Let's Write Python

[2.5]

အပိုင်း ၂မှာပြောခဲ့တဲ့ Lists တွေအကြောင်းကိုပဲဆက်ပြောပါမယ်ဗျာ။

2.6 List methods

Python မှာ Lists အတွက်အသုံးပြုနိုင်တဲ့ methods လေးတွေရှိပါတယ်။ ပထမတစ်ခုကတော့ append() ပါ။

Code:

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```

ဒါဆို b ဆိုတဲ့ List ထပ်လုပ်ပြီး b ထဲက elements တွေကို a ထဲထည့်မယ်ဗျာ။ append() ကိုပဲသုံးရမလားဆိုတော့မဟုတ်ပါဘူး။ သုံးရင် Error တော့မတက်ပါဘူး ဒါပေမယ့် ကျွန်တော်တို့ ဖြစ်စေချင်သလိုဖြစ်မှာမဟုတ်ပါဘူး။

Code:

```
>>> a
[1, 2, 3, 4]
>>> b = [5, 6]
>>> a.append(b)
[1, 2, 3, 4, [5 ,6]]
```

မြင်တယ်မလားဗျ။ အခုလို append သုံးလိုက်ရင် b ရဲ့ Elements တွေကိုမထည့်လိုက်ပဲ b ဆိုတဲ့ List ကြီးကို nested အနေနဲ့ ထည့်လိုက်ပါတယ်။ ဒီတော့ဘယ်လိုလုပ်ကြမလဲ?
extend() ဆိုတဲ့နောက်ထပ် method ကိုသုံးပါမယ်။ သူကတော့ကျွန်တော်တို့ လိုချင်တဲ့ b ရဲ့ elements တွေကိုပဲယူပြီး a ထဲထည့်ဖို့ အတွက်ကိုလုပ်ဆောင်ပေးမှာပါ။

Code:

```
>>> a = [1, 2, 3]
>>> b = [4, 5]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5]
```

ဒါဆိုနောက် method တစ်ခုကိုထပ်သွားရအောင်ဗျာ။ sort() ဆိုတာကို alphabet အလိုက်ပြန်စီတာမျိုးတွေနောက် order အလိုက်(Low to High) ပြန်စီချင်ရင်အသုံးပြုပါတယ်။

Code:

```
>>> a = [‘e’, ‘b’, ‘d’, ‘f’, ‘a’, ‘c’]
>>> a.sort()
>>> a
[‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]
```

ဒါဆို ခုနကလို sort လုပ်တာကို variable တစ်ခုနဲ့ ပေးညီကြည့်လိုက်ပါ။

Code:

```
>>> a = [‘e’, ‘b’, ‘d’, ‘f’, ‘a’, ‘c’]
>>> b = a.sort()
>>> print b
None
```

ဘာမှပြန်မပြဘူးနော်။ ဘာလို့ လဲဆိုတော့ List methods တွေအကုန်လုံး List ကို modify လုပ်ပြီးရင် return None ပဲပြန်ပါတယ်။ ပြောရရင် List methods တွေအကုန်လုံးက void တွေပါပဲ။ C သမားတွေဆိုသိပြီးသားဖြစ်မှာပါ။ void function တွေဆို return 0 လို့ အောက်ဆုံးမှရေးစရာမလိုတာကို... အခုလဲဒီသဘောပါပဲ။

2.7 Map, filter and reduce

ကဲ a = [1, 2, 3] ဆိုတာထားပါတော့ ဒီထဲက elements တွေကိုပေါင်းတဲ့ရလဒ်သိချင်ဘယ်လိုလုပ်မလဲ? အခုဥပမာကတော့ဘယ်ပြောစရာလိုတုံး 6 ဆိုတာလူတိုင်းသိတယ် ဒါပေသိ elements က အခုဥ၊ ၄၊ ဆိုဘယ်လိုလုပ်မလဲ? sum() ဆိုတာလေးကိုသုံးပါမယ်။

Code:

```
>>> a = [1, 2, 3]
>>> sum(a)
6
```

အခုလိုမျိုး sequence of elements တွေကို single value ဖြစ်အောင်ပေါင်းတဲ့လုပ်ဆောင်ချက်ကို reduce လို့ ခေါ်ပါတယ်။
နေအုံးအခုလို sum() ဆိုတာကိုကိုယ့်ဘာသာရေးကြည့်မယ်ဆို...

Code:

```
def add_up(a):
    total = 0
    for x in a:
        total += x # equal to total = total + x
    return total
```

initial အနေနဲ့ total ကို သုညပေးထားပါတယ်။ a = [1, 2, 3] နဲ့ ပြောရမယ်ဆို အရင်ဆုံး a ကနေ 1 ကိုယူပါတယ် ပြီးရင် 0 ဆိုတဲ့ total ကို x

တန်ဖိုး 1 နဲ့ ပေါင်းပြီး total နဲ့ ပေးညီလိုက်တော့ total က 1 ဖြစ်သွားတယ်။ နောက် 2 ကိုယူပြီးပေါင်းတယ် ပြန်ပေးညီတော့ 3 ဖြစ်သွားတယ်။ နောက် 3 ကိုထပ်ယူပြီး total (လက်ရှိမှာ 1 နဲ့ 2 ကိုပေါင်းထားလို့ 3 ဖြစ်နေပါတယ်) ပေါင်းလိုက်တော့ 6 ဖြစ်သွားပါတယ်။ ပြီးရင်အဲ့ဒါကို return ပြန်လိုက်ပါတယ်။ ဒါပါပဲ။ total ကဲ့သို့ အသုံးပြုထွားရတဲ့ Variable ကို accumulator လို့ ခေါ်ပါတယ်။

ဒါဆို Map အကြောင်းဆက်ပြောဖို့ အောက်ကဥပမာကိုဆက်လေ့လာကြည့်ရအောင်...
ကျွန်တော်တို့ List ထဲမှာ small letter နဲ့ ရေးထားတဲ့ arguments တွေရှိတယ်ဆိုပါစို့ ဗျာ... ဒါတွေကို Capital Letter လေးတွေပြောင်းရအောင်...

Code:

```
def capitalizee(a):
    capi = []
    for x in a:
        capi.append(x.capitalize())
    return capi
```

ဒီ Example မှာဆို capi ဆိုတဲ့ empty list လေးတစ်ခုလုပ်လိုက်ပါတယ်။ ပြီးရင် Loop ပတ်ခိုင်းပြီးတော့ a ထဲက တစ်ခုချင်းစီကို capitalize() နဲ့ ပြောင်းပြီး capi ဆိုတဲ့ empty list ထဲကိုထည့်သွားတာပါ။
အခုသုံးပြသွားတာကတော့ capitalize() ဆိုတဲ့ argument ရဲ့ရှေ့ စာလုံးကို Capital Letter ပြောင်းတာပေါ့လေ။ နောက် small letter တွေပဲပြောင်းတဲ့ lower() ဆိုတာရယ်, စာလုံးအကုန်လုံးကိုအကြီးပြောင်းပေးတဲ့ upper() ဆိုတာရှိပါသေးတယ်။ အခုပေါ်က capitalizee() ကဲ့သို့ လုပ်ဆောင်တဲ့ function တွေကို map လို့ ခေါ်ပါတယ်။
ဒါဆိုဘယ်လိုမျိုးလုပ်ဆောင်ချက်တွေလုပ်တဲ့ function ကို Filter လို့ ခေါ်လဲပြောပြပါမယ်။

Code:

```
namez = ['I', 'love', 'MSF']

def thelower(a):
    lowlist = []
    for x in a:
        if x.islower():
            lowlist.append(x)
    return lowlist

def theupper(a):
    uplist = []
```

ပထမတစ်ခုမှာဆို love ဆိုတာလေးပဲပြန်ပေးမှာပါ။ ဒုတိယတစ်ခုမှာတော့ I နဲ့ MSF ကိုဖော်ပြမှာပါ။ islower() ရယ် isupper() ဆိုတာက True or False ပြန်ပေးပါတယ်။ ဒါကို if နဲ့ တွဲသုံးလိုက်တော့အကယ်လို့ True ပြန်ပေးရင် if အောက်က function ဖြစ်တဲ့ append() ကိုဆက်

လုပ်သွားတာပါ။ ဒီလို function မျိုးတွေကို filter လို့ ခေါ်တာပါ။

2.8 Deleting the elements of a list

List အကြောင်းစပြောကထဲက List တွေက mutable ဖြစ်တယ်ဆိုတာကိုပြောခဲ့ပါသလို အမျိုးမျိုးလေသုံးပြခဲ့ပါတယ်။ List တွေက mutable ဖြစ်တယ်ဆို elements တွေထည့်လိုက် sorting လုပ်တာလောက်တင်ဘယ်ကမလဲ။ မလိုတဲ့ elements တွေကိုလဲဖယ်ထုတ်ပစ်နိုင်ပါတယ်။

Delete an element using pop()

Code:

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> b = a.pop(0) # Remember, elements counting start from 0
>>> a
[2, 3, 4, 5, 6]
>>> b
1
>>> a.pop() # If we do not provide an index, it will remove last element and return that value.
6
```

Delete the element using “del”

Code:

```
>>> a = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’]
>>> del a[1]
>>> a
[‘a’, ‘c’, ‘d’, ‘e’]
```

Index အသုံးမပြုချင်ပဲကိုယ် delete လုပ်ချင်တဲ့ element ကိုသိတယ်ဆို remove() ကိုအသုံးပြုလို့ ရပါတယ်။

Code:

```
>>> a = [‘I’, ‘Love’, ‘MSF’]
>>> a.remove(‘Love’)
```

ဒီမှာက remove() ထဲမှာကျွန်တော်ဖျက်ချင်တဲ့ကောင်ကိုထည့်လိုက်တာပါ။ string ဆိုလဲ string ပေါ့။ integer ဆိုလဲ integer ပေါ့။ တစ်ခုမှတ်ထားရမှာက remove() က None ပဲ return ပြန်ပါတယ်။

ဒါဆို List ထဲမှာ Elements 10 ခုရှိတယ်ဗျာ။ 4 ခုမြောက်ကနေ 6 ခုမြောက်ထိဖျက်တယ်ဒါမျိုးဆိုဘယ်လိုလုပ်မလဲ? ပြောရရင် Element တစ်ခုထဲမဟုတ်ပဲ အများကြီးကိုတခါထဲနဲ့ ဖျက်ချင်ရင်ပေါ့။ ကျွန်တော်ဥပမာလေးပြပါမယ်။

Code:

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> del a[3:6]
>>> a
[1, 2, 3, 7, 8, 9, 10]
```

ကျွန်တော်အပေါ်မှာရေးပြခဲ့သလို index slicing ကို del နဲ့ တွဲသုံးလို့ ရပါတယ်။

2.9 Lists and strings

Characters တွေကိုအစဉ်လိုက်စုလိုက်ရင် String ပါ။ Values တွေကိုအစဉ်လိုက်စုလိုက်ရင် List ပေါ့ ဒါပေသိ List of Character တွေသည် String တစ်ခုမဟုတ်ပါ။ String တစ်ခုကို List ပြောင်းချင်ရင် list() ဆိုတာကိုအသုံးပြုနိုင်ပါတယ်။ အောက်မှာ abc ဆိုတဲ့ beer အဲလေ string လေးကို List ပြောင်းပြပါမယ်။

Code:

```
>>> a = 'abc'
>>> thelist = list(a)
>>> thelist
['a', 'b', 'c']
```

list() ဆိုတဲ့ function က string ထဲက character ကို letter တစ်ခုချင်းစီပြောင်းလိုက်ပါတယ်။ နောက် string တစ်ခုကိုခွဲထုတ်ချင်ရင် split() ဆိုတဲ့ function ကိုအသုံးပြုနိုင်ပါတယ်။

Code:

```
>>> a = 'I Love MSF'
>>> b = a.split()
>>> b
['I', 'Love', 'MSF']
```

နေအုံးဗျ အဲဒီ split() ဆိုတဲ့ function မှာ option character ရှိပါတယ်။ ဘယ်လိုသုံးတယ်ဆိုတာဥပမာလေးနဲ့ ပြပါတယ်။

Code:

```
>>> a = 'Hello-World-MSF'
>>> b = a.split('-')
>>> b
['Hello', 'World', 'MSF']
```

Option character ရဲ့သဘောတရားကဆိုရင်ကိုယ်ထည့်လိုက်တဲ့ argument ကိုဖယ်ပြီးအဲ့ကောင်ကနေပဲခွဲခွဲသွားမှာဖြစ်ပါတယ်။ ရှင်းသွားပေအောင်နောက်ဥပမာလေးလုပ်ရအောင်။

Code:

```
>>> a = ‘Hello World! MSF’
>>> b = a.split(‘!’)
>>> b
[‘Hello World’, ‘MSF’]
```

သဘောပေါက်မယ်ထင်ပါတယ်။
split() နဲ့ ခွဲလိုက်တာတော့ဟုတ်ပြီဗျာ.... ဟာဗ္ဗောလူ..မကျေနပ်ဘူးပြန်ပေါင်းချင်တယ်ဗျာ။ ဘယ်လိုလုပ်ရမလဲဆိုတော့သူခွဲပြောင်းပြန် join() ဆိုတာရှိပါတယ်။ အောက်ကဥပမာလေးကြည့်လိုက်ရအောင်...

Code:

```
>>> a = [‘Hello’, ‘World’, ‘MSF’]
>>> b = ‘ ‘ #There is a space between two single quotes
>>> b.join(a)
‘Hello World MSF’
```

မြင်တယ်မလားမသိ space နေရာမှာကိုယ်ဆက်ချင်တာကိုထည့်နိုင်ပါတယ်။

2.10 Objects and Values

Interpreter မှာဒီလိုလေးရေးလိုက်ပါ။

Code:

```
>>> a = ‘MSF’
>>> b = ‘MSF’
>>> c = [1, 2, 3]
>>> d = [1, 2, 3]
```

Variable တွေဖြစ်တဲ့ a ကော b ကောက MSF ဆိုတာတော့လူတိုင်းကြည့်တာနဲ့ သိပါတယ် ဒါပေသိ string တစ်ခုထဲကိုပဲရည်ညွှန်းထားတာလားဆိုတာကိုတော့မသိဘူးဗျာ။ မသိတော့ဘာလုပ်မလဲ? အဖြေရှာမှာပေါ့ဗျာ.... ဒီလိုအဖြေရှာဖို့ အတွက် Object တူမတူရှာမယ် ဒီလို Object တူမတူကိုသိဖို့ အတွက် is ဆိုတာလေးကိုအသုံးပြုကြပါမယ်။
a နဲ့ b ကိုအရင်ရှင်းမယ်ဗျာ။

Code:

```
>>> a is b
True
```

a နဲ့ b ကို is နဲ့အဖြေထုတ်တာ True ဆိုတော့ Object တူပါတယ်။ အောက်ကပုံလေးနဲ့ မြင်အောင်ဆွဲပြပေးထားပါတယ်။

ဒါဆိုနောက်တစ်ခုထပ်ပြောရအောင်...

Code:

```
>>> c is d
False
```

သဘောပေါက်မယ်ထင်ပါတယ်။ c နဲ့ d က value တူပေမယ့် မတူညီတဲ့ objects တွေကိုရည်ညွှန်းထားတာပါ။

2.11 Aliasing

a = b ဆိုရင် same object ကိုပဲ refer လုပ်တယ်ဆိုတာကိုတော့သိပြီးသားဖြစ်မှာပါ။
Object တစ်ခုမှာ refer လုပ်တာခုထက်ပိုရင် name ကလဲတစ်ခုထက်ပိုပါပဲ။ ဒီလို object ကို alias လို့ ခေါ်ပါတယ်။ အကယ်လို့ object ကသာ List တွေလို့ mutable ဖြစ်ခဲ့ရင် a ကိုခြိန်းရင် b မှာလဲသက်ရောက်မှုရှိတယ်ဆိုတာကိုအရင်သင်ခန်းစာမှာထဲကသိပြီးလောက်ပြီဖြစ်ပါတယ်။ Recall ပြန်ခေါ်ရမယ်ဆို

Code:

```
>>> a = [1, 2, 3]
>>> b = a
>>> b[0] = 11
>>> a
[11, 2, 3]
```

Alias က strings တွေလို immutable တွေဆိုဘာမှမဖြစ်ပေမယ့် List တို့လို mutable တွေကျသတိထားရပါတယ်။ ဒီလို List တွေကို changes လုပ်ရင် တစ်ခုခြိန်းရင်နောက်တစ်ခုမှ changes မဖြစ်စေချင်ရင်အသုံးပြုလို့ ရတဲ့နည်းကိုအရင်အပိုင်း ၂ မှာပြောပြပြီးသားပါ။ သိအုံးမယ်ထင်ပါတယ်။ (မေ့သွားရင်တော့ပြန်ဖတ်ပေါ့လေ 😁)

Let's Write Python [2.5] ကတော့ဒီလောက်ပါပဲ။ နောက်အပိုင်း ၃ကျရင် Tuple တွေအကြောင်းကိုလေ့လာကြပါမယ်။ ကျွန်တော်ဘာလို့ [3] လို့ မစပဲ [2.5] လို့ ရေးလဲဆို ဒီအခန်းကရှေ့ကအခန်းရဲ့အဆက်ဖြစ်နေလို့ ပါ။ ဒီကြားထဲမှာဆက်တိုက်မတင်ဖြစ်နိုင်တဲ့အတွက်လဲတောင်းပန်ပါတယ်ဗျာ။.....
Ref: How to think like a computer scientist

ENJOY THE BEAUTY OF PYTHON