

1. I/O / Process

• Variables

* វិសោធន៍យោះ : var_name = value

* រូប → នៅក្នុងការអាជីវកម្ម

Ex

$X = 1$
 $X = X + 2$
 $\text{print}(X)$

$1 + 2 = 3$
 \downarrow
 3

→ [3]

• Variable Types

String	int	float	boolean	list
- សម្រាប់ការពារ " "	- តួល់លើចុច្រឹត	- ពួកគេលើការគិតថ្លែង	- ត្រូវបានត្រួតពិនិត្យ	សម្រាប់ការបញ្ជី
Hi, Hello	1, 2 -10, 3	0.12, -1.2	True/False	[1, 2, 3], ['H', 'A']

* Operators ជាការងារណ៍ដូចជា រាយការនៃ variable types

ដែលការកំណត់រាយការនៃខ្លួនខ្លួននៃ int, float និង

* functions បានឱ្យ string ធ្វើពារិត

- int(string) : បានឱ្យ string ធ្វើ int

- float(string) : បានឱ្យ string ធ្វើ float

* - str(number) : បានឱ្យ number ធ្វើ string

• Math Operators

. + បុក

. - គុប

. * រូប

. / គុច

. // ឲ្យតាមការគិតទិន្នន័យ

. % ឲ្យតាមលទ្ធផល

. ** បកការលើ

- សម្រាប់រាយការ : $X = \frac{2ab}{y+1}$ → $X = (2*a*b)/(y+1)$

Math Operator Order

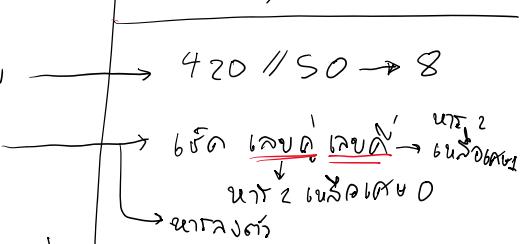
* គំរែកដោយការប្រើប្រាស់
ហៅការពិនិត្យ

1) ()

2) **

3) *, /, //, %

4) +, -



print function

- ↳ បន្ថែម ឈាមលទ្ធផល (output) នៅក្នុង ប្រព័ន្ធបញ្ជី ('/n')
- ↳ end = 'string' → នឹងនូវការបញ្ចប់ចុចខាងក្រោម រួចរាល់ជាមួយ 'string'
- Ex print("A", end=':')

print("B", end='-') → A:B-C

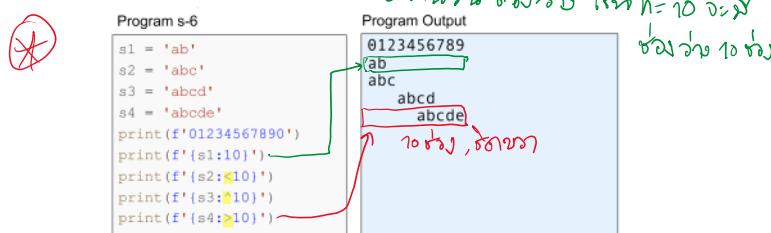
print("C")

f-string: formatted string

Syntax: f" {code} {10:.2f}"

- ↳ បន្ថែម f នូវ string ("")
- ↳ នៅ ឱ្យ ឯុទ្ធឌី និង code ដូច នូវការបញ្ជីការងារ
- Format នៃ លាស្ត

\${10.23:n,.mf}\$ float
 - d = int → ពិនិត្យ 0 លាក់អាមេរិក
 - commas នៃលាស្ត នូវការបញ្ជីការងារ (1000,000)
 - រាយការណ៍ នៃលាស្ត ដែល n=10 នៅក្បែង
 - លាស្ត នៃ 1:3d3



* < គុណករណ៍ , ^ និងការសម្រេច , > នៃ ឈាម

Named Constants

- ↳ គឺជាអតិថជន និង និរនោត នៃការបង្កើត និងការបង្កើត
- ↳ បន្ថែម និង ស្ថាបនិតាសាស្ត្រ
 - និង threshold នៃការបង្កើត if
 - និង នានាពាណិជ្ជកម្ម ឬផែនការ

Ex N_MAX = 10 0.S✓
N_MIN = 1 0.S✓

2. If / else

- Boolean Type: True , False

- Comparison Operators:

> , < , >= , <= , == , !=

- នាយករាយ និង True, False

- បន្ទាន់ការងារ: s>3 → s នៅនៅ 3 លើនៅ ? → True

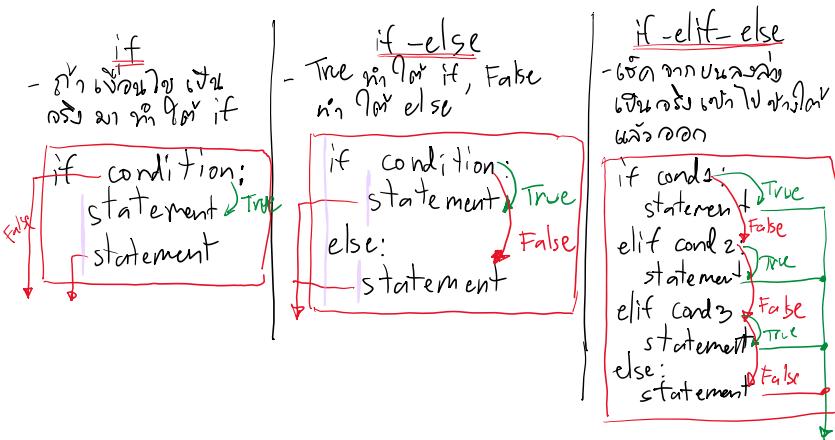
- → 1...n នាយករាយ។

- ວ່ານວ່າວິດຈະກຳນີ້ແມ່ນຫຼັງຈາກນີ້
- ລົງທະບຽນ: $s > 3 \rightarrow s$ ມີກວ່າ 3 ຍຸນ? \rightarrow True

- Boolean Operators:

and (ລົງທະບຽນ)	or (ລົງທະບຽນ)	not (ຍົກ)
- ລົງທະບຽນເປັນຫຼັງຈາກນີ້ ມີຄວາມສົນໃຈ True and True	- ລົງທະບຽນເປັນຫຼັງຈາກນີ້ ມີຄວາມສົນໃຈ False or False	- ລົງທະບຽນເປັນຫຼັງຈາກນີ້ ມີຄວາມສົນໃຈ True \rightarrow False

- Structures:



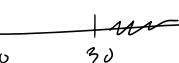
* ດີວ່າວິດຈະກຳ

ເຖິງອັນດັບໄປຈະກຳ min ຫຼື max

Ex) ດີວ່າວິດຈະກຳ 20 ຫຼື 30

if $20 \leq x \leq 30$

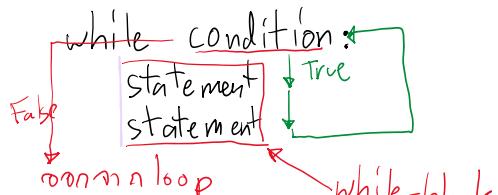
| if $20 \leq x$ and $x \leq 30$

ເຖິງອັນດັບໄປຈະກຳ 20 ຫຼື 30 
if $x < 20$ or $x > 30$

3. Loop

- while : ອັນດັບໄປຈະກຳ ໂອດນີ້ ນີ້ແລ້ວໂລດນີ້ ຢູ່ທີ່ມີຄວາມສົນໃຈ True

Syntax

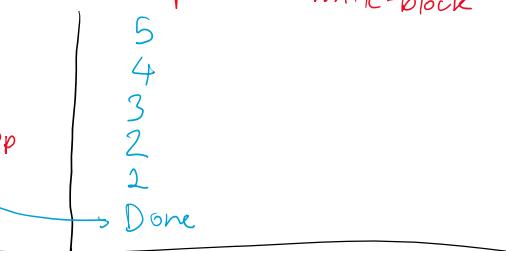


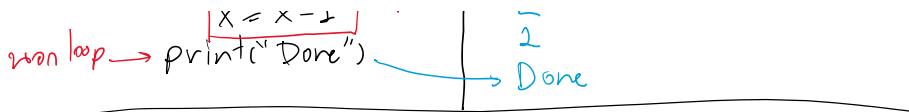
Ex) $x = 5$

while $x > 0$:

 print(x)
 x = x - 1

 loop
 print("Done")





④ Input Validation

↳ for while loop

ມີການ ນີ້ input validation ລົງທະບຽນ ມານວ່າ ຖໍ່ມີເລີຍຕົວໃຫຍ່ ອັນຫວີ

↳ ມີການ Named Constants !!

$$AGE_MIN = 0$$

$$AGE_MAX = 100$$

IVV 1

age = int(input(...))

while (age < AGE_MIN or
 age > AGE_MAX):

print("Wrong Age")
 age = int(input(...))

ບໍ່ໄດ້ຕົວ
 ຍັງບໍ່ໄດ້

IVV 2

while True:

age = int(input(...))

if (AGE_MIN <= age and
 age <= AGE_MAX):

break

print("Wrong Age")

ບໍ່ໄດ້ຕົວ
 ດ້ວຍກຳລົງ

* loop ສະແດງ ທີ່ user ສະໜັບສະໜັບ ພົບໃຈ
 ນີ້ມີການ loop ! → ບໍ່ໄດ້ຕົວ - 999999 ຂັ້ນ ພົບໃຈ

↳ ບໍ່ໄດ້ຕົວຍຸດ ສະໜັບສະໜັບ ນີ້ມີການ ດ້ວຍກຳລົງ ຢັງ break !

↳ ທີ່ໄດ້ຕົວ ມີ while True

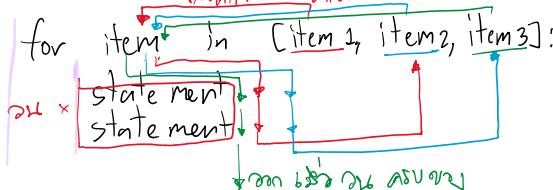
For loop

list

- ດັວງຈົບອານຸມາ ມີ ສຳເນົາຈົບອານຸມາ ຂະໜາ

inputting input

- Syntax:



Ex: for loops / iterations / range

- Ex ຕົວ ອິນເຕຣອນ ລາຍໄລ ຕະຫຼາມ 2 ຢູ່ 4 (ຕົວ 2, 4 ດີວ່າ)

count = 0
 for num in [10, 2, 3, 1, 5, 9]:

if 2 <= num and num <= 4:
 count += 1 # count = count + 1

print(count)

Short-hand notation

Count = Count + 1
 Count ↑
 Count += 1

$$- x \leftarrow 1 \rightarrow x = x * 2$$

$$- x \leftarrow (3+5) \rightarrow x = x / (3+5)$$

range function

↳ \Rightarrow sequence von \rightarrow von \leftarrow number stop

1) `range(start, stop, step)`

↳ ex: `range(10, 50, 5)` $\rightarrow [10, 15, 20, 25, 30, 35, 40, 45]$

2) `range(start, stop)` \rightarrow step=1

↳ ex: `range(20, 25)` $\rightarrow [20, 21, 22, 23, 24]$

3) `range(stop)` \rightarrow start=0, step=1

↳ ex: `range(4)` $\rightarrow [0, 1, 2, 3]$

List (Part 1): \rightarrow von variablen mit index (position)

• Syntax: `my_list = [item1, item2, item3]`

↳ Ex: `list1 = [10, 20, 30, 40]`

`print(list1)` $\rightarrow [10, 20, 30, 40]$

`list2 = ['A', 'aaa', 40]`

`print(list2)` $\rightarrow ['A', 'aaa', 40]$

• Indexing: `index 0`

↳ beispiel: `my_list[index]`

↳ Ex `my_list = [20, 30, 40]`

`print(my_list[1])` $\rightarrow [30]$

, Update: `on 2 index 98 =`

↳ Syntax: `my_list[index] = new_value`

↳ Ex `my_list = [20, 30, 40]`

`my_list[0] = 10`

`my_list[1] = my_list[0] + my_list[2]`

`print(my_list)` $\rightarrow [10, 50, 40]$

. len() function \rightarrow von \rightarrow von \rightarrow list

. on loop list on index 70

Ex `list1 = [10, 20, 30, 40]`

`list2 = [5, 10, 15, 20]`

`for i in range(len(list1))` \rightarrow range(4) $\rightarrow [0, 1, 2, 3]$

`[print(f'{i+1} {list1[i]} {list2[i]}')]` \rightarrow loop

`[print(f'{i+1} {list1[i]} {list2[i]}')]` \rightarrow loop

Output: 1 10 5

print([1, 10, 5, 20, 10, 30, 15, 40, 20])

Output:

1	10	5
2	20	10
3	30	15
4	40	20

List Operators

- $+$ → 6번 list 2번 add & 2번 (concatenation)
↳ 989번 list 6번 string

Ex1: list1 = [10, 20] | [10, 20] + [5, 7] → [10, 20, 5, 7]
 list2 = [5, 7]
 list3 = list1 + list2 → [10, 20, 5, 7]
 print(list3) → [10, 20, 5, 7]

Ex2: str1 = "AA",
 str2 = "bb",
 str3 = str1 + str2 → "AA" + "bb" → "AAbb"
 print(str3) → AAbb

- $*$ → 6번 list 2번 add & n번

→ Syntax: list * n

↳ Ex1: my_list = [1, 2, 3] ↳ 3번 반복되는 1, 2, 3

your_list = my_list * 3

[1, 2, 3] * 3 → [1, 2, 3, 1, 2, 3, 1, 2, 3]

print(your_list) → [1, 2, 3, 1, 2, 3, 1, 2, 3]

↳ Ex2: zeros = [0] * 5 → [0, 0, 0, 0, 0]

ones = [1] * 7 → [1, 1, 1, 1, 1, 1, 1]

Nested loop

```
total = 0
for n in range(10, 20):
    for m in [2, 3]:
        if n * m > 20:
            total += (n+m)
print(total)
```

#loop 1: outer

n = 10

#inloop 1:

m = 2

if 10*2 > 20:

X

#inloop 2:

m = 3

if 10*3 > 20: ✓

X

total += (10+3)

total = (10+3) + (15+2) + (15+3) → ...

#loop 2: (outer)

n=1 S

#loop 2:

M=2

if 2S*2 > 20:

+ total += (2S+2)

#loop2:

M=3

if 1S*3 > 20: ✓

+ total += (1S+3)