່ວມກັນເຮັດວຽກ, ຊຶ່ງວ່າບໍ່ຈຳເປັນຕ້ອງມີຄວາມຮູ້ເທົ່າທຽມກັນ, ແຕ່ຕ້ອງມີຄວາມສາມາດເຮັດວຽກເປັນຄູ່ຮ່ວມງານ. ການຈັບຄູ່ນັກຮຽນທີ່ບໍ່ສາມາດເຂົ້າກັນໄດ້ມັກຈະເຮັດໃຫ້ການມີສ່ວນຮ່ວມທີ່ບໍ່ສົມດຸນກັນ. ຄູສອນຕ້ອງເນັ້ນຫນັກວ່າການຂຽນໂປຣແກຣມຄູ່ບໍ່ແມ່ນຍຸດທະສາດ “divide-and-conque”, ແຕ່ຈະເປັນຄວາມພະຍາຍາມຮ່ວມມືເຮັດວຽກທີ່ແທ້ຈິງໃນທຸກໆດ້ານສໍາລັບໂຄງການທັງຫມົດ. ຄູຄວນຫຼີກເວັ້ນການຈັບຄູ່ນັກຮຽນທີ່ອ່ອນຫຼາຍກັບນັກຮຽນທີ່ເກັ່ງຫຼາຍ.

ກະຕຸ້ນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈູງໃຈພິເສດ

ການສະເໜີແຮງຈູງໃຈພິເສດສາມາດຊ່ວຍກະຕຸ້ນນັກຮຽນໃຫ້ຈັບຄູ່, ໂດຍສະເພາະກັບນັກຮຽນທີ່ມີຄວາມສາມາດຫຼາຍ. ຈະເຫັນວ່າມັນເປັນປະໂຫຍດທີ່ຈະໃຫ້ນັກຮຽນຈັບຄູ່ເຮັດວຽກຮ່ວມກັນພຽງແຕ່ຫນຶ່ງຫຼືສອງວຽກເທົ່ານັ້ນ.

Pair Programming Practice

ປ້ອງກັນການໂກງໃນການເຮັດວຽກຮ່ວມກັນ

ສິ່ງທ້າທາຍສຳລັບຄູແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງບຸກຄົນ, ໃນຂະນະທີ່ນຳໃຊ້ຜົນປະໂຫຍດຂອງການຮ່ວມມື. ຈະຮູ້ໄດ້ແນວໃດວ່ານັກຮຽນຕັ້ງໃຈເຮັດວຽກ ຫຼື ກົ້ງແຮງງານຜູ້ຮ່ວມງານ? ຜູ້ຊ່ຽວຊານແນະນຳໃຫ້ທົບທວນຄືນການອອກແບບຫຼັກສູດ ແລະ ການປະເມີນ ພ້ອມທັງປຶກສາຫາລືຢ່າງຈະແຈ້ງ ແລະ ຊັດເຈນກ່ຽວກັບພຶດຕິກຳຂອງນັກຮຽນທີ່ຈະຖືກຕີຄວາມວ່າຂີ້ຕົວະ. ຜູ້ຊ່ຽວຊານເນັ້ນໜັກໃຫ້ຄູເຮັດການມອບໝາຍໃຫ້ມີຄວາມໝາຍຕໍ່ນັກຮຽນ ແລະ ອະທິບາຍຄຸນຄ່າຂອງສິ່ງທີ່ນັກຮຽນຈະຮຽນຮູ້ໂດຍການເຮັດສຳເລັດ.

ສະພາບແວດລ້ອມການຮຽນຮູ້ໃນການຮ່ວມມື

ສະພາບແວດລ້ອມການຮຽນຮູ້ໃນຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ນັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳການຮຽນຮູ້. ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນສາມາດມີສ່ວນຮ່ວມທັງກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນໂດຍກົງ. ຕົວຢ່າງ, ນັກສຶກສາຄູ່ເຮັດວຽກມອບໝາຍຮ່ວມກັນໃນການຂຽນໂປຣແກຣມ; ນັກສຶກສາກຸ່ມນ້ອຍໆສົນທະນາຄຳຕອບທີ່ເປັນໄປໄດ້ຕໍ່ກັບຄຳຖາມຂອງອາຈານໃນລະຫວ່າງການບັນຍາຍ; ແລະ ນັກຮຽນເຮັດວຽກຮ່ວມກັນນອກຫ້ອງຮຽນເພື່ອຮຽນຮູ້ແນວຄວາມຄິດໃໝ່. ການຮຽນຮູ້ການຮ່ວມມືແມ່ນແຕກຕ່າງຈາກໂຄງການທີ່ນັກຮຽນ “divide and conquer.” ເມື່ອນັກຮຽນແບ່ງວຽກກັນ, ແຕ່ລະຄົນຮັບຜິດຊອບພຽງແຕ່ສ່ວນໜຶ່ງຂອງການແກ້ໄຂບັນຫາ ແລະ ຈະບໍ່ຄ່ອຍມີບັນຫາຫຍັງໃນການເຮັດວຽກຮ່ວມກັບຄົນອື່ນໃນທີມ. ໃນສະພາບແວດລ້ອມການເຮັດວຽກຮ່ວມກັນ, ນັກຮຽນມີສ່ວນຮ່ວມໃນການສົນທະນາປຶກສາຫາລືເຊິ່ງກັນແລະກັນ.

Q1. ຂຽນໂປຣແກຣມເພື່ອນັບຈຳນວນຄຳສັບທີ່ໃຊ້ໃນປະໂຫຍກທີ່ຜູ້ໃຊ້ປ້ອນເຂົ້າມາ.

- ໃນປະໂຫຍກທີ່ຜູ້ໃຊ້ປ້ອນເຂົ້າມາ, ແຕ່ລະຄຳສັບໄດ້ຖືກຈຳແນກໂດຍຍະຫວ່າງ.
- ໃຊ້ຟັງຊັນ input().split() ເພື່ອສ້າງ list S ທີ່ມີແຕ່ລະຂໍ້ຄວາມເປັນອົງປະກອບຂອງມັນ.
- ໃຊ້ຟັງຊັນ input() ເພື່ອຮັບຄຳສັບທີ່ຜູ້ໃຊ້ຈະຄົ້ນຫາ ແລະ ເອົາໄປເກັບໄວ້ໃນ x.
- ຟັງຊັນ word_count() ຮັບຄ່າ S ແລະ x ເປັນພາລາມິເຕີ ແລະ ສົ່ງຄືນຄ່າໂດຍການນັບຈຳນວນ x ທີ່ມີຢູ່ໃນ S.

```
1 S = list(input("Input a sentence: ").split())
2 x = input("Input a word to search: ")
3 count = word_count(S, x)
4 print(f"In S, {x} is appeared in {count} times.")
```

```
Input a sentence: the quick brown fox jumps over the lazy dog
Input a word to search: the
In S, the is appeared in 2 times.
```