Let's Write Python

[4]

ဒီနှစ်လျှောက်လည်တဲ့သူတွေမှားရှိလားတော့မသိ ကျွန်တော်တော့မလည်ဖြစ်တာ ၃နှစ်လောက်ရှိပြီဖူ အိမ်ထဲမှာပဲပုန်းနေတာ..... ထားပါ။ ကျွန်တော်တို့ အရင်တစ်ခေါက်က Tuple အကြောင်းပြောခဲ့ကြတယ်နော် ဒီတစ်ခေါက်တော့ Dictionary အကြောင်းလေးစလိုက်ကြရအောင်ဗျာ...

Dictionaries

သူကတော့ကျွန်တော်တို့ အရင်ကပြောခဲ့တဲ့ List လိုပါပဲ ဒါပေသိမတူတာက List မှာကို indices က integers ဖြစ်ပေမယ့် ဒီကောင်မှာတော့ indices က any types ပါ။ (Exceptions လေးတွေတော့ရှိတာပေါ့) ပြီးတော့သူက Curly Bracket တွန့် ကွင်းလေးနဲ့ ရေးတယ်ဗျ။ {}

Dictionary တစ်ခုမှာဆို Keys and Values ဆိုပြီးရှိတယ်ဗျ။ Key တစ်ခုစီတိုင်းကဆိုင်ရာဆိုင်ရာ value တွေနဲ့ ချိတ်ထားပါတယ်။ E လိုပြောရမယ်ဆို mapping ပေါ့။ ဒီလို key and value တွဲရက်ကို key-value pair လို့ ဆိုသလို item လို့ လဲခေါ်ပါသေးတယ်။ ပြီးတော့ key and value ကို : နဲ့ ခြားပြီး item တွေကြားကိုတော့ , နဲ့ ခြားပါတယ်။ Tuple တစ်ခုပြုလုပ်ရင် tuple() ဆိုတဲ့ function ရှိသလို dictionary တစ်ခုအတွက်လဲ dict() ဆိုတာရှိပါတယ်။

Code:

```
>>> means = dict()
>>> means
{}
>>> means['MSF'] = 'Myanmar Security Forum'
>>> means
{'MSF':'Myanmar Security Forum'}
```

ဒါပေမယ့် အခုလိုမရေးပဲ Dictionary ထဲမှာသင်က items ၃, ၄ ခုထည့်မယ်ဆိုအောက်ကလိုရေးသွားလို့ ရပါတယ်။

Code:

```
>>> means = {'MSF': 'Myanmar Security Forum', 'GNU': 'GNU is Not Unix', 'GPL': 'General Public
License'}
```

ဒါပေသိ အခုလို Dictionary လေးလုပ်ပြီး print ထုတ်ကြည့်ပါ။ ခုနကရေးလိုက်တဲ့ order အတိုင်းပြန်ထွက်လာမှာမပ ဂုတ်ပါဘူး။ သို့ သော်

ဘာမှတော့အရေးထားစရာမလိုပါဘူး ကျွန်တော်တို့ က key နဲ့ ပဲ value ကိုပြန်ခေါ်မှာပါ။ Lists တို့ လို index နဲ့ ခေါ်စရာမှမလိုတာ။ နောက် dictionary တစ်ခုထဲမှာကိုယ်သိချင်တဲ့ key ရှိမရှိကို in နဲ့ စစ်ကြည့်လို့ ရပါတယ်။ သို့ သော်သူက key ကိုပဲတိုက်သွားမှာနော် value နဲ့ တိုက်တာမပ ုတ်ပါဘူး ဒါလေးကိုတော့သတိထားပေးပါ။

Code:

```
>>> means['MSF']
'Myanmar Security Forum'
>>> 'MSF' in means
True
>>> 'Myanmar Security Forum' in means
False
>>> # Because there's no key called 'Myanmar Security Forum' in means
…
>>>
```

သဘောပေါက်တယ်မလားဗျ။ သူက key ကိုပဲတိုက်စစ်တာပါ။ ဒုတိယတစ်ခုမှာကျတော့ 'Myanmar Security Forum' ဆိုတဲ့ key က မရှိဘူးလေ ဒီအတွက်ကြောင့် False ပြန်တာပါ။

4.1 Dictionary as a set of counters

ကျွန်တော်တို့ Word Counter လေးတစ်ခုလောက်ရေးကြည့်ရအောင်ဗျာ။

Code:

```
def counter(words):
    thedictionary = dict()              #1
    for x in words:                        #2
        if x not in thedictionary:        #3
            thedictionary[x] = 1
        else:                              #4
            thedictionary[x] += 1
    return thedictionary
```

#1 မှာကျွန်တော်တို့ အရင်ဆုံး empty dictionary တစ်ခုလုပ်လိုက်ပါတယ်နောက် #2 မှာကျွန်တော်တို့ for နဲ့ looping ပတ်ပါတယ်။ အဲ့သလို looping ပတ်တဲ့အခြိန်မှာ #3 မှာဆိုအကယ်လို့ looping ပတ်တဲ့ထဲကကောင်က thedictionary ဆိုတဲ့ dictionary ထဲမှာမပါခဲ့ရင် အဲ့ word ကို key အဖြစ်ထားပြီး value 1 နဲ့ ပေးသိပေးပါတယ်။ #4 ဆိုတဲ့ else (မပါတာမပ ုတ်တော့ Dictionary ထဲမှာပါပြီးသားဖြစ်နေရင်) ဒါဆိုရင်တော့ value ကို 1 ပေါင်းပေးသ[သြား]လိုက်ပါတယ်။ နောက်ပတ်စရာရှိတာပတ်ပြီးသွားရင်တော့ dictionary ကို return ပြန်ပါတယ်။ ပိုရှင်းအောင် function လေးကို interpreter မှာရေးပြီးတော့စမ်းကြည့်ပေါ့ဗျာ။

4.2 Reverse lookup

ကျွန်တော်တို့ တွေ Value Lookup Program လေးတစ်ခုဆို dictionary ကို d လို့ ဆိုပါစို့ နောက် key ကိုတော့ k ပေါ့။ ဒါဆို key နဲ့ သက်ဆိုင်ရာ value ဆိုရင်တော့ v = d[k] ဖြစ်မယ်ပေါ့ ဒီလိုမျိုးကို lookup လို့ ခေါ်ပါတယ် ဒါပေသိ ကျွန်တော်တို့ က value ကိုရိုက်ထည့်ပြီး key ကိုပြန်သိချင်ရင်ဘယ်လိုလုပ်ကြမလဲ?
ဒီလိုမျိုးအတွက်တွေးကြည့်တော့ပြဿနာလေးရှိလာပါတယ်။ အဲ့ဲတာက အဲ့ဲဒီ value ကိုပဲ key ဘယ်နခုကတောင် mapping ထားတာလဲ? ပြောရရင်ဖွာ d = {'a': 'MSF', 'b': 'MSF'} ဒါမျိုးကိုပြောတာပါ။ အဲ့လိုမျိုးဖြစ်ခဲ့ရင်ကော ကျွန်တော်တို့ ကဘယ်လို return ပြန်စေချင်တာလဲ။
အောက်ကကောင်လေးကတော့ value ကိုတောင်းပြီးတော့ အဲ့ value နဲ့ mapping ဖြစ်နေပြီး စတွေ့ တွေ့ ချင်း key ကို return ပြန်မှာပါ။

Code:

```
def reverse_lookup(d, v):
    for k in d:
        if d[k] == v:
        return k
    raise ValueError
```

for နဲ့ looping ပတ်ထားတာမှာသဘောက d ဆိုတဲ့ dictionary ထဲမှာရှိနေတဲ့ k ဆိုတဲ့ key တွေထဲက အကယ်လို့ key ရဲ့ value (d[k]) ကသာ v ဆိုတဲ့ value နဲ့ တူညီနေမယ်ဆိုရင် အဲ့ key ကို return ပြန်ပေးပါလို့ ပြောတာပါ။ နောက်တူညီတာမျိုးမရှိရင် ValueError ဆိုတဲ့ exception ကိုဖော်ပြခိုင်းတာပါ။ ရှင်းတယ်မလားဗျ။ ပိုမြင်လ[ဿယ်သွားအောင် မိမိဘာသာ dictionary တစ်ခုဆောက်နောက် d မှာ asign လုပ်နောက် v မှာလဲ search ချင်တဲ့ value ကိုထည့်ပြီးစမ်းကြည့်လိုက်ပါ။ ဒီလောက်တော့လုပ်တတ်မယ်ထင်ပါတယ်နော် ကျွန်တော်ရေးမပြတော့ပါဘူး။

4.3 Dictionaries and Lists

Dictionaries တွေထဲမှာ Lists တွေကို values အနေနဲ့ သုံးနိုင်ပါတယ်။
Key နဲ့ Value ကို invert လုပ်မယ့် program လေးတစ်ခုကိုတွေးကြည့်ရအောင်ဗျာ။ Function မှာအရင်ဆုံး empty dictionary တစ်ခုဆောက်ထားမယ် function parenthesis မှာ dictionary တောင်းမယ် ဒီ program ကနေ key နဲ့ value ကိုယူမယ်ဗျာ။ ဒီအထိတော့အိုကေပြီ ဒါဆို condition တစ်ခုလုပ်မယ်အဲ့ဲ condition ထဲမှာ value ကမရှိရင် value ကို key အနေနဲ့ ယူမယ်နောက် equal sign ရဲ့ right hand side မှာ key ကိုရေးမယ်။ အကယ်လို့ value ကရှိနေမယ်ဆိုရင် သူရဲ့ key ကို append လုပ်မယ်။ နောက်ဆုံးရလာတဲ့ result ကို return ပြန်လိုက်ပါမယ်။
စားသွားလားမသိဘူး။ ပြောနေကြာတယ် ရှင်းသွားအောင်အောက်က code လေးတွေကိုဖတ်လိုက်ပါတော့။

Code:

```
def invert_dict(d):
    inverse = dict()
    for key in d:
        val = d[key]
        if val not in inverse:
            inverse[val] = [key]
        else:
            inverse[val].append(key)
    return inverse
```

အဲ့ function လေးကိုမိမိဘာသာစမ်းကြည့်လိုက်ပါ။ စမ်းပြီးရင်မေးစရာတစ်ခုရှိပါလိမ့်မယ် ကျွပ်ထည့်လိုက်တုံးက string တွေလေ invert ပြန်ထွက်လာတော့ ဘာလို့ value တွေက List ဖြစ်နေရတာလဲ? ဒီလိုဖူ (ဒီလိုမပ ဂုတ်လဲပ ဂိုလို 😜) အမှန်ဆို key and value invert လုပ်ဖို့ အတွက်က List မလုပ်လဲရပါတယ် ဒါပေသိ ကျွန်တော်ကအပေါ်မှာရေးခဲ့တဲ့ counter() ဆိုတဲ့ function လေးနဲ့တွဲသုံးပြစရာလေးရှိလို့ ကျွန်တော့်အစမှာ Lists က value ဖြစ်နိုင်တယ်လို့ ပြောတယ် ပ ဂုတ်တယ်ပ ဂုတ်.... အကယ်လို့သာခင်ဗျားစီမှာ Letters ကနေ frequencies ကို map တဲ့ dictionary ကို frequencies ကနေ letters ကိုပြန် map တဲ့ dictionary လိုချင်တယ်ဗျာ invert မယ်ပေါ့ ဒါပေသိ အကယ်လို့သာ frequency တူတဲ့ letter တွေသာပါလာမယ်ဆို inverted dictionary ထဲမှာပါလဲမယ့် value တွေသည် string မပ ဂုတ်ပဲ Listed Letter တွေြဖစ်နေရပါမယ်။
လည်နေသေးတယ်ထင်တယ်။
(counter() ဆိုတဲ့ကောင်က built-in မပ ဂုတ်ဘူးနော် အပေါ်မှာကျွန်တော်တို့ ရေးခဲ့ပြီးသားပါ)

Code:

```
>>> mydict = counter('hello')
>>> mydict
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
>>>
>>> inverse = invert_dict(mydict)
>>> inverse
{1: ['h', 'e', 'o'], 2: ['l']}
```

အခုလိုမျိုး frequency တစ်ခုထဲရှိတဲ့ letter တွေကိုတစ်တွဲ နောက် frequency နှစ်ခုရှိတာကိုတစ်တွဲဒါမျိုးစုခွင်တဲ့အတွက် invert_dict() မှာကျွန်တော်တို့ value ပြန်ရင် List နဲ့ လုပ်ခဲ့တာပါ။

ကျွန်တော်ကက Lists တွေကို Dictionary ရဲ့ Value ဖြစ်နိုင်တယ်လို့ ပြောခဲ့တယ်။ နေအုံးဒါဆို key ကောမဖြစ်နိုင်ဘူးလားဆိုတော့ မဖြစ်နိုင်ပါဘူးဘာလို့ လဲဆိုတာအောက်ကဥပမာလေးနဲ့ ပြပါမယ်။

Code:

```
>>> x = [1, 2, 3]
>>> y = dict()    # Creating Empty Dictionary
>>> y[x] = 'value'
```

ဒီတော့ဘာဆက်ဖြစ်မလဲ? ဘာဖြစ်မလဲဆိုတော့ Error တက်တာပေါ့ကွယ် ဒါဆိုဘယ်လို Error လဲ? ပ ဂိုလို Error အဲလေ TypeError ပါ။ တိတိကျကျပြောရရင် TypeError: list objects are unhashable
ဘာလဲ unhashable? ဒီလိုဖူ (လာပြန်ပြီ အဲ့ဒီ ဒီလိုဖူ က 😜) Dictionary က hashtable ကိုအသုံးပြုပြီး implementation ပြုလုပ်ပါတယ်ဆိုတော့ကာပြောခွင်တာက keys တွေသည် hashable ဖြစ်ရပါမယ် ဒါပေသိအခုပြောပုံဆို Lists က hashable မဖြစ်ဘူး ဒါကိုဆိုလိုတာပါ။

hash ဆိုတာက value တစ်ခုကိုယူပြီး integer ပြန်ပေးတဲ့ function တစ်ခုပါပဲ။ Dictionaries တွေက items (keys and values) တွေကိုကြာညှဲ့ဖို့ အတွက်ခုနက return ပြန်ပေးတဲ့ integers တနည်းပြောရရင် hash values တွေကိုအသုံးပြုပါတယ်။
Keys တွေက immutable ဖြစ်နေရင်ပြောစရာမရှိပါဘူး ဒါပေသိခုနက List လို mutable type ဆိုရင်ပြောစရာရှိလာပြီ။ ဒီလိုဖူ (လာပြန်ပြီ 😜) သင်က key-value pair တစ်ခုကို create လုပ်လိုက်ပြီဆို Python က key ကို hash လုပ်ပြီး corresponding location တစ်ခုခုမှာ သိမ်းထားလိုက်ပါတယ်။ အကယ်လို့ သာသင်က key ကိုခိုန်းလိုက်တယ်ဆို Python ကလဲ hash ပြန်လုပ်ပြီးနောက်တစ်နေရာမှာထပ်သသိမ်းတယ် ဒါဆို key တစ်ခုအတွက် entry ၂ခုဖြစ်သွားပြီ ဒါမှမပ ဂုတ်ရင် key ကိုရှာကိုမတွေ့ တာပဲဖြစ်သွားမှာပါ။ ဒါဆို dictionary ကအလုပ်မလုပ်ေတာ့ဘူးပေါ့ဖျာ။
ဒါကြောင့် keys တွေသည် hashable နဲ့ immutable ဖြစ်ရတာပါ။ ဒါကြောင့် constant အဖြစ်သုံးနိုင်တဲ့ tuple တွေကို key အဖြစ်သုံးနိုင်တာပါ။ (tuple အကြောင်းကိုအရင်အခန်းမှာပြောခဲ့ပြီးပါပြီ။)

4.4 Global Variables

Python မှာ special frame ဖြစ်တဲ့ __main__ ဆိုတာရှိတယ်ဖျ။ __main__ ထဲမှာရှိတဲ့ Variables တွေသည် global variables တွေပါ ဘာလို့ လဲဆိုတဲ့ တခြား function တွေနေယူသုံးလို့ရလို့ ပါ။ နောက်ကြုံတုံးပြောအုံးမယ် Local Variables တွေဆိုတာလဲရှိပါတယ် သူတို့ ကတော့ သူတို့ ရဲ့ function ပြီးသွားရင် function နဲ့ အတူဖျောက်သွားတယ်ပေါ့။ နောက် Global Variables တွေကို flags တွေအနေနဲ့လဲသုံးကြပါတယ်။ အောက်ကဥပမာလေးလိုပေါ့။

Code:

```
verbose = True

def example1():
    if verbose:
        print “It's True!”
```

ရှင်းစရာမလိုဘူးထင်ပါတယ် သဘောပေါက်တယ်ပ ဂုတ်တယ်ပ ဂုတ်။ နောက် Global, Local ကိုစဖတ်တဲ့သူတွေမှားတာတစ်ခုရှိတယ်ဖျ အဲ့တာကတော့...

Code:

```
verbose = False

def example2():
    verbose = True
```

အဲ့မှာစတာပဲကိုယ့်လူရေ သူတို့ ကတော့ verbose ရဲ့ value ကိုခွိုန်းတယ်ပေါ့ ဒါပေသိထင်သလိုမဖြစ်ပဲမပြောင်းဘူးဗျ။ ဒါကတကယ်တော့ Global ကကောင်ကိုခွိုန်းတာဖြစ်မသဿြားပဲနဲ့ Local ထဲမှာ verbose ဆိုတဲ့ variable အသစ်တစ်ခုလုပ်လိုက်တာပါ။ ဒီတော့ Global က variable ကိုသက်ရောက်မှုမရှိဘူးပေါ့ဗျာ။ ဒါဆိုပေ ?လူ ကျုပ်ကတော့ Global ကကောင်ကိုခွိုန်းချင်တယ်ဗျ ဒီလိုမလုပ်လို့ ဘယ်လိုသွားလုပ်မှာလဲ? ဒီလိုရှိတယ်ဗျ Global ကကောင်ကိုခွိုန်းခွင်အရင်ဆုံး declare ကြေညာပေးရတယ်ပေါ့ဗျာ။ (ကဲကဲပေ ဂျင့် Global က verbose လာခဲ့စမ်း 😋)

Code:

```
verbose = False

def example2():
    global verbose
    verbose = True
```

အောက်ကကောင်လေးကတော့ Global က variable ကိုခွိုန်းချင်တဲ့ကောင်လေးပေါ့ အဲ့လိုခွိုန်းဖို့ လုပ်တဲ့အခါမှာ UnboundLocalError တက်ပါတယ်။

Code:

```
>>> count = 0
>>> def example3():
...     count += 1
...
UnboundLocalError: ….............
```

ဒီတော့ဘာခက်သလဲအပေါ်မှာပြောသလိုကြေညာပေးလိုက်ပေါ့ကွယ် မရေးပြတော့ဘူးနော်ရေးတတ်တယ်မလား 😁
(တစ်ခုတော့ပြောအုံးမယ် အကယ်လို Variable ကသာ mutable type ဖြစ်နေခဲ့ရင်တော့ declare လုပ်စရာမလိုပါဘူး)

Dictionary အကြောင်းကတော့ဒီလောက်ပါပဲ။

Enjoy The Beauty of Python