Chapter – 3 Variables and Operators

Contents

Chapter – 3 Variables and Operators

3.1 Variable ဆိုတာ ဘာတွေလဲ?

Variable ဆိုတာ တန်ဖိုး(value)များကို သိမ်းထားပေးမည့် memory location ဟုလည်း သိမှတ်နိုင်သည်။ တန်ဖိုး(value)များသည် နောင်တချိန်တွင် ပြောင်းလဲကောင်း ပြောင်းလဲနိုင်သည်။ တန်ဖိုး(value)များ ပိုင်ဆိုင်ထားသည့် ဂုဏ်သတ္တိ(property)များအရ data type များ ကွဲပြားသွားသည်။

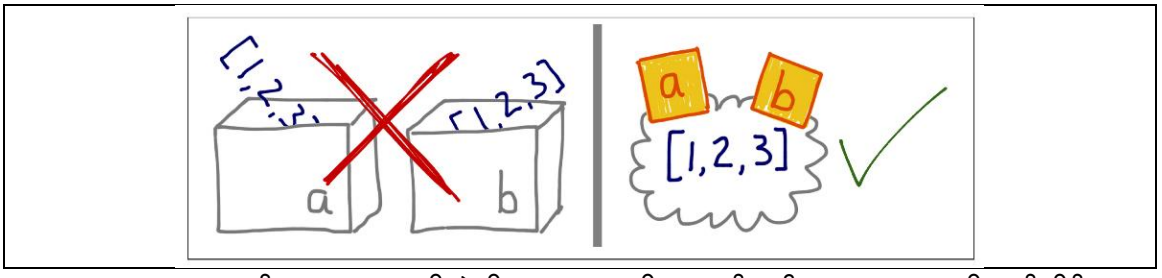
Python တွင် နံပါတ်(number)နှင့် စာသား(text value)များကို encode လုပ်ခွင့်ပြုထားသည်။ သခ်ၤာဆိုင်ရာ ပေါင်း၊ နုတ်၊ မြှောက်၊ စား လုပ်ခြင်းများ(arithmetic operations)ကို ပြုလုပ်နိုင်သည်။ ထိုသို့ ပြုလုပ်ရန် လိုအပ်သည့် variable နှင့် ရလာသည့် ရလာဒ်များ(results)ကို သိမ်းဆည်းရန် variable များ ပါဝင်သည်။ အထူး ပြုလုပ်ထားသည့် container(ကွန်တိန်နာ)များထဲတွင် ထိုတန်ဖိုးများကို ထည့်ထားသည်။

ကွန်တိန်နာထဲတွင် နံပါတ်(number)နှင့် စာသား(text value)တို့၏ တန်ဖိုးများ ထည့်ထားသည်။ ထိုကွန်တိန်နာကို အသုံးပြုရန် ပေးထားသည့် နာမည်သည် variable name ဖြစ်သည်။ **Python ရဲ့ variable တိုင်းတွင် သီးသန့် မည်သူ နှင့်မှ မတူသည့် နာမည်(variable name)တစ်ခုစီ ရှိရမည်။** ကွန်တိန်နာထဲတွင် ထည့်ထားသည့် တန်ဖိုးများ(content of the container)ရှိသည်။

ကွန်တိန်နာ ရှိနေသည့် တန်ဖိုးများ(value)သည် နံပါတ်(number)ဖြစ်နိုင်သလို စာသား(text value) များလည်း ဖြစ်နိုင်သည်။ Variable တွင် variable နာမည်(name) နှင့် variable ၏ တန်ဖိုး(value) ဟူ၍ အပိုင်း (၂)ပိုင်း ပါဝင်သည်။

Variable တစ်ခု ကြေငြာရန် ညီမျှခြင်းလက္ခဏာ(=)ကို သုံးသည်။ ညီမျှခြင်းလက္ခဏာ(=)၏ ဘယ်ဘက်တွင် variable နာမည်(name)ကို ထားရေးသည်။ ညီမျှခြင်းလက္ခဏာ(=)၏ ညာဘက်တွင် variable ၏ တန်ဖိုး (value) များကို ထားရေးသည်။

သခ်ၤာဘာသာရပ်တွင် ညီမျှခြင်းလက္ခဏာ(=)သည် ညီမျှသည်၊ တူညီသည်ဟုသည့် အဓိပ္ပာယ်(= means equal to)ကို ဆောင်သည်။ ကွန်ပျူတာဘာသာရပ်တွင် ကီးဘုတ်ပေါ်ရှိ ခလုပ်များကိုသာ အသုံးပြုရသောကြောင့် ညီမျှခြင်း လက္ခဏာ(=)ခလုပ်ကို assignment operator အဖြစ် အသုံးပြုသည်။ “a = 7” ၏ အဓိပ္ပာယ်မှာ a ဟုနာမည် ပေးထားသည့် ကွန်တိန်နာ(variable)ထဲတွင် တန်ဖိုး(value) 7 ကို ထည့်ထားသည်။ Assign လုပ်ထားသည်ဟု အဓိပ္ပာယ် ရသည်။ Variable ၏ နောက်ထပ် ဥပမာ တစ်ခုမှာ



a = [1,2,3] နှင့် b = [1,2,3] တို့ကဲ့သို့ ဒေတာ တူညီနေသည့်အခါ variable များကို ကွန်တိန်နာ ဟု မယူဆဘဲ Post-it notes သို့မဟုတ် နာမည်ရေး၍ ရသည့် အဝါရောင် စာရွက်ကလေးများကဲ့သို့လည်း မှတ်ယူနိုင်သည်။ [1,2,3] ကဲ့သို့ ဒေတာ တူညီနေသည့်အခါ ကွန်တိန်နာ နှစ်ခုအဖြစ် မပြုလုပ်ဘဲ ကွန်တိန်နာ တစ်ခု

အဖြစ်သာထားရှိပြီး နာမည် (၂)မျိုးပေးသည်။ Assignment နှစ်ခု ပြုလုပ်သည်။ Python တွင် `a, b = 400` ဟုလည်း ရေးနိုင်သည်။

```
a = 400
b = 400
id(a)
```

```
139736842153872
```

```
id(b)
```

```
139736842153872
```

Variable a နှင့် variable b တို့၏ တန်ဖိုးသည် 400 ဖြစ်သည်။ ထို့ကြောင့် Variable a နှင့် variable b တို့၏ id အတူတူပင်ဖြစ်သည်။ ကွန်တိန်နာတစ်ခု အဖြစ်သာထားရှိပြီး နာမည် (၂)မျိုးပေးထားခြင်း ဖြစ်သည်။

3.1.1 Variable နာမည်

Variable များသည် ပရိုဂရမ်တွင် အလိုအလျောက် ပေါ်မလာပါ။ ပရိုဂရမ်တွင် မည်သည့် variable ကို အသုံးပြုမည်ကို ပရိုဂရမ်မာက ဆုံးဖြတ်ပေးရမည်။ Variable တစ်ခုကို နာမည်တစ်ခု ပေးရန်အတွက် တင်းကျပ်သည့် စည်းမျဉ်းအချို့ကို လိုက်နာရမည်။ Variable အမည်ကို စာလုံးအကြီး၊ အသေးစာလုံးများ၊ ဂဏန်း များနှင့် _ (underscore) အက္ခရာများဖြင့် ရေးရမည်။ Variable name ပေးရာတွင် အောက်ပါ အချက်များကို သတိပြုရန် လိုအပ်သည်။

- Variable ၏အမည်ကို စာလုံး(alphabet/letter) ဖြင့် စတင်ရမည်။
- အသေးစာလုံးများ(Lowercase letters (a through z)) လက်ခံသည်။
- အကြီးစာလုံးများ(Uppercase letters (A through Z)) လက်ခံသည်။
- Digits(0 through 9) လက်ခံသော်လည်း variable name ၏ အစဆုံးနေရာသည် ကိန်း(digit) မဖြစ်ရပါ။
- Variable ၏အမည်တွင် space မပါဝင်ရ။ Underscore character (_)လက်ခံသည်။
- **: '" ,< > / ? | \ ! @ # % ^ & *~-+** စသည့် သင်္ကေတများ(symbols) မပါစေရ။
- PEP8 နှင့် အညီ နာမည်ကို အသေးစာလုံးများဖြင့် ရေးနိုင်အောင် လေ့ကျင့်သင့်သည်။
- Python built-in keyword များကို variable name အဖြစ် မသုံးရပါ။
- Variable ၏အမည်ကို အင်္ဂလိပ်အက္ခရာ အိုင်(I) ၊ အို (o)တစ်လုံးတည်း(single characters) မပေးသင့်ပါ။
- Python တွင် အသုံးပြုနေသည့် "list" ၊ "str" စသည့် စာလုံးများကိုလည်း Variable အမည် အဖြစ် အသုံးမပြုသင့်ပါ။
- Underscore (_) လက်ခံသည်။ Underscore ဖြင့် စထားသည့် variable name များအားလုံးသည် ထူးခြားသည့် variable များ ဖြစ်ကြသည်။ (treats in special ways)။

စာလုံးအကြီး(upper-case letters)နှင့် စာလုံးအသေး(lower-case letters)ကို မတူညီဟု Python က

သတ်မှတ်သည်။ အောက်တွင် ဖော်ပြထားသည့် အချက်များသည် function အမည်များနှင့် သက်ဆိုင်သည်ကို သတိပြုပါ။ Python သည် variable name များ၏ အရှည်ကို ကန့်သတ်မထားပါ။ သို့သော် အလွန် ရှည်လျားသော variable name များသည် ဖတ်ရန် ခက်ခဲသည်။ မှတ်မိရန် ခက်ခဲသည်။ ထို့ကြောင့် နားလည်လွယ်ပြီး တိုတောင်းသည် variable name များ ဖြစ်သင့်သည်။ **my_Variable , exchange_Rate , days_to_christmas** စသည်တို့သည် သင့်လျော်သည့် variable name များ ဖြစ်ကြသည်။

Python variable name တွင် လက်တင်အက္ခရာများ သာမက အခြားအက္ခရာများကိုလည်း အသုံးပြုနိုင်သည်။ အစဆုံးနေရာတွင် ကိန်းများဖြင့် စသည့် နာမည်များကို ခွင့်မပြုပါ(not start with a number)။ ဥပမာ- Exchange Rate, my list, စသည်ဖြင့် space ပါသော နာမည်များကို ခွင့်မပြုပါ။

PEP 8 သည် Python Code များ ရေးရန်အတွက် စတိုင်လမ်းညွှန်(Style Guide)ဖြစ်သည်။ Python ရှိ variable များနှင့် function များအတွက် အောက်ပါ အမည်ပေးခြင်း အချက်များကို အကြံပြုထားသည်။ ဖတ်ရှုနိုင်မှု လွယ်ကူရန်အတွက် variable အမည်များသည် စာလုံးအသေးများနှင့်စကားလုံးများဖြင့် ခွဲထားသည့် စကားလုံးများ ဖြစ်သင့်သည်။ Function name များအတွက် လိုက်နာရမည့် အချက်များသည် variable name များနှင့် အတူတူပင် ဖြစ်သည်။ (ဥပမာ၊ fun, my_function)။ ရောနှောစာလုံးပေါင်းသည့် mixed case ကိုလည်း အသုံးပြုနိုင်သည်၊ (ဥပမာ, myVariable)

ဥပမာ- စတုရန်း ဧရိယာတွက်မည့် variable name ကို squareArea ဟု အမည်လိုက်ခြင်းသည် auntJane ဆိုသည်နာမည်ထက် ပိုကောင်းပြီး ဖတ်သူက အလွယ်တကူ သိနိုင်သည်။

3.1.2 Keywords

Python ပရိဂရမ်တိုင်းတွင် keyword များ ပါဝင်သည်။ အောက်တွင် Python keyword များကို ဖော်ပြထားသည်။

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break',
'class', 'continue', 'def', 'del', 'elif', ' else ',' except ',' final
',' for ',' from ',' global ',' if ',' import ',' in ',' is ',' lambda
',' nonlocal ',' not ' , 'or', 'pass', 'raising', 'return', 'try',
'while', 'with', 'yield']]
```

Keyword များကို နာမည်အပြည့်အစုံမှာ reserved keyword ဖြစ်သည်။ Keyword များကို variable name အဖြစ် အသုံးပြုခွင့် မရှိပါ။ မသုံးရန် တားမြစ်ထားသည်။ (ဥပမာ- variable တစ်ခုကို import ဟု နာမည်ပေးရန် မဖြစ်နိုင်ပါ။) အဘယ်ကြောင့်ဆိုသော် သင်၏ variable နာမည်များ၊ function နာမည်များသည် keyword များဖြင့် ရောထွေး သွားလိမ့်မည်။ Reserved keyword များသည် ကြိုတင်သတ်မှတ်ထားပြီး reserve လုပ်ထားသောကြောင့် မည်သည့်နည်းနှင့်မျှ ပြောင်းလဲ၍ မရပါ။

Variable နာမည်များ ရေးသည့်အခါ အကြီးစာလုံး နှင့် အသေးစာလုံးတို့ သီးခြား ဖြစ်သည်။ Case sensitive ဖြစ်သည်။ Python တွင် အကြီးစာလုံး နှင့် အသေးစာလုံးသည် မတူသောကြောင့် အကြီးစာလုံးကို အသေးစာလုံး သို့ ပြောင်းလဲခြင်းဖြင့် variable name အသစ်တစ်ခု ဖန်တီးနိုင်သည်။

3.1.3 Variable တစ်ခု အတွင်းမှာ ဘာတွေထည့်လို့ ရလဲ။(Assigning Variables)

Variable တစ်ခုထဲတွင် မည်သည့်တန်ဖိုး(any value)မဆို သိမ်းဆည်း ထားနိုင်သည်။ နာမည် တစ်ခု ပေးပြီး မိမိထည့်လိုသည့် တန်ဖိုး(any value)တစ်ခုခု ထည့်ထားလိုက်လျှင် variable တစ်ခု တည်ဆောက်ပြီး ဖြစ်သည်။ Variable တစ်ခု၏ တန်ဖိုး(value)သည် သင်ထည့်လိုက်သောအရာ ဖြစ်သည်။

ကွန်ပျူတာ စကားဖြင့် ပြောလျှင် value တစ်ခုကို assign လုပ်ခြင်း(assigning a value)ဖြင့် variable တစ်ခု ဖြစ်ပေါ်လာသည်။ (A variable comes into existence as a result of assigning a value to it.)

တခြား language များနှင့် မတူသည့်အချက်မှ value များကို အသုံးမပြုခင် ကြိုတင်၍ ကြေငြာ(declare) နေမလိုပါ။ (you don't need to declare it in any special way.)။ ညီမျှခြင်း လက္ခဏာ (equal sign (=))ကို သုံး၍ variable တစ်ခု ပြုလုပ်နိုင်သည်။

Variable တစ်ခုထဲတွင် ထည့်ထားသည့် တန်ဖိုးနှင့် ထိုတန်ဖိုးနှင့် သက်ဆိုင်သည့် အမျိုးအစားသည် ပြောင်းလဲနေနိုင်သည်။ ကွဲပြားနိုင်သည်။ Variable တစ်ခုသည် အခုအချိန်တွင် integer ဖြစ်ပြီး၊ ခဏအကြာတွင် float တစ်ခု ဖြစ်လာနိုင်ပြီး နောက်ဆုံးတွင် string ဖြစ်လာလိမ့်သည်။ a ဆိုသည့် variable တစ်ခုထဲတွင် တန်ဖိုး 3 (integer) ထည့်ထားသည့်။ “3” (string) အဖြစ်သို့လည်း ပြောင်းနိုင်သည်။ 3 (integer) ကို 3.00 (float) အဖြစ်သို့ ပြောင်းနိုင်သည်။ Built-in **type()** function ကို အသုံးပြု၍ variable များ၏ data type ကို စစ်နိုင်သည်။

```
a = 3
print(type(a))    # variable a သည် integer data type ဖြစ်သည်။
```

```
<class 'int'>
```

```
a = str(3)        # integer ကို string အဖြစ် ပြောင်းသည်။
print(type(a))    # variable a သည် string data type ဖြစ်သည်။
```

```
<class 'str'>
```

```
a = 3/3
print(type(a))   # variable a သည် float data type ဖြစ်သည်။
```

```
<class 'float'>
```

```
var = 1                             # integer data type variable
account_balance = 1000.0            # float data type variable
client_name = 'John Doe'            # string data type variable
print(var, account_balance, client_name)
```

```
1 1000.0 John Doe
```

3.1.4 Variable များကို အသုံးပြုခြင်း

ရှိပြီးသား variable တစ်ခု အတွင်းမှ တန်ဖိုး ပြောင်းလဲခြင်း

```
my_var = 1                          # integer data type variable
print(my_var)
my_var = my_var + 1
print(My_var)
```

```
1
2
```

မှတ်ရန် အချက်များ

၁။ Variable ဆိုသည်မှာ တန်ဖိုးများကို သိမ်းဆည်းနိုင်သည့် memory ပေါ်ရှိ ကွန်တိန်နာတစ်ခု နှင့် ၎င်း၏ အမည် ဖြစ်သည်။ ပထမဆုံးအကြိမ် တန်ဖိုးသတ်မှတ်ပေးလိုက်သောအခါ variable တစ်ခုကို အလို အလျောက် ဖန်တီးပြီးသား ဖြစ်သည်။

၂။ Variable တိုင်းတွင် သီးခြားနာမည်တစ်ခု ရှိရမည်။ နာမည်သည် အက္ခရာစာလုံး အလွတ်များ သို့မဟုတ် underscore(_) သို့မဟုတ် အက္ခရာနှင့် စတင်ရမည်။ ၎င်းသည် Python reserved keyword မဖြစ်စေရ။ ပထမဆုံးအက္ခရာ မဟုတ်လျှင် underscores, letter နှင့် digit များဖြင့် လိုက်နိုင်သည်။ Python variable name တွင် အကြီးစာလုံး အသေးစာလုံးသည် လုံးဝ မတူညီပါ။

၃။ Python သည် dynamically-typed programming language တစ်ခုဖြစ်သည်။ Python တွင် variable များကို ကြေငြာ စရာ မလိုခြင်းဖြစ်သည်။ Variable ထဲတွင် တန်ဖိုးများကို assign လုပ်ရန် equal (=) sign ကို assignment operator အဖြစ် အသုံးပြုနိုင်သည်။

၄။ Variable ၏ တန်ဖိုးအသစ်များ assign လုပ်ရန် compound assignment operators သုံးနိုင်သည်။ (ဥပမာ var + = 1)

Python တွင် compound assignment operator များကို သုံး၍ အောက်ပါတိုင်း ရေးနိုင်သည်။

```
i = i + 2 * j               # ⇒ i += 2 * j
var = var / 2               # ⇒ var /= 2
rem = rem % 10              # ⇒ rem %= 10
j = j - (i + var + rem)     # ⇒ j -= (i + var + rem)
x = x ** 2                  # ⇒ x **= 2
```

ရှုပ်ထွေးနေသော variable အမည်များ အသုံးမပြုရန်နှင့် မပြည့်စုံသော အချက်အလက်များ ပါသည့် comment ကို မထားမိစေရန် အသိပြုပါ။

3.1.5 Built-in data containers

Variable များသည် ဒေတာများကို သိမ်းဆည်းထားပေးသောကြောင့် ဒေတာ ကွန်တိန်နာများ(data containers) ဟု၍လည်း ခေါ်ဆိုကြသည်။ အဆင်သင့် သုံးနိုင်အောင် ပြုလုပ်ထားပေးသည့် ဒေတာကွန်တိန်နာ များကို built-in data container များ ဟုခေါ်သည်။ Python ထဲတွင် built-in data container များရှိသည်။ ထို built-in data container များကို variable type ဟုခေါ်သည်။

Built-in data container များသည် variable မြောက်များစွာကို သိမ်းထား(hold)နိုင်သည့် data type များ ဖြစ်သည်။ သင့်အိမ်ရှိ အဝတ်အစားထည့်ထားသည့် သေတ္တာများ၊ ကျောက်မျက်ရတနာ ထည့်ထားသည့် သေတ္တာများ စသည်တို့ကဲ့သို့ integer များ ထည့်ထားမည့် built-in data container များ၊ float များ ထည့်ထား

မည့် built-in data container များ၊ string များ ထည့်ထားမည့် built-in data container များ စသည့်ဖြင့် အမျိုးမျိုး ကွဲပြားသည်။

Container တစ်ခုစီတွင် သက်ဆိုင်သည့် အမျိုးအစား(type) ကိုယ်စီရှိကြသည်။ ဥပမာ - list များ ထည့်ရန် ကွန်တိန်နာ၊ tuple များ ထည့်ရန် ကွန်တိန်နာ၊ set များ ထည့်ရန် ကွန်တိန်နာ၊ dict များ ထည့်ရန် ကွန်တိန်နာ စသည်ဖြင့် အမျိုးမျိုး ကွဲပြားကြသည်။ ထို့အတူ Python variable များအားလုံးတွင် သက်ဆိုင်သည့်အမျိုးအစားများ ရှိကြသည်။ Integer အမျိုးအစား variable ၊ floating-point အမျိုးအစား variable ၊ Strings (str) အမျိုးအစား variable များ စသည်ဖြင့် ကွဲပြားသည်။

Python တွင် built-in data type အမျိုးစုံပါဝင်သည်။ Data type များကိုလည်း ဒေတာဖွဲ့စည်း ထားပုံ (data structure)ဟုပြောဆိုလေ့ ရည်ညွှန်းလေ့ရှိသည်။ ယေဘူယျအားဖြင့် ကွန်ပျူတာသိပ္ပံတွင် ဒေတာဖွဲ့စည်းပုံ(data structure) အမျိုးအစား နှစ်မျိုးရှိသည်။

(၁) Primitive data structure သည် ရိုးရှင်းသည့်(simple) data type များဖြစ်သည်။

- Strings
- Integers
- Float
- Boolean

(၂) Non-primitive data structure များသည် complex data type များဖြစ်သည်။

- Lists
- Tuples
- Dictionaries
- Sets

ထိုအပြင် memoryview, bytearray, bytes, frozenset, range, and complex စသည့် ဒေတာအမျိုးအစားများလည်း ရှိသည်။ packages or libraries ထဲတွင် ပါရှိသည့် dataframes, arrays စသည့် ဒေတာအမျိုးအစားများလည်း ရှိသည်။

3.2 Determining variable type with type()

Built-in **type()** function ကို အသုံးပြု၍ variable များ၏ data type ကို စစ်နိုင်သည်။ အသုံးများသည့် data type များမှာ int (for integer), float, str (for string), list, tuple, dict (for dictionary), set, bool (for Boolean True/False) စသည်တို့ ဖြစ်သည်။

ဥပမာ-

```
a = 5
type(a)
my_var = (1,2)
type(my_var)
```

```
tuple
```

3.3 Numbers

Data type တစ်ခုချင်းစီတွင် အသုံးပြုရန်အတွက် သတ်မှတ်ထားသည့် စည်းကမ်းများ(specific rules) ရှိသည်။ ထို data type တစ်ခုချင်းစီကို ကွန်ပျူတာက တမျိုးစီ ကိုင်တွယ် စီမံဆောင်ရွက်သည်။ (handled differently by the computer)။ အသုံးပြုပုံချင်းလည်း မတူညီကြပေ။

Python တွင် booleans, integers, floats, strings စသည်တို့အပြင် ကြီးမားသည့် data structures များ function များနှင့် program များ အာလုံးတို့သည် object များဖြစ်သည်။ Object များသာ ဖြစ်သောကြောင့် စီမံဆောင်ရွက်နိုင်သည့် အချက်များ တူညီကြသည်။ စီမံဆောင်ရွက်ရမည့် ပုံစံများလည်း တူညီကြသည်။

3.4 Integers

သခ်ၤာသေဘာအရ integer မ်ားသည္ ကိန္းျပည့္မ်ား ျဖစ္ၾကသည္။ အေပါင္းကိန္းျပည့္(positive integer) နွင့္ အနုတ္ကိန္းျပည့္(negative integer)နွစ္ခုစလုံးပါဝင္သည္။ Integer မ်ားသည္ integer class ထဲတြင္ ပါဝင္သည္။ `<class 'int'>`

Integer မ်ားသည္ ကိန္းျပည့္မ်ား ျဖစ္သည္။ Negative infinite မွ positive infinite အထိ အစမရွိ အဆုံးမရွိေသာ ကိန္းျပည့္မ်ားသည္ integer data type ထဲတြင္ ပါဝင္သည္။ ကြန္ပ်ဴတာတြင္ ရနိုင္သမ်ွ မွတ္နိုင္သမ်ွ(maximum available memory) ျပည့္သွားသည့္တိုင္ေအာင္ မ်ားသည့္ ကိန္းမ်ား ျဖစ္နိုင္သည္။ Python တြင္ defines the ကို 'int' class ျဖင့္ သတ္မွတ္သည္။ ဥပမာ- 10, 4041, -7, 1111, 987 စသည္တို့သည့္ ကိန္းျပည့္မ်ား(integers) ျဖစ္ၾကသည္။

- Python တြင္ 5 ကို 05 ဟု ေရးပါက လက္မခံပါ။ SyntaxError: invalid token ေပးလိမ့္မည္။
- positive integer ကို 123 နွင့္ +123 ဟု၍ ေရးနိုင္သည္။ အေပါင္းလကၡဏာ(+)ထည့္၍ ေရးနိုင္သည္။ normal arithmetic ေပါင္း နုတ္ မ်ွေဘက္ စား လုပ္နိုင္သည္။
- 4 + 3 - 2 - 1 + 6 ကိန္းမ်ားကို တစ္ၿပိုင္နက္ ေပါင္း နုတ္ မ်ွေဘက္ စား လုပ္နိုင္သည္။
- ကိန္းနွင့္ အေပါင္းလကၡဏာ (+)အၾကားတြင္ space ထားခြင့္း မထားခြင့္းမွာ ဂရုစိုက္ရန္ မလို။ (ဥပမာ 5+9+3 နွင့္ 5+9 + 3 တို့ သည္ အဖြေတူၾကသည္။) သို့ေသာ္ မလိုအပ္ဘဲ space ထား၍ ေရးသားခြင့္းမ်ိဳး မျပလုပ္ရပါ။

အစား(division) (၂)မ်ိဳးရွိသည္။

/ သည္ floating-point (decimal) division ျဖစ္သည္။

// သည္ integer (truncating) division ျဖစ္သည္။

```
9/5     #    / operator ကို သုံးလ်ွင္ floating-point ရလဒ္(result)ေပးသည္။
```

1.8

// operator သည္ integer (truncating) division ျဖစ္သည္။ integer division ကို သုံးလ်ွင္ အဖြေသည္ integer အျဖစ္ ျဖတ္(Truncating လုပ္)ေပးသည္။ (integer division gives you an integer answer)။

```
9/5      #    // operator သည္ integer (truncating) division ျဖစ္သည္။
```

1

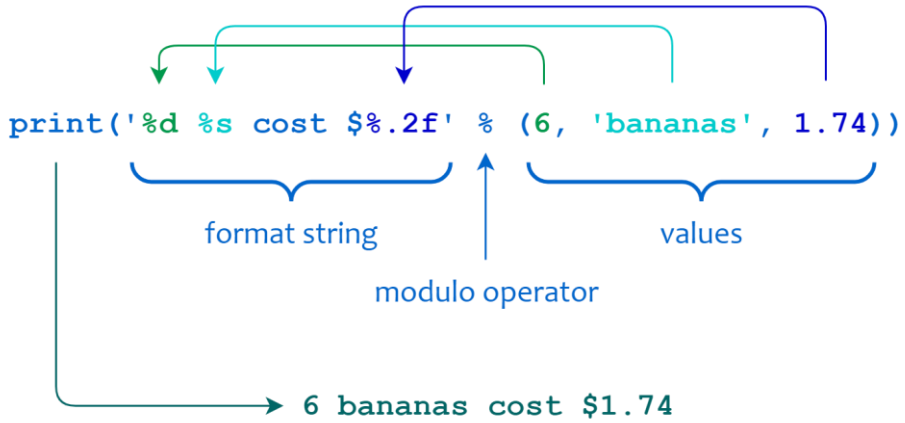
9 ကို 5 နွင့္စားလ်ွင္ 1.8 ရ ရမည္ျဖစ္ေသာ္လည္း ျဖတ္ခ်လိုက္ေသာေၾကာင့္ အဖြေသည္ (၁) ျဖစ္သည္။ ဒဿမ ရွ့ကာ ကိန္းကိုသာယူသည္။ (0.5 ကၽ်ာ္၍ တစ္တိုးခြင့္းမ်ိဳး မဟုတ္ပါ။) မည္သည့္ကိန္းကို မဆို သုည နွင့္ စားလ်ွင္ **ZeroDivisionError** ေပးလိမ့္မည္။

အထက်တန်းသချိုာတွင် ညီမျှခြင်းသကေ်တ(=)၏ အဓိပ္ပာယ်မှာ ဘယ်ဘက်နှင့် ညာဘက် တန်ဖိုး တူညီသည့် အဓိပ္ပာယ် ဖြစ်သည်။ Python တွင် ညီမျှခြင်းသကေ်တ(=)၏ အဓိပ္ပာယ်မှာ ညီမျှခြင်း ညာဖက်မှာ ကိန်းများကို ပေါင်း၊ နုတ်၊ မြှောက် စားလုပ်ပါ၍ ရသည့်တန်ဖိုးကို ဘယ်ဘက်ရှိ variable ထဲသို့ assign လုပ်ပါ ဖြစ်သည်။

ဥပမာ- 2 နှင့် 3 ကို ပေါင်းပါ။ ရရှိသည့် အဖြေ တန်ဖိုး 5 ကို variable a ထဲသို့ ထည့်(assign လုပ်)ပါ။

```
a = 2 + 3
```

5



3.5 Strings

String သည် စကားလုံး(word)များ သို့မဟုတ် character များ ဖြစ်ကြသည်။ Character များ အားလုံးသည် string အမျိုးအစားတွင် ပါဝင်သည်။ Python တွင် string သည် ပြောင်းလဲ၍ မရနိုင်သည့် Unicode character အစဉ်အတန်း ဖြစ်သည်။(a sequence of Unicode characters and are immutable) ။ Python တွင် string များသည် 'str' class တွင် ပါဝင်သည်။ `<class 'str'>`

Strings ကို ‘ ‘ သို့မဟုတ် “ “ (enclosed in single or double quotes) အတွင်းတွင် ထည့်ရေးရသည်။ ဥပမာ a = 3 တွင် 3 သည် integer ဖြစ်သည်။ တန်ဖိုး (၃)ခု သဘောကို ဆောင်သည်။ a = ”3” တွင် ”3” သည် အင်္ဂလိပ် အက္ခရာ 3 ဖြစ်သည်။ (၃)ခုဟု အဓိပ္ပာယ် မသက်ရောက်ပါ။

Object အမျိုးအစားများ

Python တွင် “class” နှင့် “type” သည် လုံးဝနီးပါး တူညီကြသည်။ (mean pretty much the same thing)

Prefix	Interpretation	Base
0b (zero + lowercase letter 'b') 0B (zero + uppercase letter 'B')	Binary	2
0o (zero + lowercase letter 'o') 0O (zero + uppercase letter 'O')	Octal	8

Prefix	Interpretation	Base
0x (zero + lowercase letter 'x') 0X (zero + uppercase letter 'X')	Hexadecimal	16

ဥပမာ-

Binary number

```
print(0b10)                          # Binary integer
print(type(0b10))
```

```
2
<class 'int'>
```

Octal number

```
print(0o10)                          # Octal integer
print(type(0o10))
```

```
8
<class 'int'>
```

Hexadecimal number

```
print(0x10)                          # Hexadecimal integer
print(type(0x10))
```

```
16
<class 'int'>
```

3.6 Float

Float သည် ဒဿမကိန်းများ ဖြစ်သည်။ Integer နှင့် Float တို့၏ ကွာခြားချက်မှာ ဒဿမပါခြင်း၊ မပါခြင်း ဖြစ်သည်။ Float ၏ နာမည်အပြည့်အစုံမှာ “floating point number” ဖြစ်သည်။ `class 'float'` ဖြစ်သည်။ ဒဿမ ထည့်ရေးလျှင် “floating point number” ဟု Python က အလိုအလျောက် သိသည်။ အဖြေတွင် “floating point number” ဖြစ်ရမည်ဆိုလျှင် ဒဿမ ထည့်ပေးသည်။ ဥပမာ 10 ဟု ထည့်ပေးလျှင် (integer)ဟု သတ်မှတ်ပေးလိုက်သည်။ ဥပမာ- 10.0 ဟု ထည့်ပေးလျှင် float ဟု သတ်မှတ်ပေးလိုက်သည်။

```
print(type(4))                       # 4 ၏ data type စစ်ရန်
print(type(4.0))                     # 4.0 ၏ data type စစ်ရန်
```

```
<class 'int'>
<class 'float'>
```

Floating point number ၏ ဒဿမ အရှေ့နှင့် နောက်တွင်တို့တွင် ကိန်းများစွာရှိနေလျှင် chara cter `'e'` သို့မဟုတ် `'E'` ဖြင့် ဖော်ပြရေးသားလေ့ရှိသည်။ ဥပမာ-

```
.4e7
```

```
4000000.0
```

```
type(.4e7)
```

```
<class 'float'>
```

```
4.2e-4
```

```
0.00042
```

3.7 Boolean

Boolean တွင် True နှင့် False ဆိုသည့် တန်ဖိုး (၂)မျိုးသာ ရှိသည်။ Booleans သည် Python built in data type များ ဖြစ်ကြသည်။ True နှင့် False ကို ရေးသည့်အခါ T နှင့် F ကို အကြီးစာလုံးဖြင့် ရေးရမည်။ Boolean သည် logical type ဖြစ်သည်။ ကျန် data type များ အားလုံး numeric type များသည် ဖြစ်သည်။

ကွန်ပျူတာထဲတွင် မှားသည်(False) သို့မဟုတ် မှန်သည်(True)ကို Boolean data type ဖြင့် သိမ်းဆည်းပြီး မှတ်ထားသည်။ "True" နှင့် "False" တို့သည် တန်ဖိုး(value)များ ဖြစ်ကြသည်။ "True" နှင့် "False" ကို integers 1 နှင့် 0 အဖြစ် အပြန်အလှန် ပြောင်းနိုင်သည်။ "True" မှ 1 အဖြစ်သို့လည်းကောင်း ၊ "False" မှ 0 အဖြစ်သို့လည်းကောင်း ပြောင်းနိုင်သည်။ class 'bool' အဖြစ် သတ်မှတ်သည်။

မှားသည်(False) သို့မဟုတ် မှန်သည်(True) စသည့် Boolean Value များကို သိမ်းဆည်းပေးမည့် variable သည် Boolean data type ဖြစ်သည်။

```
True
```

```
True
```

```
true                    # T ကို အသေးစာလုံးနှင့် ရေးလျှင် NameError ပေးလိမ့်မည်။
```

```
NameError: name 'true' is not defined
```

```
a = 3
b = 5
a == b
```

```
False
```

```
a != b                # != ကို not equal အတွက် သုံးသည်။
```

```
True
```

```
a > b                 # greater than operator
```

```
False
```

```
a < b                 # less than operator
```

```
True
```

```
print(type(True))      # data type ကို စစ်သည်။
```

```
<class 'bool'>
```

အပေါင်းတန်ဖိုး အနုတ်တန်ဖိုး ရှိသည့်ကိန်းများကို boolean အဖြစ်ပြောင်းလျှင် True ပေးသည်။ သုည (zero)ကို boolean အဖြစ် ပြောင်းလျှင် False ပေးသည်။ Python တွင် သုည(zero)ကို False အဖြစ် သတ်မှတ်ပြီး ကျန်ကိန်းများကို True အဖြစ် သတ်မှတ်သည်။

```
bool(28)
```

```
True
```

```
bool(-2.71828)   # အနုတ်တန်ဖိုးသည် True ဖြစ်သည်။
```

```
True
```

```
bool(0)       # သုည (zero)ကို boolean အဖြစ် ပြောင်းလျှင် False ပေးသည်။
```

```
False
```

String များကိုလည်း boolean အဖြစ်ပြောင်းနိုင်သည်။

Python တွင် empty string ကို False အဖြစ် သတ်မှတ်ပြီး ကျန် string များကို True အဖြစ် သတ်မှတ်သည်။

```
bool('Python')
```

```
True
```

```
bool('')  # empty string သည် False ဖြစ်သည်။
```

```
False
```

Boolean များကိုလည်း တခြားသော object type အဖြစ်သို့ ပြောင်းနိုင်သည်။ အောက်တွင် str() ကို သုံး၍ string data type သို့ပြောင်းသည်။

```
str(True)
```

```
'True'
```

```
str(False) # output 'False' တွင် quotes ပါသည်။ Boolean တွင် quotes မပါ ပါ။
```

```
'False'
```

Boolean များကို integer ကဲ့သို့ ပေါင်း၊ နုတ် ၊ မြှောက် ၊ စား လုပ်နိုင်သည်။

```
5 + True       # 5 +1 = 6
```

```
6
```

```
10 * False      # 10 * 0 = 0
```

```
0
```

```
5 / True        # 5 / 1 = 5
```

```
5.0
```

3.8 Complex Numbers

Complex number များတွင် real part နှင့် imaginary part ဟူ၍ အပိုင်း (၂)ပိုင်း ပါဝင်သည်။ အောက်ပါ အတိုင်း ရေးသားလေ့ ရှိသည်။

<real part> + <imaginary part>j

```
complex(2, 3)
```

```
(2+3j)
```

```
2+3j
type(2+3j)
```

```
(2+3j)
<class 'complex'>
```

```
abs(complex(2, 3))                # Complex number တစ်ခုကို absolute တန်ဖိုး ယူသည်။
```

```
3.605551275463989
```

Formatted string ကို သုံး၍ အောက်ပါအတိုင်း ရေးနိုင်သည်။

```
print("|{0}| = {1}".format(complex(2, 3), abs(complex(2, 3))))
```

```
|(2+3j)| = 3.605551275463989
```

Complex number မြှောက်ခြင်း

```
complex(1, 1), complex(2, 3)
```

```
((1+1j), (2+3j))
```

```
(1 + 2j) * (3+4j)
```

```
(-5+10j)
```

```
print("{0}*{1} = {2}".format(complex(1,1), complex(2,3), complex(1,1) *
                 complex(2, 3)))
```

```
(1+1j)*(2+3j) = (-1+5j)
```

Python Special Variables

Python တွင် ထူးခြားသော(special) variable များရှိသည်။ True, False, None နှင့် NotImplemented တို့ဖြစ်သည်။

None Is Not Zero!

None သည် သုည မဟုတ်ပါ။ None သည် Python တွင် special variable ဖြစ်သည်။ မည်သည့် တန်ဖိုးမျှ မရှိ(no value)ဟု အဓိပ္ပာယ် ရသည်။ မည်သည့်ကိန်းမျှ မရှိ။ သုည(zero)သည် ကိန်းတစ်ခု ဖြစ်သည်။ (Zero is a valid number)။ None သည် ဘာကိန်းမှ မရှိ။ Empty string ဖြစ်သည်။ Python မှ None သည် C/C++ မှ NULL ၊ JavaScript မှ null တို့နှင့် တူညီသည်။

NotImplemented Is Not None!

NotImplemented ၏ အဓိပ္ပာယ်ကို အဓိပ္ပာယ်တစ်ခုခုဖြင့် ဖော်ပြပြောဆိုရန် မဖြစ်နိုင်။ Not defined ဖြစ်သည်။ လုပ်ဆောင်ပေးရန် မဖြစ်နိုင်(action is impossible)ပေ။ Nonsensical သို့မဟုတ် non-existent ဟု အဓိပ္ပာယ် ရသည်။ ဥပမာ - string တစ်ခုကို ဒဿမကိန်း(float) တစ်ခုဖြင့် စားရန် မဖြစ်နိုင်ပါ။ TypeError ပေးလိမ့်မည်။

```
"Gorgus" / 2.718
```

```
TypeError Traceback (most recent call last)
TypeError: unsupported operand type(s) for /: 'str' and 'float'
```

3.9 Operators

Operator ဆိုသည်မှာ data နှင့် variable များကို ပေါင်း နုတ် မြှောက် စား လုပ်ရန် အတွက်သုံးသည့် syntax ဖြစ်သည်။ Python တွင် operator (၃)မျိုးရှိသည်။ Unary, binary နှင့် ternary တို့ဖြစ်သည်။

Unary operator သည် variable တစ်လုံးကိုသာ argument အဖြစ်ယူသည်။ တစ်နည်းအားဖြင့် variable တစ်လုံးတည်းကိုသာ အလုပ် လုပ်ပေးသည်။ ဥပမာ- variable တစ်လုံးဖြစ်သည့် (x)ကို အနုတ်ကိန်း ပြောင်းခြင်း။

Binary operator သည် variable နှစ်လုံးကို ပေါင်း နုတ် မြှောက် စား လုပ်ပေးခြင်းဖြစ်သည်။ ဥပမာ- x + y

Ternary operator သည် variable သုံးလုံးပါဝင်သည့် operation ဖြစ်သည်။ ဥပမာ- x = y = z

3.9.1 Unary operator

Name	Usage	Returns
Unary operators		
Positive	+x	For numeric types, returns x.
Negative	-x	For numeric types, returns -x.
Negation	not x	Logical negation; True becomes False and vice versa.
Bitwise Invert	~x	Changes all zeros to ones and vice versa in x's binary representation.
Deletion	del x	Deletes the variable x.
Call	x()	The result of x when used as a function.
Assertion	assert x	Ensures that bool(x) is True.

3.9.2 Binary operators

Name	Usage	Returns
Binary Operators		
Assignment	x = y	Set the name x to the value of y.
Attribute Access	x.y	Get the value of y which lives on the variable x.
Attribute Deletion	del x.y	Remove y from x.
Index	x[y]	The value of x at the location y.
Index Deletion	del x[y]	Remove the value of x at the location y.
Logical And	x and y	True if bool(x) and bool(y) are True, False otherwise.
Logical Or	x or y	x if bool(x) is True, otherwise the value of y.

3.9.3 Arithmetic Binary Operators

Name	Usage	Returns
Arithmetic Binary Operators		
Addition	x + y	The sum.
Subtraction	x - y	The difference.
Multiplication	x * y	The product.
Division	x / y	The quotient in Python 2 and true division in Python 3.
Floor Division	x // y	The quotient.
Modulo	x % y	The remainder.
Exponential	x ** y	x to the power of y.
Bitwise And	x & y	Ones where both x and y are one in the binary representation, zeros otherwise.
Bitwise Or	x | y	Ones where either x or y are one in the binary representation, zeros otherwise.
Bitwise Exclusive Or	x ^ y	Ones where either x or y but not both are one in the binary representation, zeros otherwise.
Left Shift	x << y	Shifts the binary representation of x up by y bits. For integers this has the effect of multiplying x by 2^y.
Right Shift	x >> y	Shifts the binary representation of x down by y bits. For integers this has the effect of dividing x by 2^y.
In-Place	x op= y	For each of the above operations, op may be replaced to create a version which acts on the variable 'in place'. This means that the operation will be performed and the result will immediately be assigned to x. For example, x += 1 will add one to x.

Combine the arithmetic operators

a = a – 3 ကို a -= 3 ဟုရေးသည်။ a -= 3 အဓိပ္ပာယ်မှာ a ထဲက 3 ကို နုတ်ပြီး a ထဲတွင် assign ပြန်လုပ်ခြင်း ဖြစ်သည်။

```
a = 95
a -= 3
a
```

92

```
a += 8      # ထို့အတူ a = a + 8 ကို a += 8 ဟုရေးသည်။
```

```
a
```

```
100
```

3.9.4 Comparison Binary Operators

Name	Usage	Returns
Comparison Binary Operators		
Equality	x == y	`True` or `False`.
Not Equal	x != y	`True` or `False`.
Less Than	x < y	`True` or `False`.
Less Than or Equal	x <= y	`True` or `False`.
Greater Than	x > y	`True` or `False`.
Greater Than or Equal	x >= y	`True` or `False`.
Containment	x in y	`True` if x is an element of y.
Non-Containment	x not in y	`False` if x is an element of y.
Identity Test	x is y	`True` if x and y point to the same underlying value *in memory*.
Not Identity Test	x is not y	`False` if x and y point to the same underlying value *in memory*.

3.9.5 Ternary Operators

Name	Usage	Returns
Ternary Operators		
Ternary Assignment	x = y = z	Set x and y to the value of z.
Attribute Assignment	x.y = z	Set `x.y` to be the value of z.
Index Assignment	x[y] = z	Set the location y of x to be the value of z.
Ternary Compare	x < y < z	`True` or `False`, equivalent to `(x < y) and (y < z)`. The < here may be replaced by >, <=, or >= in any permutation.
Ternary Or	x if y else z	x if `bool(y)` is `True` and z otherwise. It's equivalent to the C/C++ syntax `y?x:z`.

3.10 Operator precedence

```
2 + 3 * 4
```

အရင်ပေါင်းမှာလား? အရင်မြှောက်ရမလား? အရင်ပေါင်းလျှင် အဖြေသည် (၂၀) ဖြစ်သည်။ အရင်မြှောက်လျှင် အဖြေသည် (၁၄) ဖြစ်သည်။ Python မှာ အရင်မြှောက်ရသည်။ (multiplication has higher precedence than addition) Python operator များကိုယ်စီတွင် ဦးစားပေးအဆင့်(priority) များရှိကြသည်။ ဦးစားပေးအဆင့် (priority)သည့် Python operator ကို အရင်ဆုံး ဆောင်ရွက်ပေးရသည်။

အပေါ်ဘက်ရှိ operator များသည် အောက်ဘက်ရှိ operator များတွက် ဦးစားပေးအဆင့်(priority) မြင့်သောကြောင့် အရင်ဆုံး ဆောင်ရွက်ပေးရမည်။ ဥပမာ **()** အတွင်းရှိ ကိန်းများကို အရင် ပေါင်း နှုတ် မြှောက် စား လုပ်ရမည်။ အပေါ်ဆုံးရှိ operator ကို highest precedence ဟု သတ်မှတ်သည်။ အပေါ်ဆုံးရှိ operator ကို lowest precedence ဟု သတ်မှတ်သည်။

အပေါ်အောက် တူညီနေသည့် အကွက်ထဲတွင် ရှိလျှင် ဘယ်ဘက်တွင် ရှိသည့် operator သည် ညာဘက်တွင် ရှိသည့် operator ထက် ဦးစားပေးအဆင့်(priority) ပိုမြင့်သည်။ (Operators in the same box evaluate left to right.)

Python documentation မှ operator precedence ဖြစ်သည်။

Operator	Description
`()`	Parentheses (grouping)
`f(args...)`	Function call
`x[index:index]`	Slicing
`x[index]`	Subscription
`x.attribute`	Attribute reference
`**`	Exponentiation
`~x`	Bitwise not
`+x, -x`	Positive, negative
`*, /, %`	Multiplication, division, remainder
`+, -`	Addition, subtraction
`<<, >>`	Bitwise shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`in, not in, is, is not, <, <=, >, >=,< >, !=, ==`	Comparisons, membership, identity
`not x`	Boolean NOT

and	Boolean AND
or	Boolean OR
lambda	Lambda expression

Operator များကို မဖြစ်မနေ အမြဲလိုလို အသုံးပြုနေရသောကြောင့် နောက်ပိုင်းတွင် ဥပမာ များစွာ တွေ့မြင်ရမည် ဖြစ်သောကြောင့် operator ဥပမာများကို ဤအခန်းတွင် ဖော်ပြတော့ပါ။

Floating-point division

```
a = 200
b = a / 3
b
```

66.66666666666667

```
9 % 5                    # 9 ကို 5 နှင့်စား၍ ရသည့် အကြွင်းကိုရှာရန်
```

4

Python တွင် % character ကို နှစ်မျိုး သုံးမျိုးသုံး သည်။ 9 ကို 5 နှင့်စား၍ ရသည့် အကြွင်းကို ပေးသည်။ တစ်နည်းအားဖြင့် % အရှေ့တွင်ရှိသည့် ကိန်းကို တည်၍ % နောက်တွင် ရှိသည့်ကိန်းဖြင့် စားသည်။ ရသည့် အကြွင်းကို ပေးသည်။ 5 ဖြင့် စားသောကြောင့် ဖြစ်နိုင်သည့်အကြွင်းမှာ 0, 1, 2, 3, 4 တို့သာ ဖြစ်နိုင်သည်။

9 % 5

```
divmod(9,5)
```

(1, 4)

divmod()ကို သုံးလျှင် တည်ကိန်းနှင့် စားကိန်းထည့်ပေးရသည်။ တည်ကိန်းသည် 9 ဖြစ်သည်။ စားကိန်းသည် 5 ဖြစ်သည်။ ပြန်ထုတ်ပေးသည် ရလာဒ်မှာ (1 စားလဒ်, 4 အကြွင်း) ဖြစ်သည်။ ကိန်းနှစ်လုံး (two-item) ပြန်ထုတ်ပေးသောကြောင့် tuple ပုံစံဖြင့် ပြန်ထုတ်ပေးသည်။

အောက်တွင် base 10 မဟုတ်သည့် Binary ကိန်းပြည့်များ(integers)၊ Octal ကိန်းပြည့်များ (integers)၊ Hexadecimal ကိန်းပြည့်များ(integers) ကို ဖော်ပြထားသည်။

-End –