Phone Myat Min

a book for Beginner Programming observer...

```
class interface:
    def __init__(self):
        self.win = Tk()
        self.win.title("English To Myanmar Dictionary")
        self.win.geometry("600x400")
        self.win.resizable(0,0)
        self.win.config(bg='#6098ff')
        self.add_widgets()
        self.database()

    def add_widgets(self):
        self.frame_1 = Frame(self.win,bg='#
        Label1 = Label(self.frame_1,te
        Label1.config(font='AdobeGot
        Label1.grid()
        self.frame_1.pack(expand='r
```

Beyond Basic

Python Programming

Beginner Book

pycharm

python 3

Qpython for android

Videos DVD

Published By mcoder2016.blogspot.com

စာရေးသူ၏အမှာစာ

ဒီစာအုပ်လေးက ကျွန်တော် Python Programming စတင်လေ့လာတဲ့မှတ်စုလေးတွေကို စာပြန်စီလိုအပ်တာလေးတွေထပ်ထည့်ပြီးပြန်လည်စုစည်းပေးထားတာလေးဖြစ်ပါတယ်။အဓိကအားဖြင့် Python ကို စတင်လေ့လာသူများအတွက် ရည်ရွယ်ပါတယ်။စာအုပ်မှာ Video Tutorials များပါဝင်တဲ့ DVD တစ်ချပ်ပါဝင် မှာဖြစ်ပြီး လိုအပ်တဲ့ ဆော့ဝဲလ်များလဲပါဝင်မှာဖြစ်ပါတယ်။Tutorial Videos တွေပါဝင်တယ်ဆိုပေမယ့် စာအုပ် နဲ့တွဲဖက်လေ့လာမှသာ ပိုမိုထိရောက်စွာနားလည်မှာဖြစ်ပါတယ်။စာအုပ်ပါ အကြောင်းအရာများကို အားထည့် ရေးသားထားပါတယ်။ဒါက ကျွန်တော်ဘဝရဲ့ပထမဆုံးသောစာအုပ်လဲဖြစ်ပါတယ်။ဒါကြောင့် စာရေးမချောမွေ့ မှုအားနည်းချက်များရှိနေပါတယ် ဒါကြောင့် မိတ်ဆွေများနားလည်ခွင့်လွှတ်ပေးပါလို့ပြောချင်ပါတယ်။စာရေး သူသည် အတွေ့အကြုံနှယ်သေးတဲ့အတွက် အမှားအယွင်းများရှိခဲ့ရင် မိတ်ဆွေများ နားလည်ပေးပါလို့ အနှုံးညွှန့်တောင်းဆိုပါတယ်။ဒီစာအုပ်လေးဟာ နည်းပညာကိုစိတ်ဝင်စားသူ ညီကိုမောင်နမများ အတွက် တစ်ထောင့် တစ်နေရာကနေ သဲတစ်ပွင့် အုတ်တစ်ချပ်လို့ အထောက်ကူဖြစ်သွားမယ်ဆိုရင် ကျွန်တော် ဝမ်း မြောက်ဝမ်းသာဖြစ်မိမှာပါ။နောက်လည်း mcoder ကနေပြီးမိတ်ဆွေတို့အတွက် Free Tutorials တွေတင်ပြီး ကြိုးစားပါအုန်းမယ်။နောက်ထပ်စာအုပ်တွေလဲ ကောင်းသထက်ကောင်းအောင်ကြိုးစားပြီးတင်ဆက်ပါအုန်းမယ် လို့ပြောရင်းနဲ့ ကျွန်တော့်ရဲ့အမှာစာလေးကိုဒီမှာဘဲ နိဂုံးချုပ်လိုက်ပါတယ်။

၂၀၁၇ခုနှစ်ဘာ (၁၉) ရက်နေ့။

ဘုန်းမြတ်မင်း
aungmoekyaw1995@gmail.com
www.mcoder2016.blogspot.com

မာတိကာ

Python Introduction

Python Programming ရေးဖို့အတွက် computer တစ်လုံးတော့လိုပါတယ်။မိတ်ဆွေတို့ အနေနဲ့ Android ပေါ်မှာ Python ရေးနိုင်တယ်ဆိုပေမယ့် တကယ်တော့ စုံလင်တဲ့ လုပ်ဆောင်ချက်တွေ ရမှာ မဟုတ်ပါဘူး Libraries တိုင်းကိုလည်း ထည့်လို့မရနိုင်ပါဘူး ဒါက အားနည်းချက်ပါ sqlite3 လို database ချိတ်ဆက်မှုအထောက်ပံ့ပေးထားပေမယ့် PC လို လုပ်မရတာတော့အမှန်ပါဘဲဒါကြောင့် Python ရေးမယ်ဆိုရင် အနည်းဆုံး အောက်ပါအချက်လက်နဲ့ညီမျှတဲ့ PC တစ်လုံးတော့ရှိဖို့လိုပါတယ်။

အနည်းဆုံး RAM 1GB CPU Core2Dual နဲ့ Harddisk Space 10GB လောက်လိုပါတယ်။OS အနေနဲ့ Linux သို့မဟုတ် windows 7 , 8, 10 တစ်ခုမဟုတ်တစ်ခုရှိရင်လုံလောက်ပါတယ်။လက်ရှိစျေးကွက်ထဲမှာ windows 10 နဲ့ လာတဲ့ Tablet တွေရှိနေပါပြီ 9 သောင်း 1 သိန်းထက်မပိုပါဘူး ဒီလို Tablet လေးတစ်လုံးလောက် ဂယ်ရင်လေ့လာဖို့လုံလောက်ပါတယ်။Pycharm IDE အတွက်တော့ RAM 1GB 2GB ကတော့အကောင်းဆုံးပေါ့။ 1024x768 resolution ရှိတဲ့ PC ဆိုရင်ရပါပြီ။ဒါကြောင့် ဖုန်းနဲ့ရေးလို့ရပေမယ့် PC နဲ့ရေးတာတော့ အကောင်းဆုံးပါလို့ပြောချင်တယ်။

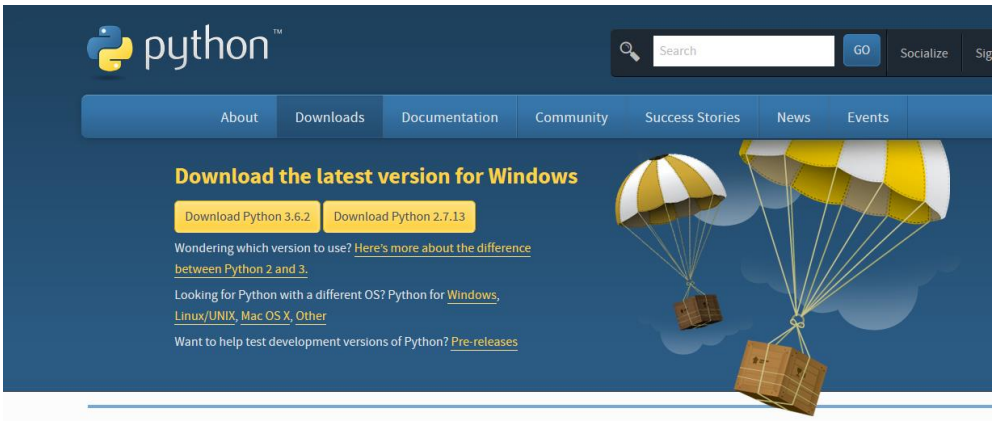
Python က High-level programming language ဖြစ်ပြီး general-purpose အတွက်ဖြစ်ပါတယ်။1991 မှာ Guido van Rossum က ပထမဆုံး released လုပ်ခဲ့တာဖြစ်ပါတယ်။interpreter နဲ့ အလုပ်လုပ်တဲ့ language ဖြစ်ပြီး Python က ဖတ်ဖို့လွယ်ကူပြီး ရိုးရှင်းမှုကို အလေးထားတဲ့ Language တစ်ခု ဖြစ်ပါတယ်။Python မှာ Python တစ်ခုကို Code Line အနည်းငယ်နဲ့ Program တည်ဆောက်နိုင်ပါတယ်(Project ကြီးရင်ကြီးသလောက်တော့ Code များမှာပါ တခြား Programming တွေလောက်မများဘူးပြောတာ)။OOP အထောက်ပံ့ပေးတဲ့ Language တစ်လည်းဖြစ်ပြီးတော့ Powerful ဖြစ်တဲ့ Language လည်းဘဲဖြစ်ပါတယ်။ Python System ဟာ Operation System တော်တော်များများအတွက် အထောက်ပံ့ပေးထားပြီး Python ရဲ့ core system interpreter က C နဲ့ရေးထားတာဖြစ်တဲ့အတွက် Hardware Programming အတွက်ပါ အထောက်ကူပေးနိုင်ပါတယ်။

Python 3.x.x Installation

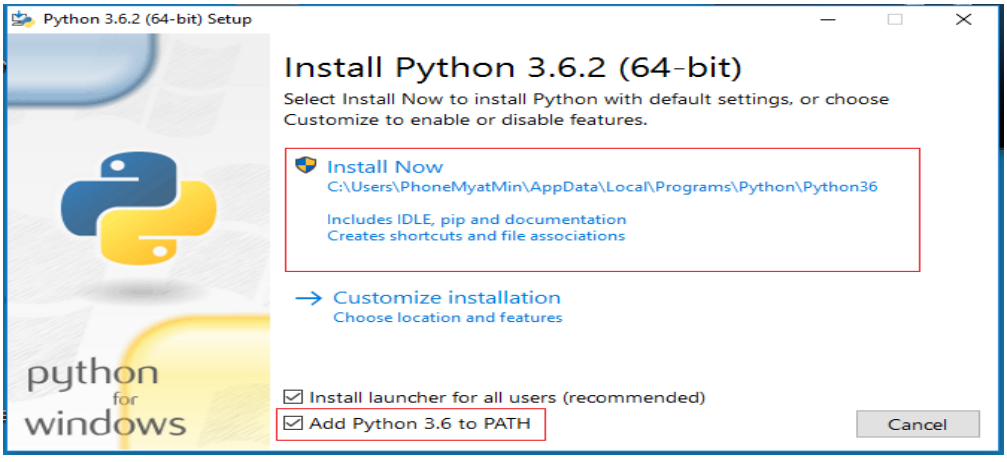
ကျွန်တော် ဒါကိုရေးနေတဲ့အချိန်မှာ Python က version 3.6.2 ရောက်နေပါပြီ ဒါကြောင့် ကျွန်တော်က lasted ဖြစ်တဲ့ Python 3.6.2 ကို Download ဆွဲပါမယ်။

https://www.python.org/downloads/

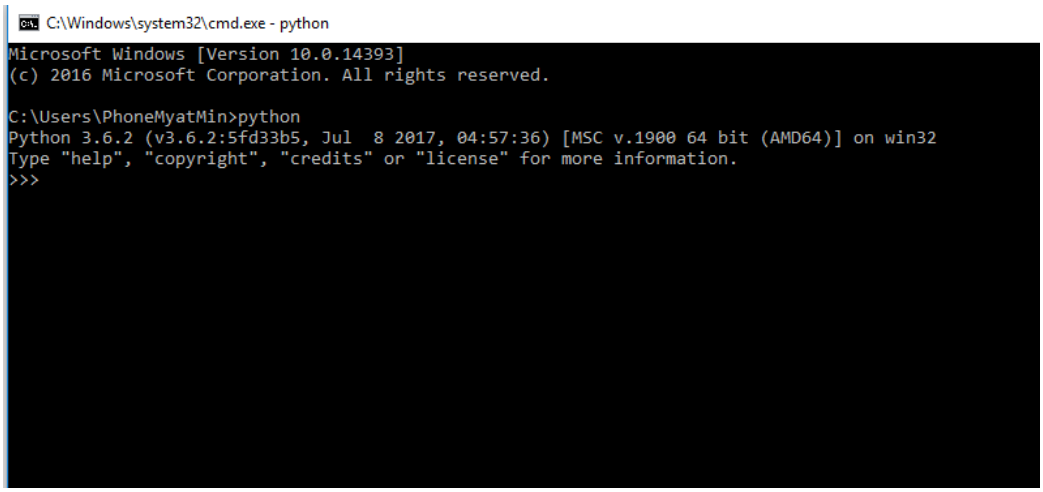
အပေါ်က လင့်ကနေ Python Website ကိုဝင်ပြီး Downloadလုပ်နိုင်ပါတယ်။



ကျွန်တော် အနီလေးနဲ့ ပြထားတဲ့နေရာတွေမှာ အမှန်ခြစ်ပေးပြီး အပေါ်က Install Now ကိုနှိပ်ရမှာဖြစ်ပါတယ်။



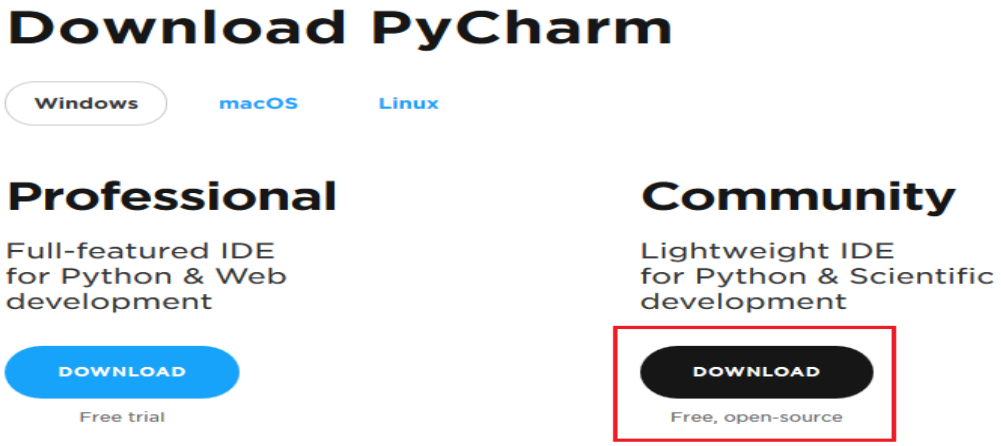
ပြီးရင် ကျွန်တော်တို့ run box (windows Key + R) ကိုခေါ်ပြီး python လို့ရိုက်ကြည့်ပါ Python Shell ပေါ်လာခဲ့ရင် ကျွန်တော်တို့ Python ကို install လုပ်တာအောင်မြင်ပါတယ်။



Python ကို install လုပ်ပြီးပြီဆိုရင် ကျွန်တော်တို့ အသုံးပြုမယ့် IDE(Integrated Development Environment) ကို install လုပ်ပါမယ် ဒါက ဘာလဲဆိုရင် ကျွန်တော်တို့ Programming တွေရေးတဲ့အခါမှာ လွယ်ကူအောင် အထောက်ပံ့ပေးမယ့် Software တစ်မျိုးဘဲဖြစ်ပါတယ်။ကျွန်တော်ကတော့ Python အတွက် Pycharm IDE ကိုအသုံးပြုသွားမှာဖြစ်ပါတယ်။

https://www.jetbrains.com/pycharm/

အပေါ်က လင့်မှာ Download ဆွဲယူနိုင်ပါတယ်။Community Edition ကိုဘဲ Download ဆွဲမှာ ဖြစ်ပါတယ် ဘာကြောင့်လဲဆိုရင် Community Edition က Free ဖြစ်ပါတယ်။



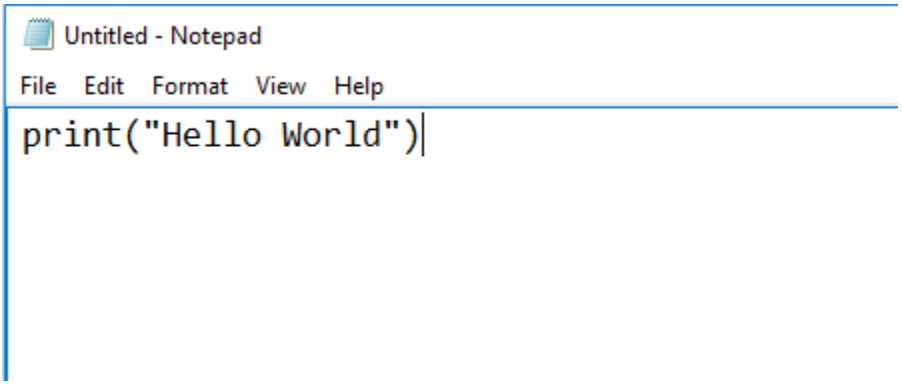
ပြီးရင် Installation လုပ်လိုက်ပါ။ဒါဆိုရင် ကျွန်တော်တို့ Python ရေးဖို့ ပြည့်စုံသွားပြီလို့ပြောလို့ရပါတယ်။ဒါဆိုရင် Python အကြောင်းနည်းနည်းပြောပါ့မယ်။

Python ဆိုတာကတော့

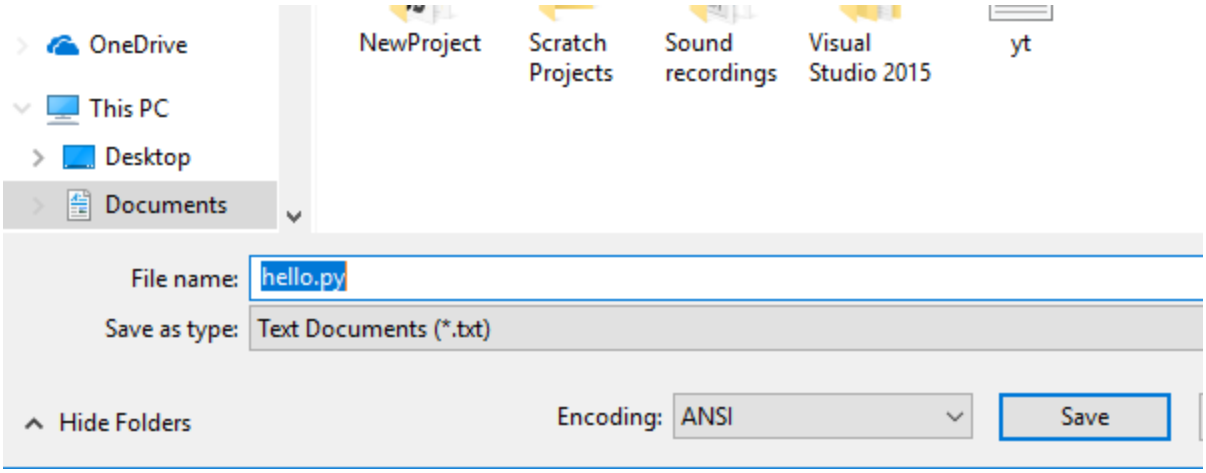
Python ဆိုတာ Programming Language တစ်ခုဘဲဖြစ်ပါတယ်။သူက OOP Language တစ်ခု ဖြစ်ပါတယ်။Python ကို Google,Youtube,Dropbox,Netflix,Hulu တို့မှာ အသုံးပြုထားပါတယ်။ဒါကို ကြည့် ခြင်းအားဖြင့် Python က ခပ်ညံ့ညံ့ Language တစ်ခုတော့မဟုတ်မှန်းသိမှာပါ။ဟုတ်ပါတယ် Python ဟာ Powerful Language တစ်ခုဖြစ်ပါတယ် အခုအခါမှာ Python နဲ့ရေးလို့ရတဲ့ Web Framework တွေထွက်ပေါ် လာပါပြီး ဒါကြောင့် CMS တွေ ecommerce website တွေကို Python အသုံးပြုပြီးတည်ဆောက်နိုင်ပြီ ဖြစ်ပါတယ်။Python ကို Hacker တွေက လည်းအသုံးပြုပါတယ်။လွယ်ကူ ရိုးရှင်းတယ် ပေါ့ပါးတယ် ပြီးတော့ exploit တွေရေးတဲ့အခါမှာ အထောက်ပံများတယ် စတဲ့အချက်တွေကြောင့် Python ကို Hacker တွေအသုံး ပြုကြပါတယ်။နောက်ပြီး Artificial Intelligence ကိုလည်း python က အထောက်ပံ့ပေးပါတယ် ဒါက ဘာလဲ ဆိုရင် မှတ်ဉာဏ်တုစနစ်ဖြစ်ပါတယ် လူမဟုတ်ဘဲ တွေးခေါ်နိုင်တဲ့ Program မျိုးပါ ဥပမာ- Ironman ဇာတ်ကား တွေထဲက Javic လိုမျိုးပါ။နောက်တစ်ခုကတော့အားသာချက်ကတော့ Python Libraries တွေအများ ကြီးရှိတဲ့အတွက် အထောက်ပံ့တွေအများကြီးရရှိပါတယ်။2D game ရေးဖို့အတွက် pygame GUI ရေးဖို့ အတွက် Tkinter,Wx,PyQt စတဲ့ Libraries တွေရှိသလို Web အတွက်တော့ Odoo,Django တို့ရှိပါတယ် အခု ဆို Android အတွက်အထောက်ပံ့ပေးတဲ့ SL4A နဲ့ Kivy တို့လို python အထောက်ပံ့တွေထွက်ပေါ်လာပါ ပြီ။

Chapter 1: Hello World

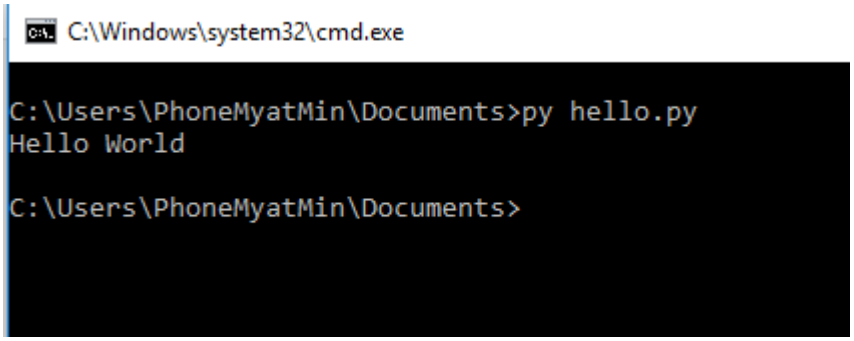
ပထမဆုံးကျွန်တော်တို့ Programming တွေစပြီးလေ့လာတိုင်း လုပ်နေကြဖြစ်တဲ့ Hello World ကို အရင်ဆုံးရေးကြည့်ရအောင် ကျွန်တော်တို့ အခု Program ကိုရေးဖို့အတွက် Notepad ကို အသုံးပြုမှာဖြစ်ပါတယ်။အရင်ဆုံး Notepad ဖွင့်လိုက်ပါ။



Notepad ထဲမှာ print("Hello World") ဆိုပြီးရေးလိုက်ပါပြီးရင် save လုပ်လိုက်ပါ extension အနေနဲ့ .py နဲ့ ဘဲ save လိုက်ပါ။



ပြီးရင် cmd ကိုခေါ်ပြီး ကျွန်တော်တို့ save ထားတဲ့ ဖိုင်ကို py hello.py (or) python hello.py လို့ ခေါ်ပြီး သုံးကြည့်ပါ .py ဆိုတာကတော့ python Program file extension ပါ py သို့မဟုတ် python နဲ့ ခေါ်ပြီး အသုံး ပြုရပါတယ်။



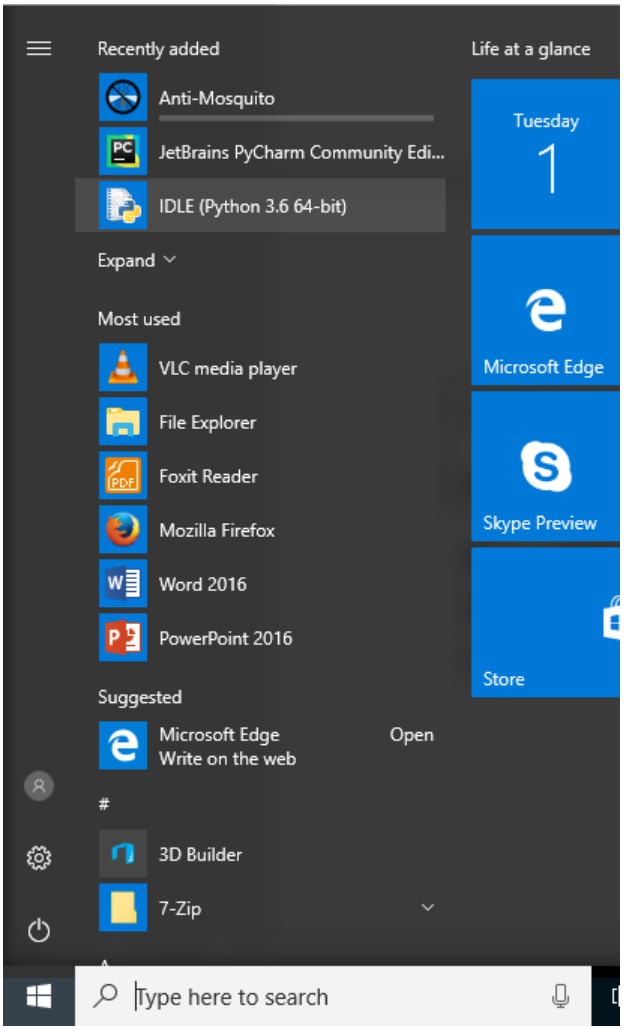
ဒါဆိုရင်တော့ ကျွန်တော်တို့ရေးထားတဲ့ Hello World ဆိုတဲ့ Program လေး Run တာတွေ့ရမှာပါ။print ဆိုတာကတော့ ကျွန်တော်တို့ ထုတ်ချင်တဲ့စာသားတွေ အဖြေတွေကိုဖော်ပြချင်တဲ့အခါမှာ သုံးတဲ့ command ပါဘဲ။

Variables

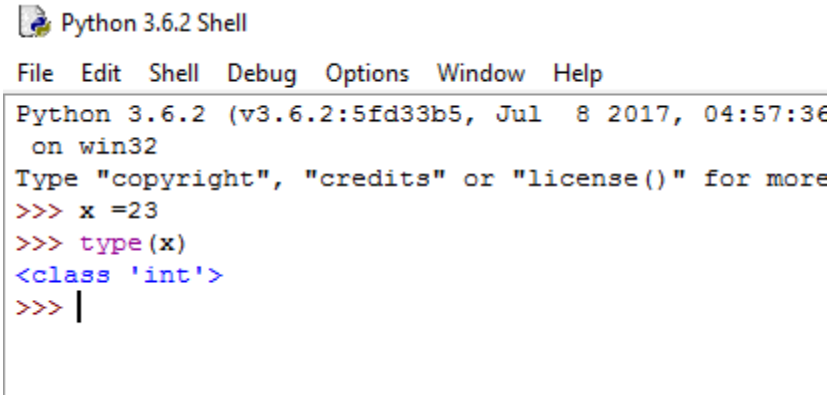
Variables ဆိုတာကတော့ ကျွန်တော်တို့ ကွန်ပျူတာ မန်မိုရီပေါ်မှာ data သိမ်းဆည်းတဲ့အခါနေရာယူလိုက်တဲ့အရာ သို့မဟုတ် data သိုလှောင်တဲ့အခန်းကို variable လို့ခေါ်ပါတယ်။Variables တွေမှာ Data Types တွေပါပါတယ်။Python မှာတော့ တခြား Programming တွေလို သီးသန့် Data အမျိုးအစားတော့ သတ်မှတ်ပေးစရာမလိုပါဘူး။ဒါပေမယ့် Data Types တွေကိုသိထားဖို့လိုပါလိမ့်မယ်။

```
x = 23

print(x)
```

ဒါဟာ variable တည်ဆောက်မှုနမူနာပုံစံဘဲဖြစ်ပါတယ်။Number တစ်ခုကို x ဆိုတဲ့ variable name ထဲမှာ သိုလှောင်လိုက်တာပါ print နဲ့ x ကို အဖြေထုတ်တဲ့အခါမှာ 23 ကိုရပါလိမ့်မယ်။ဒါဘာ Types လည်းသိချင်ရင် တော့ စစ်ဆေးဖို့အတွက် IDLE ဆိုတဲ့ Python Shell Program တစ်ခုကို အသုံးပြုသွားမှာ ဖြစ်ပါတယ်။



ဒါဆိုရင် ကျွန်တော်တို့ အဲဒီထဲမှာ x = 23 ဆိုပြီး variable သတ်မှတ်ပြီးရင် type() ဆိုတဲ့ Function ကို အသုံး ပြု ပြီး ကျွန်တော်တို့ Data Type ကြည့်လို့ရပါတယ်။



23 ကိုသိုလှောင်ထားတဲ့ variable x ရဲ့ Data Type က int ဖြစ်ပါတယ်။Integer ကိုဆိုလိုတာပါ ကိန်းပြည့်အားလုံးကို Integer(int) Data Type ကိုအသုံးပြုရပါတယ်။တခြားသော Programming တွေမှာ Data Type သတ်မှတ်ပေးရပါတယ် ဥပမာ Java မှာဆိုရင် int x = 23; ဆိုပြီးသတ်မှတ်ရပါတယ်။

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

ဒါ Python မှာပါတဲ့ Data Type အမျိုးအစားတွေဘဲဖြစ်ပါတယ်။ကျွန်တော်တစ်ခုခြင်းကိုအသေးစိတ်ရေး သွားမှာဖြစ်ပါတယ်။

What is a String?

String ဆိုတာ စာသား Data အမျိုးစား သိုလှောင်မှုဘဲဖြစ်ပါတယ်။စာသားတွေကိုရေးချင်တဲ့အခါ သို့မဟုတ် စာသားတွေကိုသိမ်းဆည်းချင်တဲ့အခါမှာ String ကိုအသုံးပြုရပါတယ် ဒါမှမဟုတ်သေးပါဘူး Number တွေ ကနေ သို့မဟုတ် တခြားသော Data အမျိုးအစားတွေကနေ စာသားအနေနဲ့ပြောင်းလဲချင်တဲ့

အခါမှာလဲ str() ဆိုတဲ့ String Function တစ်ခုကိုအသုံးပြုပြီးပြောင်းလဲနိုင်ပါတယ်။String ကို အသုံး ပြုမယ်ဆိုရင်တော့ quotes တွေကိုအသုံးပြုရပါတယ်အဲဒါတွေကတော့ single quote(' '),double quote(" "),triple quote("""") တွေကိုအသုံးပြုနိုင်ပါတယ် Characters တွေအသုံးပြုဖို့ဆိုရင် single quote(' ') ကိုအသုံးပြုပြီး အဲဒီ Character ကိုရေးရပါတယ်။စာသားအရှည်တွေရေးဖို့အတွက်တော့ double quote(" ") ကိုအသုံးပြုရပါတယ်။စာပိုဒ်တွေရေးဖို့ဆိုရင်တော့ triple quote("""") သုံးပါတယ် ဒါကထူးခြားပါတယ် တခြားသော Programming Languages တွေမှာ single quote နဲ့ double quote ဘဲအသုံးပြုပါတယ်။ဒီမှာတော့ စာပိုဒ်လိုက်တွေရေးဖို့အတွက် triple quote("""") ကိုအသုံးပြုနိုင်ပါတယ်။

Variable တစ်လုံးတည်ဆောက်လိုက်ရင် RAM ကွန်ပျူတာ မန်မိုရီ ပေါ်မှာ ပမာဏတစ်ခုနေရာယူလိုက်ပါတယ် ဒီ String မှာ ဘယ်လောက်ပမာဏ နေရာယူလဲဆိုတာကိုကျွန်တော်တို့ကြည့်ရအောင်။String တန်ဖိုးကတော့ ရေးထားတဲ့ Characters တွေပေါ်မူတည်ပြီး အကြီးအသေးကွာခြားပါလိမ့်မယ်။

- ASCII Character ဆိုရင် 1 Character ကို 8bit
- UTF-8 Character ဆိုရင် 8bit ကနေ 32 bit(4 Bytes)
- UTF-16 Character ဆိုရင် 16bit ကနေ 32 bit (4 Bytes)

အထိနေရာယူနိုင်ပါတယ်။နေရာယူမှု့ကိုကြည့်ခြင်းအားဖြင့် Variables တွေအများကြီးတည်ဆောက်တာ နဲ့ RAM ပေါ်မှာနေရာယူမှု့များမယ်ဆိုတာ သဘောပေါက်ရပါမယ်။ဒါကြောင့်မလိုအပ်ဘဲအများကြီးတည်ဆောက်အသုံးပြုရင် မိမိ software က ကွန်ပျူတာကိုလေးလံစေနိုင်ပါတယ်။

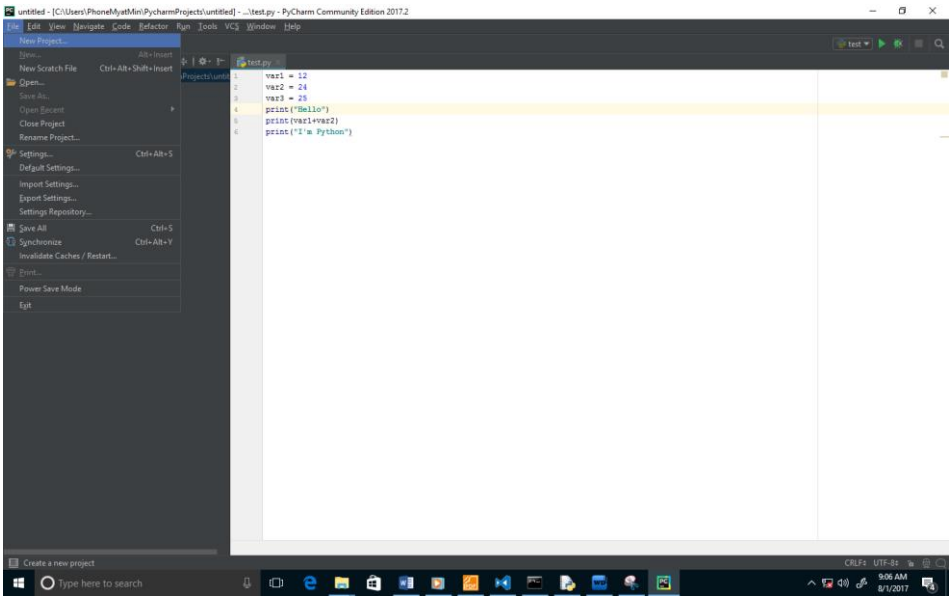
```
>>> name = "AungMoeKyaw"
>>> print(name)
AungMoeKyaw
>>>
```

ဒီမှာ ကျွန်တော်က name ဆိုတဲ့ variable ကိုတည်ဆောက်ထားပြီး အဲဒီထဲမှာ ကျွန်တော်နာမည် ဖြစ်တဲ့ AungMoeKyaw ကို double quote ကိုအသုံးပြုပြီး သိုလှောင်ထားပါတယ်။ပြီးတော့ print() နဲ့ အဲဒီ variable ကို အဖြေပြန်ထုတ်တဲ့အခါမှာ သိုလှောင်မှု့တန်ဖိုးဖြစ်တဲ့ AungMoeKyaw ကိုရစေပါတယ်။အမျိုးအစားကိုသိချင်ရင် type() ကိုအသုံးပြုပြီး စစ်ဆေးနိုင်ပါတယ်။

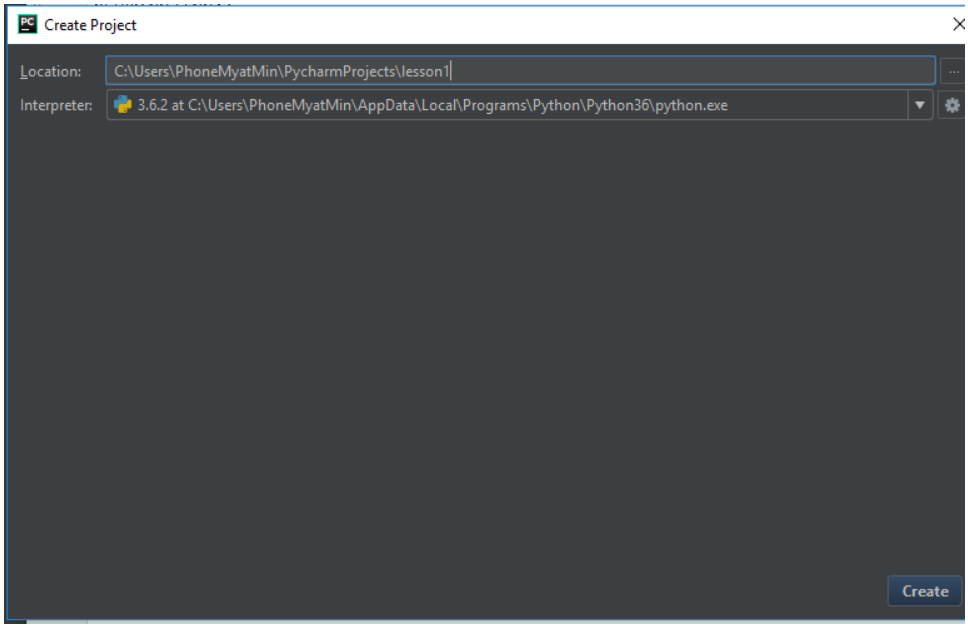
```
name = "AungMoeKyaw"

type(name)
```

Quotes သုံးမျိုးရှိတာ ဘာတွေကွာလဲဆိုတာကိုရေးပြပါမယ် ဒါကိုရေးဖို့အတွက်တော့ Pycharm ကိုအသုံးပြုသွားပါ့မယ်။ဒါဆိုရင် Pycharm IDE ကိုဖွင့်လိုက်ပါပြီးရင် ကျွန်တော်တို့ File ထဲက New Project ကိုယူပြီး ကျွန်တော်တို့ နာမည်တစ်ခုပေးမယ် ကျွန်တော်ကတော့ Lesson1 လို့ဘဲပေးပါတယ်။အဲဒီလို Project တည်ဆောက်ပြီးတဲ့အချိန်မှာ အဲဒီ project folder လေးကို Right Click နှိပ်ပြီး New ထဲက Python File ကို ယူပေးလိုက်ပါ အဲဒီ File နာမည်ကိုတော့ String လို့ပေးလိုက်ပါတယ်။

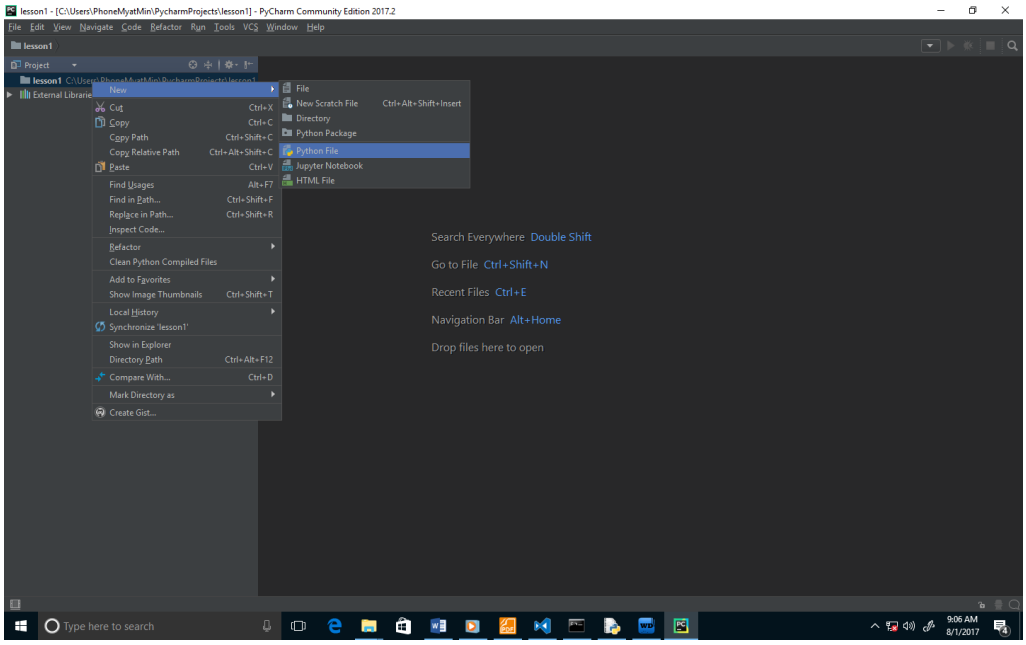
Step 1:



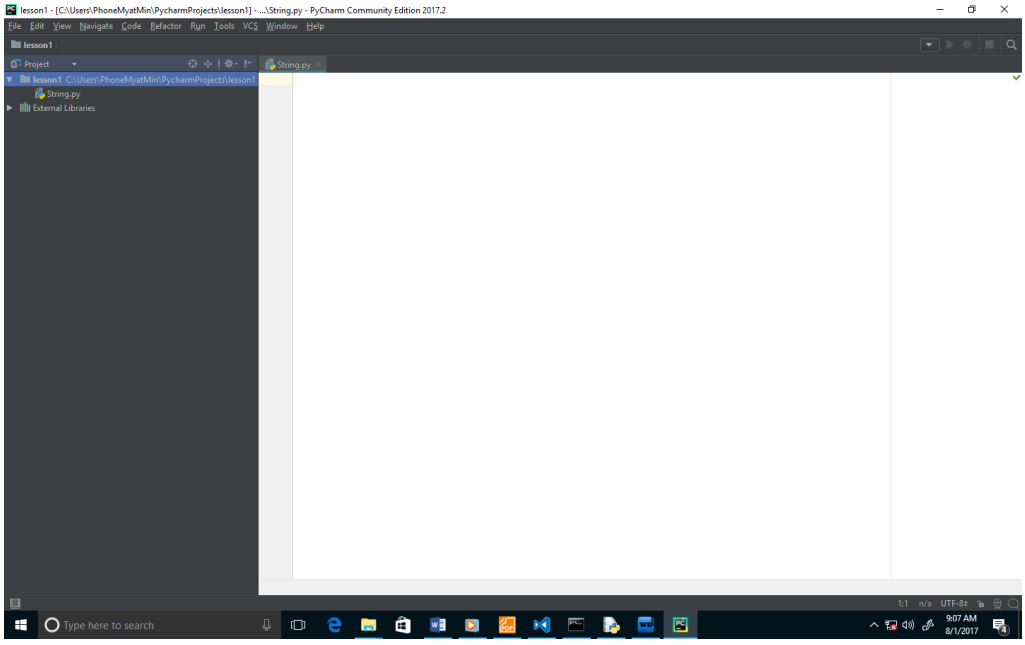
Step 2:



Step 3:


lesson1 - [C:\Users\PhoneMyatMin\PycharmProjects\lesson1] - PyCharm Community Edition 2017.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
lesson1
Project
lesson1 C:\Users\PhoneMyatMin\PycharmProjects\lesson1
External Libraries
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Inspect Code...
Refactor
Clean Python Compiled Files
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Local History
Synchronize 'lesson1'
Show in Explorer
Directory Path Ctrl+Alt+F12
Compare With... Ctrl+D
Mark Directory as
Create Gist...
File
New Scratch File Ctrl+Alt+Shift+Insert
Directory
Python Package
Python File
Jupyter Notebook
HTML File
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open
Type here to search
9:06 AM
8/1/2017

Step 4:


lesson1 - [C:\Users\PhoneMyatMin\PycharmProjects\lesson1] - ...\String.py - PyCharm Community Edition 2017.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
lesson1
Project
String.py
lesson1 C:\Users\PhoneMyatMin\PycharmProjects\lesson1
String.py
External Libraries
Type here to search
9:07 AM
8/1/2017

ဒါဆိုရင် String.py ဆိုတဲ့ ဖိုင်လေးထဲမှာ ကျွန်တော်တို့ Code တွေရေးလို့ရပါပြီ အခုဆိုရင် Code စရေးပါတော့မယ်။

```
name = "AungMoeKyaw"
work = 'Programmer,Student,Founder of mcoder'
home ="""
    No.147,between 2x3 stree
    PyiTharYar
    Meiktikla"""
print(name)
print(work)
print(home)
```

String.py ထဲမှာ ကျွန်တော်က အထက်ပါ Code အတိုင်းရေးလိုက်ပါတယ် ပြီးရင် Run ကြည့်ဖို့အတွက် ကျွန်တော်တို့လုပ်ရမယ်။Run ထဲမှာ Run String ဆိုတာပါတယ် အဲဒါကိုနှိပ်ပြီး Run လို့ရတယ် မဟုတ်ရင်လဲ Shift + F10 နဲ့ Run လို့ရပါတယ် အဖြေကတော့။

```
C:\Users\PhoneMyatMin\AppData\Local\Programs\Python\Python36\python.exe
C:/Users/PhoneMyatMin/PycharmProjects/lesson1/String.py
AungMoeKyaw
Programmer,Student,Founder of mcoder

    No.147,between 2x3 stree
    PyiTharYar
    Meiktikla
```

အထက်ပါ အဖြေအတိုင်းရပါတယ်၊single quote နဲ့ double quote ကအတူတူပါဘဲ ဒါပေမယ့် ဘယ်မှာ ကွာသွားလဲဆိုတာကိုပြပါမယ်။

```
print("I don't like to eat this cake")
```

ဒီလိုစာကြောင်းမျိုးမှာ single quote သာသုံးမယ်ဆိုရင် don't နဲ့ရောထွေးသွားနိုင်ပါတယ် ဒါကြောင့် Programming အရ double quote သာအသုံးပြုသင့်ပါတယ်။

Print() function တစ်ခုထဲကိုသာအသုံးပြုပြီးတော့ စာကြောင်းတွေနောက်တစ်ကြောင်းဆင်းတာ space ခုန်တာ မျိုးလုပ်နိုင်ပါတယ်။အဲဒါကိုကြည့်ရအောင်။

```
print("I don't like to eat this cake\nI need other food \t HaHa ")
```

ဒီမှာ \n နဲ့ \t ကို အသုံးပြုထားပါတယ် Run ကြည့်လိုက်ပါ I need other food က တစ်ကြောင်းဆင်းသွား ပြီး HaHa နဲ့ food ကြားမှာ space ခြားနေပါလိမ့်မယ်။ဒါက \n ကနောက်တစ်ကြောင်းဆင်းဖို့ command ဖြစ် ပြီး နောက်တစ်ခု \t ကတော့ space or tab အတွက် command တစ်ခုဖြစ်ပါတယ်။

အခုနောက်တစ်ခုကတော့ စာသားတွေပေါင်းစပ်တာကိုပြောမှာဖြစ်ပါတယ် String အတွက် အခြေခံ ပုံစံ တွေ ထဲက တစ်ခုဘဲဖြစ်ပါတယ်။

```
first_name = "Aung"
middle_name = "Moe"
last_name = "Kyaw"

print(first_name+middle_name+last_name)
```

ဒီမှာ variable သုံးခုရှိပါတယ် အဲဒါကို သုံးခုပေါင်းပြီးအဖြေထုတ်ဖို့အတွက် ကျွန်တော်က (+) ကိုအသုံးပြုထား ပါတယ် အဖြေကလည်း ကျွန်တော်တို့လိုချင်သလို ၃ လုံးပေါင်းနာမည်ဘဲဖြစ်ပါတယ်။

```
print("aung"+"moe"+"kyaw"+"1995"+"@"+"gmail.com")
```

ဒါကလည်း ကျွန်တော်တို့ String ပေါင်းနည်းတစ်ခုဘဲဖြစ်ပါတယ်။နောက်ထပ် String ကို Format Specifier ကို အသုံးပြုနည်းကိုပြောပါမယ်။

Format Specifer ဆိုတာကတော့ Data Type အမျိုးမျိုးအတွက် အသုံးပြုနိုင်ဖို့ထုတ်ထားတဲ့ Command မျိုးတွေပါဘဲ C programming ကိုလေ့လာဖူးသူဆိုသိမှာပါ။C မှာ အဖြေထုတ်တဲ့အခါ တခြားသော Data အမျိုး အစားတွေနဲ့တွဲဖက်အသုံးပြုဖို့ Format Specifier ကိုအသုံးပြုရပါတယ်။Python မှာ String အတွက် FS ကတော့ %s ပါဘဲ။အခြားသော Data Types တွေအတွက်လဲ အမျိုးမျိုးသော FS တွေရှိပါတယ်။FS ကို အသုံး ပြုဖို့အတွက်

print(“Hi %s” %(“ ”))

ဆိုပြီးအသုံးပြုပါတယ်။

```
print ("Hi %s"%("AungMoeKyaw"))
```

%s နေရာတွေကို နောက်က % နဲ့ ထည့်တဲ့ စာသားတန်ဖိုးတစ်ခုကနေရာယူပါတယ် နှစ်လုံး ထည့်ချင်ရင်လဲ ရပါတယ်။

```
print ("Hi %s %s"%("AungMoeKyaw","Founder of Mcoder"))
```

ဒါဆိုရင်တော့ စာသား တန်ဖိုးနှစ်ခုထည့်ဖို့အတွက် FS နှစ်ခုကိုအသုံးပြုသွားပုံဘဲ ဖြစ်ပါတယ်။

Numbers and Operators

ကိန်း ဂဏန်းတွေအကြောင်းကိုတော့ ကျွန်တော်တို့ငယ်ငယ်ကလေးဘဝ ကတည်းက ကျောင်းသင်ခန်းစာတွေမှာစတင်ရင်းနှီးခဲ့ဖူးပါတယ်။ကိန်း ဂဏန်းတွေနဲ့အတူတူ သချာ်ရောရင်းနှီးခဲ့ပါတယ်။Programming မှာလဲမပါမဖြစ်နဲ့အခြေခံသဘောတရားက ကိန်း ဂဏန်းနဲ့ သချာ်ဆိုင်ရာတွက်ချက်မှု့ဘဲဖြစ်ပါတယ်။

Programming တွေမှာ Numbers တွေကိုဘဲ Data Types အမျိုးမျိုးနဲ့သတ်မှတ်သိမ်းဆည်းလေ့ရှိပါတယ် int(integer),float(float),double(double) စသည်တို့ဘဲဖြစ်ပါတယ် ကွာခြားမှု့ကတော့ int ဆိုတာ ကိန်းပြည့် တန်ဖိုးတွေဖြစ်ပြီး - ရော + ရောသုံးပါတယ်။float နဲ့ double ကတော့ ဒသမကိန်းတွေအတွက် အသုံးပြုတာဖြစ်ပါတယ်။Python မှာတော့ သီးခြားခွဲခြားထားသလိုမရှိပါဘူး အဓိက အသုံးပြုမယ့် Data အမျိုးအစားကတော့ int(integer),float(Float),Imaginary Numbers တို့ဘဲဖြစ်ပါတယ်။

Integer(int)

ကိန်းပြည့်တွေကိုကိုယ်စားပြုတဲ့ Data Type အမျိုးအစားတစ်ခုပါ ကိန်းပြည့်လို့ဆိုရာမှာလည်း အနှုတ်ကိန်းရော အပေါင်းကိန်းရောပါဂင်ပါတယ်။သူ့ရဲ့ Range ကတော့ -2,147,483,648 to +2,147,483,647 ဖြစ်ပါတယ်။ပိုကြီးတဲ့ ကိန်းတွေအသုံးပြုဖို့အတွက် long အမျိုးစား Data Type လိုအပ်ခဲ့ပေမယ့် Python မှာတော့ int တစ်ခုထဲမှာဘဲ အားလုံးဖြစ်စေပါတယ် long ကိုပါအသုံးပြုခွင့်ရထားတဲ့သဘောပါဘဲ။

```
num1 = 100
num2 = 200
print(type(num1))
print(type(num2))
```

ဒါကတော့ variable ရဲ့ Type ကိုကြည့်တာဖြစ်ပါတယ် နှစ်ခုစလုံးက ကိန်းတွေဖြစ်လို့ class 'int' ဆိုတာ ထွက်မှာဖြစ်ပါတယ်။

Float(float)

ဒသမကိန်းတွေကို အသုံးပြုဖို့အတွက် float Data အမျိုးအစားကိုသုံးရပါတယ်။တခြား Programming Language တွေမှာ float ဆိုပြီးသီးသန့်ကြေညာစရာလိုပေမယ့် Python 2.x မှာတော့မလိုပါဘူး ဒါပေမယ့် ဒသမ မှန်းသိအောင် 1.0 ဆိုတာမျိုးလုပ်ပြီးသိုလှောင်ရပါတယ် ဒါမှ အဖြေမှာလဲ ဒသမနဲ့ထွက်လေ့ရှိပါတယ်။အခု Python 3.5.6 မှာတော့တိုက်ရိုက်အသုံးပြုလို့ရပါတယ်။

```
var1 = 7
var2 = 3
var4 = var1 /var2
print(type(var4))
```

ဒီ Program လေးမှာ var1 ရော var2 ရော integer (int) ဖြစ်ပါတယ် ဒါပေမယ့် var4 မှာ var1 / var2 ဆိုတာက var1 ကို var2 နဲ့စားထားတာပါ 7 ကို 3 နဲ့စားတာ မပြတ်ပါဘူး ဒသမကိန်းထွက်ပါတယ် ဒီတော့ var4 က Type က <class 'float'> ဖြစ်ပါတယ် ဒသမ data အမျိုးစားဖြစ်တဲ့ float ကိုသူ့အလိုလိုပြောင်းသွားတာပါ။

Imaginary Number

Imaginary Number ဆိုတာ Python မှာ complex class အမျိုးစားတစ်ခုဖြစ်ပါတယ် complex math မှာ အသုံးပြုဖို့အတွက်ဖြစ်ပါတယ်။နောက်အခန်းတွေမှာ Imaginary Number ဘယ်မှာပါမယ်ဆိုတာတွေ့ရမှာပါ။Imaginary Number တွေရဲ့နောက်မှာ j ဆိုတဲ့ letter ပါပါတယ်။

```
var1 = 7j
print(type(var1))
```

ဒီလို တည်ဆောက်လိုက်တဲ့အခါမှာ class complex လို့အဖြေထွက်လာမှာဖြစ်ပါတယ်။ 7j ဆိုတာ Imaginary Number တစ်ခုဘဲဖြစ်ပါတယ်။Imaginary Number တစ်ခုဟာ တခြားသော nonImaginary Number တွေနဲ့ ပေါင်းလို့ နှုတ်လို့မရနိုင်ပါဘူး။

```
var1 = 7j
var2 = 7j +1
print(var2)
```

7j +1 ဆိုတဲ့အဖြေကိုဘဲထွက်ပါတယ် ဒါဟာ Imaginary Number ဘဲဖြစ်ပါတယ်။

Integer နဲ့ float အတွက် Format Specifiers တွေရှိပါတယ် အဲဒါတွေကတော့ Integer အတွက် %d နဲ့ အသုံး ပြုပြီး float အတွက်တော့ %f နဲ့အသုံးပြုပါတယ်။

```
num1 = 44
print("Number 1 is %d" %(num1))
```

%d ကိုအသုံးပြုပြီး integer ကို String နဲ့ပေါင်းစပ်အဖြေထုတ်တဲ့ပုံဖြစ်ပါတယ်။

```
num1 = 1.2
print("Number 1 is %f" %(num1))
```

ဒါကတော့ float အတွက် အသုံးပြုထားတဲ့ပုံဖြစ်ပါတယ်။ဒီမှာ ဒသမ နောက်မှာ သုညတွေအများကြီးကိုတွေ့ရ မှာပါ အဲဒီသုညတွေအများကြီးမတွေ့ချင်ရင် ကျွန်တော်တို့ ဖြတ်ချင်သလောက် Point ကိုသတ်မှတ်လို့ရပါ တယ်။

```
num1 = 1.2
print("Number 1 is %.1f" %(num1))
```

ဒသမကို တစ်နေရာထဲဖြတ်စေချင်လို့ %.1f လို့ရေးပေးလိုက်ပါတယ် ဒါဆိုရင် ဒသမနောက်မှာ စာလုံး တစ်လုံး ကိုဘဲဖော်ပြပါတော့တယ်။

Basic Math

အခြေခံသချာ်ဆိုတာ အပေါင်း နှုတ် မြောက် စား ဘဲဖြစ်ပါတယ်။Programming မှာလဲ တွက်ချက်မှု့ တွေပြုလုပ်ဖို့အတွက် Math Operator တွေအသုံးပြုပြီး ပေါင်းတာ နှုတ်တာ နဲ့ မြောက်တာ စားတာ တွေပြုလုပ်နိုင်ပါတယ်။

Addition

ကိန်းတွေကိုပေါင်းတဲ့နေရာမှာသုံးပါတယ် ကျောင်းသင်ခန်းစာထဲက အပေါင်းနဲ့အတူတူပါဘဲ အပေါင်း လက္ခာကိုဘဲအသုံးပြုပြီး တန်ဖိုးတွေကိုပေါင်းလေ့ရှိပါတယ်။

```
num1 = 12
num2 = 12
print(num1 + num2)
```

ဒါရဲ့အဖြေက 24 ရပါလိမ့်မယ် ဘာကြောင့်လဲဆိုတော့ num1 နဲ့ num2 ကိုပေါင်းထားပြီး အဖြေထုတ်ထား လို့ ဖြစ်ပါတယ်။

Subtract

ကိန်းတွေကိုနှုတ်တဲ့အခါမှာသုံးပါတယ် ကျောင်းသင်ခန်းစာထဲက အနှုတ်နဲ့အတူတူပါဘဲ အနှုတ်လက္ခာကိုဘဲအသုံးပြုပြီး တန်ဖိုးတွေခြားနားလေ့ရှိပါတယ်။

```
num1 = 58
num2 = 12
print(num1 - num2)
```

ဒါ program ရဲ့အဖြေမှာ 46 ရပါလိမ့်မယ် num1 နဲ့ num2 ရဲ့ခြားနားခြင်း(နှုတ်ခြင်း) ရဲ့ အဖြေက 46 မို့ပါဘဲ။

Multiply

ကိန်းတွေကို မြောက်တဲ့အခါမှာသုံးပါတယ် ကျောင်းသင်ခန်းစာထဲက အမြောက်နဲ့တူပါတယ် လက္ခာအနေနဲ့ (*) ကိုအသုံးပြုပါတယ်။

```
num1 = 36
num2 = 12
print(num1 * num2)
```

ဒီ program ရဲ့အဖြေက 432 ရပါလိမ့်မယ် num1 နဲ့ num2 မြောက်ခြင်းရဲ့အဖြေပါဘဲ။

Division

ကိန်းတွေရဲ့ စားလဒ်ကိုလိုချင်တဲ့အခါမှာ အသုံးပြုပါတယ် သင်္ကေတအနေနဲ့ / ကို အသုံး ပြုပါတယ်။

```
num1 = 36
num2 = 12
print(num1 / num2)
```

အဖြေ 3.0 ရပါလိမ်မယ် စားလိုက်တဲ့တန်ဖိုးက float အဖြစ်ပြောင်းသွားတာကိုလည်းတွေ့ရပါတယ်။

Modulus or Remainder

စားလို့ကျန်တဲ့အကြွင်းကို သိချင်ရင် modulus ကို အသုံးပြုရပါတယ် သင်္ကေတ အသုံးပြုမှုအနေနဲ့ % အသုံးပြုပါတယ်။

```
num1 = 3
num2 = 2
print(num1 % num2)
```

အဖြေ 1 ရပါလိမ့်မယ် 3 ကို 2 နဲ့စားတဲ့အခါ ကျန်တဲ့အကြွင်းဘဲဖြစ်ပါတယ်။

Basic Math Operation

သချာ်ဆိုင်ရာတွက်ချက်မှု့တွေပြုလုပ်တဲ့အခါမှာ Python မှာ လိုက်နာရမယ့်နည်းစနစ်လေးတွေရှိပါတယ် အဲလိုမှမဟုတ်ဘူးဆိုရင် အဖြေမှန်ရမှာမဟုတ်ပါဘူး။အခက်ခဲတော့မရှိစေပါဘူး သာမန် သချာ်လိုပါဘဲ သချာ်မှာ ကျွန်တော်တို့ Operation 2 မျိုး လောက်ပါဝင်လာရင် ဘယ်ဟာကို အရင်လုပ်ရမယ်ဆိုတာကို လက်သည်း ကွင်း ခတ်ပြီး ပြုလုပ်ပေးရပါတယ် ဒီမှာလဲဒီလိုပါဘဲ မဟုတ်ရင် အဖြေမှန်မှာမဟုတ်ပါဘူး။

ဥပမာ - ကျွန်တော်တို့က 2 နဲ့ 2 ကိုပေါင်းပြီး 2 နဲ့မြောက်တဲ့ Program ကိုရေးတဲ့အခါမှာ

```
print(2 + 2 * 2)
```

အဖြေက 6 ရနေပါလိမ့်မယ် တကယ်ဆိုရင် 2 နဲ့ 2 ပေါင်းတာ 4 လေးကို 2 နဲ့မြောက်တဲ့အခါမှာ 8 ရမှာဖြစ်ပါတယ် အဖြေကတော့ လိုချင်တဲ့အတိုင်းမထွက်ပါဘူး ဘာကြောင့်လဲဆိုတော့ ရေးပုံမမှန်လို့ပါ။

```
print((2 + 2) * 2)
```

အရင်လုပ်စေချင်တဲ့အလုပ်ကို ကွင်းခတ်ပေးမှ သာ အဖြေအမှန်ရပါလိမ့်မယ်။

```
num1 = (24 * (8 + 3 + 7.0 + 9))/19
print(num1)
```

ဒီ program က 8 + 3 + 7.0 +9 ကို အရင်ပေါင်းပြီး 24 နဲ့မြောက်လို့ရတဲ့ရလဒ်ကို 19 နဲ့ပြန်စားထားတာပါ။ အဖြေအနေနဲ့ 3 4.10526315789474 ရ ပါတယ် ဒါကို ကျွန်တော်တို့က ဒသမ နောက်မှာ ကိန်းတစ်လုံးဘဲထားခြင်တဲ့ အခါမှာ Format Specifier ကို အသုံးပြုလို့ရပါတယ်။float အတွက် FS က %f ဖြစ်ပါတယ် ။

```
num1 = (24 * (8 + 3 + 7.0 + 9))/19
print("Answer is %.1f" %(num1))
```

ဒါက FS ကို အသုံးပြုပြီး ကျွန်တော်တို့ ဒသမနောက်မှာ တစ်လုံးဘဲထားခဲ့တာပါ %.1f နဲ့အသုံးပြုထားတဲ့အတွက် အဖြေက 34.1 ရပါလိမ့်မယ်။

User Input Function

User က ထည့်သွင်းပေးတဲ့ အရာကို variable ထဲသိုလှောင်တာ သို့မဟုတ် Keyboard က ထည့်သွင်းပေး လိုက်တဲ့တန်ဖိုးတွေကိုလိုအပ်တဲ့အခါမှာ input function ကိုအသုံးပြုပါတယ်။

```
name = input("Enter Your name : ")
```

input function ကို input("....") ဆိုပြီးသုံးရပါတယ် input ရဲ့နောက်က လက်သည်းကွင်းထဲမှာတော့ ကျွန်တော်တို့ user ကိုဖော်ပြချင်တဲ့ စာသားတွေကိုထည့်ပေးရပါတယ် print လိုပေါ့။

```
name = input("Enter Your name : ")
print("Hello %s"%(name))
```

ဒီ Program လေးက ရိုးရှင်းပါတယ် name ဆိုတဲ့ variable ထဲကို input သုံးပြီး user စီကနေ input data လက်ခံလိုက်ပါတယ် ပြီးတဲ့အခါမှာ အဲဒီ variable ကိုဘဲ print နဲ့ Hello ဆိုတဲ့ စာသားနဲ့ တွဲပြီး အဖြေထုတ်လိုက်ပါတယ်။Format Spectifier အနေနဲ့ လက်ခံထားတဲ့ Data က String ဖြစ်လို့ %s သုံးပါတယ်။

ဒါဆိုရင် ဂဏန်းနှစ်ခုပေါင်းတဲ့ Program လေးတစ်ခု တည်ဆောက်ကြည့်ရအောင်ဗျာ

```
num1 = input("Enter First Number:")
num2 = input("Enter Second Number:")
sum  =int(num1) + int(num2)
print(sum)
```

ဒီ Program လေးမှာ num1 နဲ့ num2 ရှေ့မှာ int ဆိုပြီး integer ပြောင်းပေးထားတာတွေ့မှာပါ အဲဒါက ဘာလဲဆိုရင် ကျွန်တော်တို့ input ထည့်လိုက်တဲ့ data တွေကို int ပြောင်းလိုက်တဲ့သဘောပါ ဒါမှလည်း ပေါင်းလို့ရတဲ့အဖြေအစစ်ကိုရမှာဖြစ်ပါတယ်။

Name you can't use and some usage rules

Python မှာ တချို့စကားလုံးတွေက Built-in ပါတဲ့ စကားလုံးတွေနဲ့တိုက်နေတဲ့အခါမျိုးရှိပါတယ် ဒါမျိုးကို ကျွန်တော်တို့က variable name ,function name စတာတွေမပေးသင့်ပါဘူး၊ဒီစကားလုံးတွေကိုလည်း reserve words လို့ခေါ်ပါတယ်။

and,as,assert,break,class,continue,def,del,elif,else,if

except,exec,False,finally,for,from,global, if,import, in,

is, lambda, not,None, or, pass,print,raise,return,try,

while,with,yield

တို့ဖြစ်ပါတယ်။ဒါအပြင်နောက်ထပ်ထပ်ပြောရရင် alphabet မဟုတ်တဲ့ ထူးထူးဆန်းဆန်းစကားလုံးတွေနဲ့ ဂဏန်းတွေက variable နာမည် data နာမည်ပေးလို့မရနိုင်ပါဘူး ဒါပေမယ့် တချို့ခွင့်ပြုထားတဲ့အရာတွေ ရှိပါတယ် အဲဒါတွေက class တချို့နဲ့ Module တစ်ချို့ပါ အဲဒီအကြောင်းကတော့ နောက်အခန်းမှာပါဂျင်မှာဖြစ်ပါတယ်။

Creating and Using Tuple

Tuple တစ်ခုမှာ string တွေရော number တွေရော နှစ်ခုရောနေတာတွေရောပါနိုင်ပါတယ်။Data တွေကိုစုစည်းထားတဲ့ variable အစုကြီးတစ်ခုလိုဖြစ်ပါတယ်။အားသာချက်အနေနဲ့ Types မျိုးစုံပါဂျင်နိုင်တာ ဖြစ်ပြီး immutable ဖြစ်လို့ သူ့ထဲက တည်ဆောက်ပြီးသား data တွေကို overwrite လုပ်နိုင်မှာတော့မဟုတ်ဘူး။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
print(tu)
```

Tuple တစ်ခုကို လက်သည်းကွင်း parenthese (), နဲ့ရေးရပါတယ်။အဲဒီထဲမှာ တန်ဖိုးတွေကို သူ့ရဲ့ အမျိုးစားသတ်မှတ်ချက်အရာ ထည့်ပေးရပါတယ် String ဆိုရင် double quote နဲ့ပေါ့။

Tuple ထဲက တန်ဖိုးတွေကို ကိုယ်လိုချင်တဲ့ပုံစံအတိုင်း ခေါ်ယူသုံးစွဲလို့ရပါတယ်။အဲဒီလို ခေါ်ယူသုံးစွဲဖို့အတွက်ကိုတော့ square bracket [] ကိုအသုံးပြုရပါတယ်။

နောက်တစ်ချက်ကတော့ ကွန်ပျူတာ programming က ကျွန်တော်တို့လူတွေလို 1 ကနေ စပြီးမရေတွက်ပါဘူး ဒီတော့ tu ဆိုတဲ့ Tuple ထဲက တန်ဖိုးတွေမှာ AungMoeKyaw ဆိုတဲ့နာမည်က လူအနေနဲ့ရေတွက်မယ်ဆိုရင် 1 ဖြစ်မှာဖြစ်ပေမယ့် ကွန်ပျူတာ programming ကတော့ 0 ကစရေတွက်ပါတယ်။ဒီတော့ AungMoeKyaw က 0, 22 ကတော့ 1 ဖြစ်ပါတယ်။ဒီတော့ ကျွန်တော်တို့က 22.1.1995 ဆိုတာကိုခေါ်သုံးကြည့်ရအောင်။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
print(tu[2])
```

ဒါဆိုရင် 22.1.1995 ဆိုတဲ့အဖြေကိုရမှာပါ tu[] ထဲမှာ ကျွန်တော်တို့လိုချင်တဲ့ တန်ဖိုးရဲ့ နေရာနာပတ်လေးကိုရေးပေးရပါတယ်။0 ကနေစရေတွက်တော့ 2 နေရာမှာရှိတာကြောင့် tu[2] လို့ခေါ်လိုက်တဲ့အခါမှာ ကျွန်တော် တို့လိုချင်တဲ့အဖြေရပါတယ်။ဒါဆိုရင်နည်းလေး ကွန့်ပြီး အဖြေထုတ်ကြည့်ရအောင်။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
print("Birthday is %s"%(tu[2]))
```

ဒါကတော့ fs သုံးပြီး ကျွန်တော်တို့ပြောချင်တဲ့စာသားနဲ့အဖြေတွဲထုတ်တဲ့ပုံစံဖြစ်ပါတယ်။ဒါဆိုရင် အစဉ် လိုက်အတိုင်းရေးကြည့်ရအောင်ဗျာ။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
print("Name is %s"%(tu[0]))
print("Age is %d"%(tu[1]))
```

```
print("Birthday is %s"%(tu[2]))
print("Founder of %s"%(tu[3]))
```

ဒါဆိုရင် Tuple အသုံးပြုပုံလေးတော့သိပါပြီ immutable ဖြစ်တယ်ဆိုတဲ့ဟာကိုပြောပါမယ်။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
print("Name is %s"%(tu[0]))
print("Age is %d"%(tu[1]))
print("Birthday is %s"%(tu[2]))
print("Founder of %s"%(tu[3]))
tu[0] ="PhoneMyatMin"
```

ဒါဆိုရင် tu[0] နေရာကို PhoneMyatMin ဆိုတဲ့နာမည် Overwrite လုပ်ကြည့်တာပါ ဒါပေမယ့် အပေါ်က အလုပ်တွေအကုန်လုပ်ပြီး အောက်မှာ error တတ်ပါလိမ့်မယ်။

```
    File "C:/Users/PhoneMyatMin/PycharmProjects/untitled/test.py", line 6, in <module>
      tu[0] ="PhoneMyatMin"
    TypeError: 'tuple' object does not support item assignment
```

အထက်ဖော်ပြထားတဲ့ Error တတ်ပါလိမ့်မယ် tuple object does not support item assignment ပါတဲ့ ဒါဆိုရင် ကျွန်တော်ပြောတဲ့ immutable ဖြစ်တာကိုမြင်ရမှာပါ။

နောက်တစ်ခုကတော့ Tuple ထဲမှာ items ဘယ်နခုရှိလဲဆိုတာကြည့်လို့ရတဲ့ len() ဆိုတဲ့ Functions လေး အကြောင်းပြောချင်ပါတယ်။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
print(len(tu))
```

len() ဆိုပြီး လက်သည်းကွင်း ထဲမှာ ကျွန်တော်တို့ရဲ့ Tuple Object ဖြစ်တဲ့ tu လေးကိုထည့်ပေးရင် tu ထဲမှာ ရှိတဲ့ items အရေတွက်ကိုရမှာဖြစ်ပါတယ်။items တွေ ကို sequence လို့ခေါ်ပါတယ်။ အခုနောက်ထပ် ပြောမှာကတော့ multidimensional Tuple အကြောင်းဘဲဖြစ်ပါတယ်။Multidimensional ဆိုတဲ့အတိုင်းဘဲ Tuple ထဲမှာ Tuple တွေပါဝင်နေတာဖြစ်ပါတယ်။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
a = ("Meiktial","Yangon","Mandalay","Kalay","PyiOoLwin")
b = ("Myanmar","US","Korea","Thai")
multi = (tu,a,b)
print(multi)
```

ဒီမှာဆိုရင် tu ရယ် a ရယ် b ရယ်ဆိုတဲ့ Tuple တွေကို multi လို့ခေါ်တဲ့ Tuple ထဲကို ထည့်လိုက်တဲ့ အခါမှာ multidimensional Tuple အဖြစ် multi က ရောက်သွားတယ်။Tuple Sequance ထဲမှာ နောက်ထပ်

Sequance တွေပါတဲ့အမျိုးအစားပေါ့။ဒါဆိုရင် ဒီထဲက တန်ဖိုးတွေကို ဘယ်လိုခေါ်သုံးမလဲဆိုတာကိုထပ်ပြောပါမယ်။

```
tu = ("AungMoeKyaw",22,"22.1.1995","mcoder")
a = ("Meiktial","Yangon","Mandalay","Kalay","PyiOoLwin")
b = ("Myanmar","US","Korea","Thai")
multi = (tu,a,b)
print(multi[0][0])
```

ခေါ်ပြီးသုံးတဲ့အခါမှာ multi[][] ဆိုပြီးခေါ်သုံးရပါတယ် ပထမ [] က multi ထဲမှာရှိတဲ့ sequance ကို ကိုယ်စား ပြုတာပါ ဒီတော့ multi ထဲက 0 ဆိုတော့ tu ပါ ။ဒုတိယ [] ကတော့ 0 နေရာမှာရှိတဲ့ Tuple ရဲ့ sequance ကိုပြောတာပါ ဒီတော့ multi ရဲ့ 0 နေရာက tu ဖြစ်ပါတယ်နောက်တစ်ကွင်းမှာလဲ 0 တော့ tu ရဲ့ 0 ဖစ်တဲ့ AungMoeKyaw ကိုဖော်ပြပေးပါတယ်။

Lists – Changeable Sequences of Data

နောက်တစ်ခုဆက်ပြီးပြောပြမှာကတော့ List ဖြစ်ပါတယ်။Tuple လိုဘဲ အသုံးပြုနိုင်ပါတယ် သို့သော် သူက Tuple လို immuable မဟုတ်ဘဲ Changeable ဖြစ်ပါတယ် သူထဲက တန်ဖိုးတွေကို ပြောင်းလဲလိုရပါတယ် ဖျက်ထုတ်တာမျိုးလုပ်နိုင်ပါတယ်။List တွေကို square brackets တွေနဲ့တည်ဆောက်ပါတယ်။Tuple မှာလိုဘဲ sequance တွေကို 0 ကနေ စတင်ရေတွက်ရပါတယ်။

```
adress = ["N0.137","2x3 Zizawar Road","PyiTharYar",1121]
print(adress)
```

ခေါ်ယူသုံးစွဲတဲ့အခါမှာ Tuple လိုဘဲ square bracket နဲ့ ခေါ်သုံးရပါတယ်။

```
adress = ["N0.137","2x3 Zizawar Road","PyiTharYar",1121]
print("Home Number %s"%(adress[0]))
```

Multidimensional အနေနဲ့ အသုံးပြုတာလဲ Tuple နဲ့အတူတူပါဘဲ။

```
adress = ["N0.137","2x3 Zizawar Road","PyiTharYar",1121]
country = ["Myanmar","India","Unite State"]
multi = [adress,country]
print(multi)
```

ဒါဆိုရင် Changeable ဖြစ်တဲ့အကြောင်းကို ကျွန်တော်တို့ဆက်သွားရအောင်။Overwrite လုပ်နိုင် Remove လုပ်နိုင်ပြီး ထပ်ပေါင်းထည့်နိုင်တဲ့ Functions တွေ List မှာ ပါဝင်လာပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ လက်ရှိရှိနေတဲ့ sequance ထဲက item တစ်ခုကို Overwrite လုပ်ကြည့်ပါမယ်။

```
students = ["Maung Maung","Kyaw Kyaw","Aung Aung","Hla Hla","Mya Mya"]
print(students)
print("OverWrite Example")
students[0] = "Maung Aung"
print(students)
```

Students ဆိုတဲ့ List တစ်ခုထဲမှာ Sequance items တွေရှိပါတယ် အဲဒီ Sequance items တွေကို ကျွန်တော်တို့က စိတ်ကြိုက် overwrite လုပ်ပြုပြုနိုင်ပါတယ် အဲဒီလို ပြုပြင်ဖို့အတွက် item တွေရဲ့နေရာ index နာပတ်တော့လိုပါတယ် အပေါ်က program မှာ index 0 ဖြစ်တဲ့ Maung Maung ကို ကျွန်တော်တို့က Maung Aung နဲ့အစားထိုးသွားပါတယ် ရေးပုံကတော့ students[0] = "Maung Aung" ဆိုပြီးဖြစ်ပါတယ်။ဒါဆို ရင် ကျွန်တော်တို့နောက်တစ်ခုလုပ်ကြည့်မယ် user input function ကိုသုံးပြီးတော့ ကျွန်တော်တို့ Overwrite လုပ်မယ့်နေရာရွေးမယ်၊Overwrite လုပ်ချင်တဲ့နာမည်ကိုရေးမယ် ပြီးရင် ဖော်ပြချင်တဲ့နာမည်ကိုဖော်ပြမယ် ဒီလို Program လေးရေးမယ်။

```
students = ["Maung Maung","Kyaw Kyaw","Aung Aung","Hla Hla","Mya Mya"]
print(students)
print("OverWrite Example")

print("Enter Your Overwrite Place:")
ind = int(input())

print("Enter Overwrite Name:")
name = input()

students[ind] = name

print("---------------------------------")
sname = int(input("Choice Number:"))

print(students[sname])

print("---------------------------------")
print("All Name Showing")
print(students)
```

ဒီ Program မှာကျွန်တော်တို့ students ဆိုတဲ့ List တစ်ခုရှိတယ် အဲဒီထဲမှာ items 5 ခုပါဝင်ပါတယ်။အဲဒီ items 5 ခုကို အရင်ဆုံး print ထုတ်ပြလိုက်တယ်။ပြီးတော့ Overwrite Example ဆိုတဲ့စာသားကို print ထုတ်တယ် ပြီးတဲ့ အခါမှာ Enter Your Overwrite Place ဆိုတဲ့ Message လာမယ် အဲဒါက ကျွန်တော်တို့ နာမည် ပြောင်းချင်တဲ့ item ရဲ့ နေရာ index number ကိုရေးခိုင်းတာ အဲဒီမှာ 0 ကိုရေးလိုက်တယ် ဆိုရင် MaungMaung ဆိုတဲ့ 0 နေရာမှာရှိတဲ့item လေးကိုပြောင်းမယ်ဆိုတဲ့သဘောဖြစ်ပါတယ် ပြီးတော့ အဲဒီ Maung Maung နေရာမှာ ပြောင်းမယ့်နာမည်ရေးခိုင်းပါတယ် အဲလို user input တွေကို input() function သုံး

ပြီး ind နဲ့ name ဆိုတဲ့ variable တွေထဲမှာသိုလှောင်ပါတယ် ပြီးတော့ List ရဲ့ students[ind] = name ဆိုပြီး index နေရာရယ် ပြောင်းမယ့်နာမည်ရယ်ကိုရေးပါတယ်။ဒါဆိုရင် ကိုယ်ပြောင်းလိုက်တဲ့ index နေရာမှာ ကိုယ် ပြောင်းချင်တဲ့နာမည်ပြောင်းသွားပါလိမ့်မယ်။နောက်တစ်ခု input တောင်းတာကတော့ sname ဂ နရာမှာဖြစ်ပါတယ် ကိုယ်ကြိုက်တဲ့ item ကို ခေါ်ပြီး print ထုတ်ကြည့်ဖို့ဖြစ်ပါတယ်။print(students[sname]) ဆိုပြီးသုံးထားပါတယ်။ပြီးတဲ့အခါမှာ List တွေအကုန်လုံးကို အဖြေပြန်ပြပါတယ်။ဒီမှာတစ်ခုပြောချင်တာ sname ရယ် ind ရယ် ရဲ့ input မှာ int(input()) ဆိုပြီးသုံးတာရှိပါတယ် ဒါက ဘာလဲဆိုတော့ ကျွန်တော်တို့ထည့်ပေး လိုက်တဲ့ input ကို int() ကိန်းပြည့် integer ပြောင်းထည့်လိုက်တာဖြစ်ပါတယ် ဘာကြောင့်လဲဆိုတော့ index numbers တွေက integer သုံးတာဖြစ်လို့ပါဘဲ။ဒါဆိုရင် ကျွန်တော်တို့ List ထဲက item တွေကို ဘယ်လို overwrite လုပ်ရလဲသိသွားပါပြီ။နောက်တစ်ခုကတော့ List ထဲကို နောက်ထပ် ထပ်ပြီး item တွေပေါင်းထည့်တဲ့ နည်းဖြစ်ပါတယ်။

```
students = ["Maung Maung","Kyaw Kyaw","Aung Aung","Hla Hla","Mya Mya"]
print(students)
print(len(students))
print("Adding item Example")
students.append("Ko Ko")
print(students)
print(len(students))
```

List တွေကို item တွေပေါင်းထည့်ချင်ရင် append() နဲ့အသုံးပြုရပါတယ် List ကိုအရင်ဆုံးရေးပြီး students.append("Name") ဆိုပြီးပြောင်းထည့်ရပါတယ်။အပေါ်က program မှာ မြင်သာအောင်လို့ len() ကိုပါထည့်ပြီး အရေတွက်ပြထားပါတယ် students.append("Ko Ko") နဲ့ပေါင်းထည့်ပြီး ပြန်ပြီး အဖြေ ထုတ်ကြည့်တဲ့အခါမှာ List ထဲမှာ Ko Ko လည်းပိုလာသလို len() ထဲမှာလဲ item ပိုလာတဲ့အတွက် index number ၁ ခုတိုးလာပါတယ်။

```
['Maung Maung', 'Kyaw Kyaw', 'Aung Aung', 'Hla Hla', 'Mya Mya']

5

Adding item Example

['Maung Maung', 'Kyaw Kyaw', 'Aung Aung', 'Hla Hla', 'Mya Mya', 'Ko Ko']

6
```

အဖြေက အထက်ပါပုံစံအတိုင်းဘဲဖြစ်ပါတယ်။append() function က List ထဲမှာ item တစ်ခုချင်းတိုးတဲ့ Function ဖြစ်ပါတယ်။List ထဲကို items တွေအများကြီးတခါထဲတိုးလိုတဲ့အခါမှာ extend() နဲ့အသုံးပြုနိုင်ပါတယ်။တခါထဲ item တွေအများကြီးထပ်ထည့်နိုင်ပါတယ်။

```
students = ["Maung Maung","Kyaw Kyaw","Aung Aung","Hla Hla","Mya Mya"]
print(students)
print(len(students))
print("Adding item Example")
students.extend(["Aung Ko Ko","Kyaw Zin","Win Win","Aye Maung"])
print(students)
print(len(students))
```

အပေါ်က Program မှာ List ထဲကို နောက်ထပ် နာမည် 4 ခုကို exten လုပ်ထည့်လိုက်တဲ့ပုံပါ။မူရင်း students ဆိုတဲ့ ထဲမှာ items 9 ခုဖြစ်သွားပါတယ်။ဒါပေမယ့် ဒါက program run နေတုန်းဘဲရပါတယ် save ထားလို့မရပါဘူး Database အသုံးပြုမှ Save ထားနိုင်ပါလိမ့်မယ်။နောက်သင်ခန်းစာတွေမှာ Database အသုံး ပြုတာတွေပါလာမှာပါ။items တွေပေါင်းထည့်တာပြီးတဲ့အခါမှာ ကျွန်တော်တို့ မလိုချင်တဲ့ items တွေကို remove လုပ်တဲ့နည်းပြောပါမယ် remove() ကိုဘဲအသုံးပြုပါတယ်။

```
students = ["Maung Maung","Kyaw Kyaw","Aung Aung","Hla Hla","Mya Mya"]
print(students)
print(len(students))
print("Remove Item Example")
students.remove("Maung Maung")
print(students)
print(len(students))
```

ကျွန်တော်တို့ remove လုပ်ချင်တဲ့ item နာမည်ကို remove ရဲ့ နောက်က လက်သည်းကွင်း ထဲမှာ ထည့်ပေးရပါတယ် ဒါဆိုရင် အဲဒီ item ကို remove လုပ်သွားပါလိမ့်မယ်။

Dictionaries – Groupings of Data indexed by Name

Dictionaries တွေမှာ Key နဲ့ Value ဆိုပြီး နှစ်ခုပါဝင်လာပါတယ်။Value ဆိုတဲ့ တန်ဖိုးတွေကို အသုံးပြုလိုတဲ့အခါမှာ Key ကိုခေါ်ရင် အဲဒီ Key ရဲ့ Value တွေရပါတယ်။ဖွဲ့စည်းပုံအရတော့ Lists တို့ Tuples တို့ နဲ့ ဆင်တူပါတယ်။Dictionaries တွေကို Curly braces ထဲမှာ တည်ဆောက်ရပါတယ်။Key နဲ့ Value ကြားမှာ colon(:) ထည့်တည်ဆောက်ပါတယ်။

```
dic ={"Name":"AungMoeKyaw","Age":"23","Phone":"09962758431","Blood":"O"}
```

ဒါက Dictionaries တည်ဆောက်ပုံဖြစ်ပါတယ်။ရှေ့က Name ,Age,Phone,Blood တွေက Key ဖြစ်ပြီး နောက်က တော့ Value ဖြစ်ပါတယ်။Value ကိုခေါ်ချင်ရင် သူနဲ့သက်ဆိုင်တဲ့ Key ကိုခေါ်သုံးရပါတယ်။

```
dic ={"Name":"AungMoeKyaw","Age":"23","Phone":"09962758431","Blood":"O"}
key = input("Enter Key.....")
print(dic[key])
```

ဒီ program မှာ key ဆိုတဲ့ variable တစ်ခုကိုတည်ဆောက်ပြီး dic[key] ဆိုပြီး print ထုတ်ထားပါတယ် ဒါက ကျွန်တော်တို့က value တွေကို key ကို user input လုပ်ပြီး ထည့်တဲ့သဘောပါဘဲ။Dictionaries တွေကို နောက် တစ်နည်း တည်ဆောက်လို့ရပါသေးတယ်။

```
dirc ={}
dirc["Name"]="AungMoeKyaw"
dirc["Age"] = "23"
dirc["Phone"] ="09962758431"
dirc["Blood"] = "O"
```

အပေါ်က တည်ဆောက်ခဲ့တဲ့ နည်းနဲ့အတူတူဘဲဖြစ်ပါတယ်။

Dictionaries မှာ မရှိတဲ့ Key ကို အသုံးပြုပြီး ရှာခဲ့ရင်တော့ Error တတ်မှာဖြစ်ပါတယ်။

```
dirc ={}
dirc["Name"]="AungMoeKyaw"
dirc["Age"] = "23"
dirc["Phone"] ="09962758431"
dirc["Blood"] = "O"

print(dirc["Address"])
```

```
C:\Users\PhoneMyatMin\AppData\Local\Programs\Python\Python36\python.exe
C:/Users/PhoneMyatMin/PycharmProjects/untitled/test.py

Traceback (most recent call last):

  File "C:/Users/PhoneMyatMin/PycharmProjects/untitled/test.py", line 7, in <module>

    print(dirc["Adress"])

KeyError: 'Adress'
```

KeyError ဆိုပြီး Address မရှိတဲ့အကြောင်း ကို Error တတ်လာပါတယ်။

Dictionaries ရဲ့အဓိက ထူးခြားချက်က သူမှာပါတဲ့ Key တွေကိုလဲကြည့်လို့ရတယ်။Value ဘယ်နှခုပါလဲ ဆိုတာ လဲကြည့်လို့ရပါတယ်။

```
dirc ={}
dirc["Name"]="AungMoeKyaw"
dirc["Age"] = "23"
dirc["Phone"] ="09962758431"
dirc["Blood"] = "O"

hud = dirc.keys()
print(type(hud))
print(hud)
print(len(hud))
print(list(hud))
print(tuple(hud))
```

ဒီ program က dirc ဆိုတဲ့ Dictionaries ထဲက Key တွေကိုကြည့်ဖို့တည်ဆောက်ထားတာပါ ပထမဆုံး hud ဆိုတဲ့variable တစ်ခုတည်ဆောက်လိုက်တယ် ပြီးတော့ dirc.keys() ဆိုတဲ့ Function နဲ့ hud ထဲကို dirc ရဲ့ Key တွေကိုထည့်လိုက်တယ် ပြီးတော့ print() ကို type(hud) နဲ့ type ကိုစစ်ဆေးတယ် အဲလိုစစ်ဆေးတော့ <class 'dict_Keys'> ဆိုပြီးဖော်ပြပါလိမ့်မယ်။ပြီးတော့ hud ဘဲ အဖြေထုတ်ကြည့်တော့ dict_keys(['Name', 'Age', 'Phone', 'Blood']) အဖြေထွက်ပါတယ်။အဲဒါကို Key ဘယ်နှလုံးရှိလဲဆိုတဲ့ အရေတွက်သိချင်တော့ len() ကိုအသုံးပြုပြီးစစ်ဆေးပါတယ် 4 ဆိုပြီး အဖြေထွက်ပါတယ်။နောက်ပြီး print(list(hud))ကတော့ List ပြောင်းတာဖြစ်ပါတယ်။tuple(hud) ကတော့ Tuple ပြောင်းတာဖြစ်ပါတယ်။

ဒါဆိုရင် Value ကိုလိုချင်တယ်ဆိုရင် values() ဆိုတဲ့ method ကိုအသုံးပြုရမှာပါ။လက်တွေ့ပြပါမယ်။

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}

value = info.values()
print(type(value))
print(len(value))
print(value)
print(list(value))
print(tuple(value))
```

ဒါက values() method ကိုအသုံးပြုပြီးခေါ်ယူသုံးစွဲတာဖြစ်ပါတယ် type နဲ့ value ကိုစစ်ဆေးတယ် အရေအတွက်ကိုသိချင်တဲ့အတွက် len() နဲ့စစ်ဆေးတယ် ပြီးတော့ value ကို ဒီအတိုင်း print ထုတ်တယ် နောက်ပြီး list ပြောင်းပြီးထုတ်တယ် tuple ပြောင်းပြီးထုတ်တယ်။

ဒါဆိုရင် Key တွေကိုဖော်ပြချင်တဲ့အခါမှာ keys() method ကိုသုံးတယ် value တွေကို ဖော်ပြချင်တဲ့အခါမှာ values() method ကိုသုံးတယ် ဒါဆိုရင် Key ရော Value ရောဖော်ပြလို့ရတဲ့ method ရှိလားဆိုရင် ရှိပါတယ် အဲဒါကို items() method လို့ခေါ်ပါတယ် အသုံးပြုပုံကတော့။

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}

item = info.items()
print(type(item))
print(item)
print(list(item))
print(tuple(item))
```

ဒါ Key ရော Value ရောကို ကြည့်ရှုလို့ရပါတယ် Tuple တွေ List တွေအဖြစ်ပြောင်းလို့ရပါတယ်။

Updating Dictionary

Dictionary တစ်ခုကို update လုပ်နိုင်ပါတယ် update လုပ်တယ်ဆိုတာ key နဲ့ value တစ်စုံကို ထပ်ထည့်တာ,ရှိနေပြီးသား item တွေကို ပြင်ဆင်တာ နဲ့ ရှိနေပြီးသား item တွေကို ဖျက်ပြစ်တာမျိုးကို ပြောတာပါ။

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}
#Adding new Key and Value
info["Work"] = "Programmer"

print(info)
print(info["Work"])
```

ဒါက Key နဲ့ Value တစ်ခု ပေါင်းထည့်တဲ့ ပုံစံဖြစ်ပါတယ် info ဆိုတဲ့ Dictionary ထဲကို ကျွန်တော်က Work ဆိုတဲ့ Key နဲ့ Programmer ဆိုတဲ့ Value ကိုထပ်ထည့်ပါတယ်။ဒါဆိုရင် နောက်တစ်ကာ ရှိပြီးသား item ကို ပြင်ဆင်တာလုပ်ကြည့်ပါမယ်။

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}
#Modifying an existing item
info["Name"] = "Phone Myat Min"
print(info["Name"])
```

Name နေရာမှာ နဂိုရှိတဲ့ AungMoeKyaw ကို Phone Myat Min ဆိုတာနဲ့ Modify လုပ်လိုက်တာပါ။ဒါဆိုရင် Dictionary item တွေကို ဘယ်လို ဖျက်ရမယ်ဆိုတာကိုဆက်လက်ပြီးလေ့လာသွားပါမယ်။

Delete Dictionary Elements

Dictionary ထဲက Elements တွေကို တစ်ခုချင်းဘဲဖြစ်စေ အားလုံးကိုဖြစ်စေ ဖျက်ပစ် နိုင်ပါတယ် method တွေအသုံးပြုပုံကို ဖော်ပြပါမယ်။

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}
#Deleting Dictionary item
del info["Name"]
print(info)
```

ဒါက item တစ်ခုခြင်းစီ Key တွေကို ခေါ်ပြီး ဖျက်နိုင်ပါတယ် del ဆိုတဲ့ method ကိုအသုံးပြုပြီး item ကိုဖျက် နိုင်ပါတယ်။

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}
#Deleting Dictionary item
info.clear()
print(info)
```

clear() method ကတော့ အားလုံး ကိုဖျက်ပစ်တာဖြစ်ပါတယ်။Clear ဆိုတဲ့အတိုင်း Dictionary ထဲမှာ ဘာမှ မကျန်တော့ပါဘူး

```
info = {"Name":"AungMoeKyaw","Age":"23","Blood":"O","Ph":"09962758431"}
#Deleting Dictionary
del info
print(info)
```

del ဆိုပြီး Dictionary ကိုထည့်တဲ့အခါမှာ Dictionary တစ်ခုလုံးကိုဖျက်ပစ်တဲ့နေရာမှာသုံးပါတယ် items တွေ ကို clear လုပ်တာမဟုတ်ဘဲတစ်ခုလုံးကိုဖျက်ပစ်တာပါ ဒါကြောင့် Dictionary မရှိတာကြောင့် ခေါ်သုံးတဲ့ အခါမှာ Error တတ်ပါလိမ့်မယ်။

```
C:\Users\PhoneMyatMin\AppData\Local\Programs\Python\Python36\python.exe
C:/Users/PhoneMyatMin/PycharmProjects/untitled/test.py

Traceback (most recent call last):

  File "C:/Users/PhoneMyatMin/PycharmProjects/untitled/test.py", line 4, in <module>

    print(info)

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

NameError ဆိုပြီး info is not defined ဆိုပြီးတတ်ပါတယ် ဘာကြောင့်လဲဆိုတော့ အပေါ်မှာ del နဲ့ ဖျက်ခဲ့ ပြီး ဖြစ်လို့မရှိတော့လို့ဖြစ်ပါတယ်။

နောက်တစ်ခုကတော့ Dictionary ၂ ခုကို ပေါင်းစပ်ပြီး တစ်ခုထဲကို Update လုပ်တဲ့အခါမှာ သုံးတဲ့ Function တစ်ခုကိုပြောပါမယ်။

```
dic = {"name":"AungMoeKyaw","Phone":"09962758431","No.":"137"}
update_dic ={"Blood":"O","Hobby":"Reading","Work":"Hardware/Software developer"}

print("Print Before Update Dictionary")
print(dic)
print(update_dic)
print("==========================================================")

dic.update(update_dic)

print("Print After Update Dictionary")
print(dic)
```

ဒီမှာ ကျွန်တော်က Dictionary 2 ခု တည်ဆောက်ထားပြီး မူရင်း Dictionary ကို dic လို့နာမည်ပေးထားပြီး နောက်တစ်ခုကိုတော့ update_dic လို့နာမည်ပေးထားပါတယ်။နောက်ပြီး အရင်ဆုံး Print Before Update Dictionary လို့ အဖြေထုတ်ပြီး print နဲ့ dic နဲ့ update_dic နှစ်ခုကိုအဖြေထုတ်ပါတယ်။ပြီးတဲ့အခါမှာတော့ .update() function ကိုအသုံးပြုပြီးတော့ dic ဆိုတဲ့ Dictionary ထဲကို update_dic ဆိုတဲ့ Dictionary ကို update လုပ်ပါတယ်။ပြုလုပ်ပုံကတော့ dic.update(update_dic) ဆိုပြီးဖြစ်ပါတယ်။ပြီးတဲ့အခါမှာ Print After Update Dictionary ဆိုပြီး အဖြေတစ်ခုထုတ်တယ် ပြီးတော့ update လုပ်ထားတဲ့ dic ကိုအဖြေ ပြန်ထုတ်လိုက်ပါတယ်။

Slicing Sequences

Slicing ဆိုမှတော့ ဖြတ်ထုတ် လှီးထုတ်ပြစ်မှာပေါ့ဗျာ ဘာတွေကို လှီးထုတ်မှာလဲ ဆိုတော့ Sequences တွေ ကိုဖြစ်ပါတယ်။Sequences တွေက Lists, Tuples, String တို့ဖြစ်ပါတယ်။ဒါတွေကို လှီးထုတ်တယ်ဆိုတာကို Slicing လုပ်တယ်လို့ခေါ်ပါတယ်။ဘယ်လိုလှီးထုတ်မယ်ဆိုတာကိုကြည့်ရအောင်။

```
tuple_Slicing=("123","name","Man","Women","girl")
print(tuple_Slicing[2:4])

list_Slicing=["car","airplane","UFO","Boat","Fighter"]
print(list_Slicing[1:4])

String_Slicing = "Hello World,I am Python Developer"
print(len(String_Slicing))
print(String_Slicing[1:6])
```

အပေါ်မှာ Tuple တစ်ခုဖန်တီးလိုက်ပါတယ် ကျွန်တော်တို့ items တွေက 0 ကနေ စတင်ရေတွက်တာ ဖြစ်တဲ့အတွက် ၄ ခုရှိတယ်လို့ပြောလို့ရပါတယ်။အဲဒီမှာ Slicing လုပ်ချင်တဲ့အခါမှာ tuple_Slicing[] လို့ရေးပြီး စပြီး

ဖြတ်မယ်နေရာရယ် အဆုံးမှတ်ရယ်ကိုရေးပေးရပါတယ်။ဒါကို Slicing လုပ်တယ်ခေါ်တာပါ။တစ်ခုထူးခြားတာ ရှိပါတယ် စတင်မှတ်သတ်မှတ်ပေးတဲ့ စလုံးကနေ နောက်ဆုံးမှတ်ကြားက ယူတာဖြစ်တဲ့အတွက် စတင်မှတ်သာဖော်ပြပြီး ဆုံးမှတ်ကြားလို့ပြောတဲ့အတွက် ဆုံးမှတ်ပါမှာမဟုတ်ပါဘူး။အခုဆို tuple_Slicing[2:4] ဖြစ်တဲ့အတွက် 0 ကနေစရေတွက်ဘူး index number 2 က Man ဖြစ်ပါတယ် index number 4 ကတော့ girl ပါ။ 2:4 ဆိုတော့ စမှတ် 2 ဖြစ်တဲ့ Man ကနေစပြီး women ထိဖြစ်ပါတယ် ဆုံးမှတ်ဖြစ်တဲ့ 4 ကတော့ girl ကတော့ ပါစရာမလိုပါဘူး။ဒါကြောင့် Man,Women ဆိုတဲ့အဖြေထွက်လာမှာပါ ကိုယ်လိုချင်တဲ့ကနေရာလေးဘဲကွပ်ပြီး လိုးထုတ်လို့ရတဲ့အတွက် Slicing လို့ခေါ်ပါတယ်။List အဲလိုပါဘဲ List တစ်ခုတည်ဆောက်ထားပြီး item 5 ခုရှိပါတယ် ဒါပေမယ့် 0 ကနေ စတင်ရေတွက်တော့ 0-4 ဘဲရှိပါတယ်။အဲဒီထဲက မှ စမှတ် 1 ကနေ 4 ထိ လိုးထုတ်မယ်ပြောလိုက်တော့ 1 ဖြစ်တဲ့ airplane ကနေစပြီးပါဂင်မှာပါ 4 က ဆုံးမှတ်ဖြစ်လို့ 4 တော့ပါဂင်မှာမဟုတ်ပါဘူး။String အတွက်လဲအဲလိုပါဘဲ အခု String မှာစလုံးပါဂင်တာ 33 လုံးရှိပါတယ်။len() နဲ့ကြည့်ထားတာတွေ့ရမှာပါ 0 ကနေစတင်ရေတွက်မယ်ဆိုရင်တော့ 0 to 32 ဖြစ်ပါတယ် အဲဒီထဲကမှ 1 ကနေ 6 ကိုဖြတ်ထုတ်မယ်လို့ပြောတော့ 1 က e ဖြစ်ပါတယ် 6 က Space ခြားလေးဖြစ်ပါတယ် ဒီတော့ ello လို့အဖြေထွက်လာမှာပါ။

.pop() Functions

Items တွေရဲ့ index number ကိုရေးပြီးတစ်ခါခေါ်သုံးပြီးတာနဲ့ List ထဲကဖျက်ထုတ်လိုက်တဲ့ Function ဖြစ်တဲ့ .pop() ကိုကျွန်တော်အခုပြောမှာဖြစ်ပါတယ်။

```
list_pop =["12","342","4y2","99","1245","145"]
pop_item = list_pop.pop(1)
print(pop_item)
print(list_pop)
```

ဒီမှာကြည့်လိုက်ပါ list တစ်ခုတည်ဆောက်ထားပြီး item က 0 ကနေဆို 5 အထိပါဂင်ပါတယ်။အဲဒီ item ကို list_pop.pop(1) ဆိုပြီး item index number 1 ကိုရွေးပြီး variable တစ်ခုထဲမှာသိုလှောင်ပါတယ် အဲဒီ variable ကို အဖြေထုတ်တဲ့အခါမှာ index item ဖြစ်တဲ့ 342 ကိုရမှာဖြစ်ပါတယ် ဒါပေမယ့် List ကို အဖြေ ပြန်ထုတ်လိုက်တဲ့အခါမှာတော့ 342 က ပါတော့မှာမဟုတ်ပါဘူး။

Removing Duplicates

တူနေတဲ့ item တွေကိုဖယ်ထုတ်ပြစ်ဖို့အတွက် ကျွန်တော်တို့က set() ဆိုတဲ့ method ကို အသုံးပြုနိုင်ပါတယ်။အဲဒီလိုအသုံးပြုတဲ့အခါမှ item တွေက အစဉ်လိုက်အတိုင်းမဖြစ်တော့တာတော့ သတိထားရမှာဖြစ်ပါတယ်။

```
list_2 =["a","b","c","d","e","f","3","a","a","100"]
var = set(list_2)
print(var)
```

ဒီမှာဆိုရင် list တည်ဆောက်ထားပါတယ်။အဲဒီထဲမှာ a တွေ သုံးလုံးလောက်ထပ်နေတာပါဂင်ပါတယ် အဲဒါကို variable တစ်ခုတည်ဆောက်ပြီး set(list_2) ဆိုပြီး set() method သုံးပြီးထည့်လိုက်တဲ့အခါမှာ variable ထဲမှာ ထပ်နေတဲ့ items တွေကိုဖန်ပြီးသား List တစ်ခုရမှာဖြစ်ပါတယ်။

Making Decisions

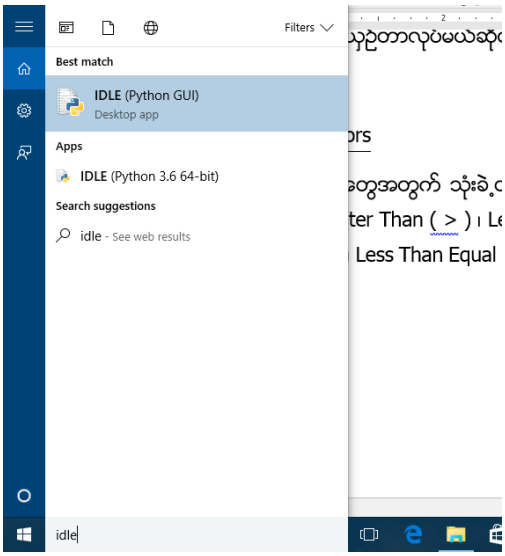
အရှေ့မှာ programs ငယ်လေးတွေကိုတည်ဆောက်ခဲ့ပေမယ့် တကယ့် program တွေ ကို တည်ဆောက် တဲ့အခါမှာသုံးတဲ့ Making Decisions လို့ခေါ်တဲ့ if,elif,if...else နဲ့ Loop တွေ ဖြစ်တဲ့ while, for စတဲ့ Decisions ကိစ္စတွေအခုအခန်းမှာဖော်ပြသွားတော့မှာဖြစ်ပါတယ်။

Special Type and Comparing Values

Special Type တွေဆိုတာ Boolean လို့လဲခေါ်လို့ရတဲ့ True or False တွေဖြစ်ပါတယ်။မှန်ကန်တဲ့ အခြေနေတွေမှာဆိုရင် အဖြေပြန်ထုတ်ပေးတာ True ဖြစ်ပြီး မှားတဲ့အခြေနေတွေမှာဆိုရင် အဖြေပြန်ထုတ် ပေး တာ False ဖြစ်ပါတယ်။အဲဒါအပြင် True နဲ့ True ကိုနှိုင်းယှဉ်တဲ့အခါမှာ True ပြန်ရပြီး True နဲ့ False ကို နှိုင်းယှဉ်ရင် False ရပါတယ်။ဒီလိုဘဲ False နဲ့ False ကို နှိုင်းယှဉ်တဲ့အခါမှာတော့ True ရပါတယ် ဒါက ဘာနဲ့ တူလဲ ဆိုတော့ Logic လိုပါဘဲ။ကျွန်တော်တို့ 10 တန်း Modern Physics မှာ Logic သင်ရသလိုနဲ့ အတူတူ ဖြစ် ပါတယ်။ဒါဆိုရင် နှိုင်းယှဉ်တာလုပ်မယ်ဆိုတော့ Comparing Operators တွေကိုစပြီးပြောမှရပါလိမ့်မယ်။

Comparing Operators

နှိုင်းယှဉ်ခြင်းတွေအတွက် သုံးခဲ့တဲ့ Operators တွေကို ကျွန်တော်တို့ ၈ တန်းလောက်က တည်းက ရင်းနှီးနေတာပါ။Greater Than (>) ၊ Less Than (<)၊Equal to (==) ၊ Not Equal to (!=) ၊ Gerater Than Equal (>=) ၊ Less Than Equal (<=) တို့ဖြစ်ပါတယ်။ဒါဆိုရင် Special Types တွေကို Comparing လုပ်ကြည့်ရအောင်။IDLE ဖွင့်ပြီးစမ်းသက်ပါမယ်။



ပြီးရင် စတင်ပြီး နှိုင်းယှဉ်ပါမယ်။

```
>>> True == True

True

>>> True == False

False

>>> True != False

True

>>> False == False

True

>>> False == True

False

>>> False != True

True
```

ဒီမှာကြည့်လိုက်ပါ။True နဲ့ True နဲ့ ညီတယ်ဆိုတာ မှန်တဲ့အတွက် True ရပါတယ်။True နဲ့ False နဲ့ တူတယ် ဆိုတာ မဟုတ်တဲ့အတွက် False ပါ။True နဲ့ False နဲ့မတူဘူးဆိုတာအတွက်တော့ တကယ်မတူ ဘူးဖြစ်တဲ့အတွက် True ဖြစ်ပါတယ်။ဒီလိုဘဲ False နဲ့ False နဲ့တူတဲ့အတွက် True ဖြစ်ပါတယ်။ဒီသဘော တရားက မရှုပ်ပါဘူး တကယ်တော့ ရိုးရှင်းပါတယ်။နောက်တစ်ခုရှိသေးတာက Python မှာ True ကို 1 လို့ သတ်မှတ်ပြီး False ကို တော့ 0 နဲ့ သတ်မှတ်ပါတယ်။

```
>>> True == 1
True
>>> False == 0
True
```

ဒီတော့ True က 1 ဖြစ်ပြီး False က 0 ဖြစ်ပါတယ်။IDLE မှာ နှိုင်းယှဉ်ကြည့်တဲ့အခါမှာ True == 1 က မှန်ကန်တာ ကို တွေ့ရပြီး False == 0 က လည်း မှန်ကန်တာတွေ့ရမှာပါ။ဒါဆိုရင် ကိန်းဂဏန်းတွေ နဲ့ Comparing Operators တွေတွဲဖက်အသုံးပြုကြည့်ရအောင်ဗျာ။

```
>>> 1 > 2
False
>>> 1 < 2
True
>>> 1 >= 2
False
>>> 1 <= 2
True
>>> 1 == 2
False
>>> 1 != 2
True
```

ကိန်းဂဏန်းတွေယှဉ်ကြည့်တာဖြစ်ပါတယ် 1 > 2 မှာ 1 က 2 တဲ့ကြီးတယ်ဆို တာ မဟုတ်တဲ့ အတွက် False ဖြစ်ပါတယ်။ 1 < 2 မှာတော့ 1 က 2 အောက်ငယ်တယ်ဆိုတဲ့ အခြေနေက မှန်တဲ့အတွက် True ဖြစ် ပါ တယ်။ 1 >= 2 ဆိုတဲ့ အတိုင်း 1 က 2 နဲ့ကြီးပြီးညီတယ်ဆိုတာမဟုတ်တဲ့အတွက် False ပါ 1 <= 2 ဒီအခြေနေမှာ တော့ 1 က 2 နဲ့ ငယ်ပြီးညီနိုင်တဲ့အတွက် True ဖြစ်ပါတယ်။ 1==2 မှာတော့ သိတဲ့အတိုင်း 1 က 2 နဲ့မတူတဲ့(မညီ) တဲ့အတွက် False ပါ။ 1 != 2 မှာတော့ မညီဘူး မတူဘူး ဖြစ်လို့ True ဖြစ်ပါတယ်။ကျွန်တော်ရေးထားတဲ့ စာအရ ဖတ်တဲ့အခါမှာ ရှုပ်ကောင်းရှုပ်နေနိုင်ပါတယ်။ဒါပေမယ့် သေချာပြန်စဉ်းစားရင်ရှင်းသွားမှာပါ။ဒါဆို ကျွန်တော်တို့ if , elif , if elif else... တို့ကို စပြီးလေ့လာတော့မှာ ဖြစ်ပါတယ်။

IF(အကယ်၍များ)

အင်္ဂလိပ်စာ IF နဲ့ဆင်တူပါတယ်။အကယ်၍ များ ကျွန်တော်က Developer တစ်ယောက် ဖြစ်ရင် Development လုပ်ငန်းစဉ်တွေကိုလုပ်ရမယ်ဆိုတာမျိုးပါ။အခြေနေတစ်ခုကိုစစ်ဆေးတယ် အဲဒီအခြေနေ ဖြစ်တယ်(မှန်တယ်) True ဆိုရင် ခိုင်းထားတဲ့လုပ်ငန်းစဉ်လုပ်မယ်၊မဟုတ်ဘူးဆိုရင်မလုပ်ဘူးဆိုတာ မျိုး ဖြစ်ပါတယ်။Let! Write Code...

```
work = "Programmer"

if (work == "Programmer"):
    print("He Write Code!!!")
```

ဒီမှာ work ဆိုတဲ့ variable ကို IF နဲ့ စစ်ဆေးထားတာတွေ့ရမှာပါ။work ထဲမှာ Programmer လို့သိုလှောင်ထားပါတယ်။အဲဒီမှာလုပ်မယ့်အလုပ်က if (work == "Programmer"): ဆိုပြီးဖြစ်ပါတယ်။အကယ်၍များ work က Programmer သာဖြစ်မယ်ဆိုရင် ဘာလုပ်မလဲဆိုတာ (:) နောက်မှာရေးရပါတယ်။print("He Write Code!!") ဖြစ်ပါတယ် အကယ်၍များ work က Programmer ဖြစ်မယ်ဆိုရင် He Write Code!! ဆိုတဲ့ Message ပေါ်လာမှာပါ။work က Programmer ဖြစ်နေလို့ IF ထဲက အလုပ်က လုပ်ပါလိမ့်မယ်။IF နောက်မှာ Condition ပါပါတယ် IF နောက်က စစ်ဆေးတဲ့ Condition သာမှန်မယ်ဆိုရင် IF က အလုပ်လုပ်မှာပါ။

```
condition = 1 < 2

if (condition):
    print("That Condition is True")
```

ဒီမှာဆိုရင် condition ဆိုတဲ့ variable ထဲမှာ အခြေနေတစ်ခု နှိုင်းယှဉ်ထားပါတယ်။အဲဒီအခြေနေကတော့ 1 < 2 ဖြစ်ပါတယ်။ဒီအခြေနေမှန်ကန်လို့ condition ထဲမှာ True ဖြစ်ပါလိမ့်မယ် အဲဒီ variable ကိုဘဲ if ထဲမှာ စစ်ဆေးစေပါတယ်။True ဖြစ်တဲ့အခါမှာ IF ထဲက အလုပ်လုပ်ပါတယ်။

IF..... ELSE(အကယ်၍များ မဟုတ်ခဲ့လျှင်)

ဒီမှာတော့ IF (အကယ်၍)များအပြင် ELSE ဆိုတဲ့ (ထိုအပြင်) ဆိုတာပါဖက်လာပါတယ်။ဒီဟာရဲ့အလုပ်လုပ်ပုံကတော့ IF (အကယ်၍) များထဲက စစ်ဆေးတဲ့ Condition က မမှန်ခဲ့ရင် Else ဆိုတဲ့ မဟုတ်ခဲ့လျှင်ထဲက အလုပ်ကိုလုပ်ပါလိမ့်မယ်။ဥပမာ- စားသောက်ဆိုင်မှာ ကျွန်တော်က VIP ခန်းယူထားပြီး Waiter ကိုမှာထားမယ် ဒီနာမည်နဲ့သူငယ်ချင်းလာရင် VIP ဝိုင်းကိုလွှတ်လိုက်ပါပေါ့။အဲတော့ Waiter က IF နဲ့စစ်ပါတယ် လာသမျှလူကို ကျွန်တော်သူငယ်ချင်းဟုတ်မဟုတ် IF ဟုတ်ခဲ့ရင် VIP ဝိုင်းကိုသွား Else မဟုတ်ခဲ့ဘူးဆိုရင် တခြားနေရာပေါ့။အဲဒီသဘောတရားပါဘဲ။

```
friend = "AungAung"
```

```
if(friend == "MaungMaung"):
    print("Welcome Sir From My VIP Room")

else:
    print("Welcome Sir From My Coffee Shop")
```

ဒီမှာကြည့်လိုက်ရင် ကျွန်တော်က ပေးထားတဲ့ သူငယ်ချင်းနာမည်က AungAung ဖြစ်ပါတယ်။ဒါပေမယ့် IF စီ ကိုလာတာက MaungMaung ဖြစ်ပါတယ်။အဲဒီတော့ VIP ROOM ကို မသွားရပါဘူး Else ထဲက ရိုးရိုး ပိုင်း ကိုဘဲ ထိုင်ရပ့ါမယ်။IF ထဲမှာ စစ်ဆေးထားတဲ့ Condition က မမှန်ဘဲ (True) မဟုတ်ဘဲ (False) ဖြစ်နေမယ်ဆိုရင် Else ထဲကိုသွားရပါတယ် ဘာကြောင့်လဲဆိုတော့ IF က စစ်ဆေးထားတဲ့ Condition မှန်မှအလုပ်လုပ်လို့ ဖြစ် ပါတယ်။

IF ..... ELIF ..... ELSE

IF တွေအပြီးနောက်တစ်ခုတိုးလာပြီ :) ဒါပေမယ့် စိတ်မပူပါနဲ့ IF မှာတုန်းက ကျွန်တော်တို့က အခြေ နေတစ်ခုကိုဘဲ စစ်ဆေးလို့ရခဲ့ပါတယ်။IF ထဲက အခြေနေတစ်ခု မမှန်ရင် Else ထဲကို တန်းသွားရပါတယ်။ELIF ကတော့ အခြေနေတွေထပ်ခါထပ်ခါ စစ်ဆေးလို့ရတဲ့ IF အပွားလေးတွေဘဲဖြစ်ပါတယ်။စောစောက ဥပမာ နဲ့ ပြောရရင် သူငယ်ချင်း Aung Aung တစ်ယောက်ထဲမဟုတ်ဘဲ KoKo MgMya ThiThi စတဲ့ လူတွေဆိုလဲ VIP ပိုင်းမှာဘဲခေါ်ချင်တယ်ဆိုရင်တော့ အခြေနေတွေအများကြီးစစ်ဆေးရတော့မှာပေါ့ဒါမျိုးဆိုရင် ELIF ကို အသုံးပြုနိုင်ပါတယ်။

```
friend = "AungAung"

if(friend == "MaungMaung"):
    print("Welcome Sir From My VIP Room")

elif(friend == "AungAung"):
    print("Welcome Sir From My VIP Room")

elif(friend == "ThiThi"):
    print("Welcome Sir From My VIP Room")

elif(friend == "MaungMya"):
    print("Welcome Sir From My VIP Room")

else:
    print("Welcome Sir From My Coffee Shop")
```

အခုဆိုရင် if တစ်ခုတည်းနဲ့မဟုတ်တော့ဘဲ elif တွေနဲ့ AungAung , ThiThi , MaungMya တို့ကိုပါ ထပ်စစ်ဆေးပါတယ်။အဲဒီလူလေးဦးမဟုတ်မှ else ထဲကိုရောက်ပါတယ်။ဒီတော့ IF ... ELIF ... ELSE တွေကို သဘောပေါက်မယ်လို့မျှော်လင့်ပါတယ်။ဒါပေမယ့် လိုအပ်တဲ့ Program တွေကိုထပ်ပြီးလေ့ကျင့်သွားမှာ ဖြစ် ပါတယ်။

အခု ကျွန်တော်က Exam Marks အကြောင်း IF ELIF ELSE သုံးပြီး ဥပမာအနေနဲ့ ရေးထားတာ ရှိပါတယ် အဲဒီ ဟာလေးကိုဥပမာပြပေးမှာဖြစ်ပါတယ်။

```
print("""
########################
# Exam Marks           #
# IF ELIF ELSE TESTING #
# Example Program      #
########################
""")
print("Enter Exam Marks :")
mark =int(input())

if (mark < 40 ):
    print("You Fail Exam")
elif(mark <=55):
    print("You Pass Exam But Marks Not Good")

elif(mark <=70):
    print("You Pass Exam But Marks  is Normal")
elif(mark <=80):
    print("You Pass Exam and Marks is Good")
elif(mark <= 100):
    print("You Pass Exam and Marks is Excellent")
else:
    print("Input Error!")
```

ဒီ Program လေးမှာ Print နဲ့စာသားတွေ Blah Blah ဆိုပြီးထုတ်ထားတာတော့ အထူးပြောစရာမလိုတော့ ပါဘူး အောက်မှာ mark ဆိုတဲ့ variable တစ်ခုတည်ဆောက်ပြီး input() ရယူထားပါတယ်။input() ရဲ့ ရှေ့ မှာ int(input()) ဆိုပြီးလုပ်ထားတာက input() method က လက်ခံသမျှဟာ String လို့ခေါ်တဲ့ စာသားတန်ဖိုး တွေဖြစ်ပါတယ် ကျွန်တော်တို့က ဒီမှာ integer ဖြစ်တဲ့ int ကိုလိုချင်တဲ့အတွက် int() သုံးပြီး ပြန်ပြောင်းထားတာဖြစ်ပါတယ်။ပြီးတော့ အဲဒီ mark ကို IF နဲ့ စစ်ဆေးထားပါတယ်။ဒီမှာတစ်ခုသိစေချင်တာက program တိုင်းက အပေါ်ကနေအောက်ကို တစ်ဆင့်ခြင်းအလုပ်လုပ်တာဖြစ်လို့ စစ်ဆေးတဲ့အချိန်မှာ လဲ တဆင့်ခြင်းစစ်ဆေးရပါမယ်။ဒါကြောင့် mark ကို 40 အောက်ကစ စစ်ဆေးပါ 40 အောက်ဆို စာမေးပွဲကြမယ် ဆိုတဲ့ Message တတ်လာပါမယ်။အဲဒီကမှတစ်ဆင့် 40 ကနေ 55 အတွင်းက You Pass Exam But Marks Not Good ဆိုတဲ့ Message လာမှာပါ။အဲဒီ 55 ရဲ့အထက်ကနေ 70 ထိက You Pass Exam But Marks is Normal လို့ Message လာမယ်။70 နဲ့အထက်ကနေ 80 အထိကတော့ You Pass Exam But Marks is Good လို့ ဖြစ်မယ် 80 နဲ့အထက်ကနေ 100 ထိကတော့ You Pass Exam and Marks is Excellent ပါ 100 ကျော် တာတွေစာသားတွေစတဲ့ ယုတ္တိမရှိတာတွေအတွက်တော့ Else ထဲမှာ Input Error အနေနဲ့ပြမှာပါ။

နောက်ထပ် Program တစ်ခုကတော့ IF …… ELIF……. ELSE ကိုအသုံးပြုပြီး Calculator တစ်ခုတည်ဆောက် မှာဖြစ်ပါတယ်။

```
print("""
########################
#  Simple Calaculator  #
########################
""")
number1 =int(input("Enter The Number 1:"))
number2 =int(input("Enter The Number 2:"))
operators = input("Enter The Operators:")

if (operators == "+"):
    sum = number1 + number2
    print(("Sum of Number1 and Number2 %d")%sum)

elif(operators == "-"):
    sub = number1 - number2
    print(("Sub of Number1 - Number2 %d")%sub)

elif (operators == "*"):
    mul = number1 * number2
    print(("Mul of Number1 x Number2 %d")%mul)

elif (operators == "/"):
    div = number1 / number2
    print(("div of Number1 / Number2 %f")%div)

elif (operators =="%"):
    mod = number1 % number2
    print(("Moud of Number1 and Number2 %d")%mod)

else:
    print("Error Input")
```

Simple Calculator လေးကိုရှင်းပြပါမယ်။Variable 3 ခုတည်ဆောက်ထားပြီး input() နဲ့ user_input လုပ်ထားပါတယ်။number1 မှာ ပထမ ဂဏန်းထည့်ရမှာဖြစ်ပြီး number2 မှာတော့ ဒုတိယ ဂဏန်းထည့်ရမှာပါ ဒီအတွက် int(input()) နဲ့လက်ခံထားပါတယ် ဂဏန်းသုံးရမှာဖြစ်လို့ပါ။တစ်ခုကတော့ Operators လက်ခံ ဖို့ဖြစ်ပါတယ် ဘာတွေလဲဆိုတော့ + , - , * , / ,% တို့ကို အသုံးပြုမှာဖြစ်ပါတယ်။ဒါတွေကိုလက်ခံဖို့ကတော့ String အနေနဲ့ဘဲရပါတယ် ဒါကြောင့်တိုက်ရိုက် String နဲ့လက်ခံထားပါတယ်။နောက်ပြီး IF နဲ့ စစ်ဆေးတဲ့အခါမှာ operators ကိုစစ်ဆေးပါတယ် ပထမဆုံး + ကို စစ်ဆေးပြီး operator == "+" ဖြစ်ခဲ့မယ်ဆိုရင် number1 + number2 ကိုပေါင်းပြီး အဖြေထုတ်ပေးဖို့အတွက်ခိုင်းစေထားပါတယ်။အဲဒီမှာ Format Specifier သုံးထားတာတွေ့ရမှာပါ "Sum of Number1 and Number2 %d" %sum ဆိုပြီးသုံးထားတာပါ။အဲဒီလိုဘဲ"-" လာရင် အနှုတ်အလုပ်ကိုလုပ်တယ် " * " လာရင်လည်း အမြောက်အလုပ်ကိုလုပ်တယ်။ဒီနေရာမှာတစ်ခုပြောချင်တာရှိပါတယ် "/ " ဂင်လာတဲ့အခါ အစားဖြစ်လို့ ဒသမကိန်းလည်းထွက်နိုင်တာမို့ Format Specifier မှာ Float ဖြစ်တဲ့ %f ကို အသုံးပြုထားတာမြင်ရမှာဖြစ်ပါတယ်။

Logical Operator

Logic တွေသုံးဖို့အတွက် Logical Operators တွေကိုအသုံးပြုနိုင်ပါတယ်။Comparing Operators တွေနဲ့ Logical Operators တွေပေါင်းသုံးပြီး Decisions တွေလုပ်တဲ့အခါမှာ ပိုပြီးလွယ်လွယ်ကူ

ကူနဲ့ Program တွေကိုရေးနိုင်လာတာတွေ့ရမှာပါ။Logical Operator တွေကတော့ AND (and) , OR (or) , NOT (not) တို့ဖြစ်ပါတယ်။

AND(and)

AND ဟာ Logic အရ အခြေနေနှစ်ခုစစ်ဆေးချင်တဲ့အခါမှာအသုံးပြုပြီး စစ်ဆေးတဲ့ အခြေနေ နှစ် ခု လုံးမှန်မှသာ True ဖြစ်ပါတယ်။ကျွန်တော်တို့ Facebook မှာ Username ရော Password ရော မှန်မှ Login ဝင်နိုင်တာ AND ဖြစ်ပါတယ်။

```
name = input("Enter Your Name:")
password = input("Enter Your Password:")

if(name=="AungMoeKyaw" and password=="1234"):
    print("Hello")
else:
    print("AND need to Same Name and Password")
```

ဒီ Program ကိုကြည့်လိုက်ပါ။name က AungMoeKyaw နဲ့ password က 1234 နှစ်ခုလုံး တူညီမှ Hello ဆိုတဲ့စာသားအဖြေထွက်မှာပါ AND ဖြစ်တဲ့အတွက် အခြေနေနှစ်ခုလုံးမှန်ဖို့လိုအပ်ပါတယ်။

OR(or)

OR ကတော့ အခြေနေ အခြေနေ နှစ်ခုစစ်ဆေးရင် တစ်ခု True ဖြစ်တာနဲ့ အခြေနေမှန်ဖြစ်ပြီး True အဖြစ်ရောက်ပါတယ်။Username နဲ့ Password ဥပမာကိုဘဲပြန်ပြောရရင် OR သုံးထားရင် Username သို့မဟုတ် Password တစ်ခုခုမှန်တာနဲ့အလုပ်လုပ်ပါတယ်။အခြေနေနှစ်ခုမှ တစ်ခုမှန်တာနဲ့ True အဖြစ် Return ပြန်ပါတယ်။

```
name = input("Enter Your Name:")
password = input("Enter Your Password:")

if(name=="AungMoeKyaw" or password=="1234"):
    print("Hello")
else:
    print("AND need to Same Name and Password")
```

ဒီ program က အပေါ်က နဲ့အတူတူပါဘဲ AND နဲ့ OR ဘဲကွာပါတယ် ဒါပေမယ့် Run ကြည့်လိုက်တဲ့အခါမှာ Username သို့မဟုတ် password တခုခုမှန်တာနဲ့ Hello အလုပ်လုပ်ပါလိမ့်မယ်။ဘာကြောင့်လဲဆိုတော့ အခြေနေ နှစ်ခုမှာ တစ်ခုမှန်ရင် OR က အလုပ်လုပ်လို့ဖြစ်ပါတယ်။

NOT(not)

NOT ကတော့ တကယ့်ပြောင်းပြန်ပါဘဲ ဖြစ်စဉ်တစ်ခုရဲ့ရှေ့မှာ not ထည့်လိုက်ရင် မှန်တာလာရင် မှားပြီး မှားတာလာမှ မှန်ပါလိမ့်မယ်။ရှုပ်သွားမယ်ထင်တယ် ဒီလိုဗျာ not x == 100 ဆိုရင် အရင်က x က 100 သာ ဖြစ်မယ်ဆို True ဖြစ်ပြီး IF ထဲက အလုပ်က လုပ်မှာပေါ့ ဒါပေမယ့် not ပါခဲ့ရင်တော့ x က 100 ဖြစ်နေရင် အလုပ်မလုပ်ဘဲ False ဖြစ်ပြီး တခြားအမှားတွေလာရင်တော့ True ဖြစ်ပါလိမ့်မယ်။

```
name = input("Enter Your Name:")
password = input("Enter Your Password:")

if(not(name=="AungMoeKyaw") and not(password=="1234")):
    print("Hello")
else:
    print("AND need to Same Name and Password")
```

ဒီ Program ကတော့ Condition တွေရဲ့ ရှေ့မှာ not တွေပါဝင်နေတော့ အဖြေမှန်ကိုက လွဲပြီး တခြား ဘယ်ဘဲ ပေါက်ကရရိုက်ရိုက် Hello ထွက်နေမှာဖြစ်ပါတယ်။အမှန်ဖြစ်တဲ့ AungMoeKyaw နဲ့ 1234 ရိုက်မိရင်တော့ False ဖြစ်ပြီး else ထဲရောက်သွားပါလိမ့်မယ်။ဒါဆိုရင် ကျွန်တော်တို့ Example Program လေးရေးပြီးလေ့ကျင့်ပါမယ်။

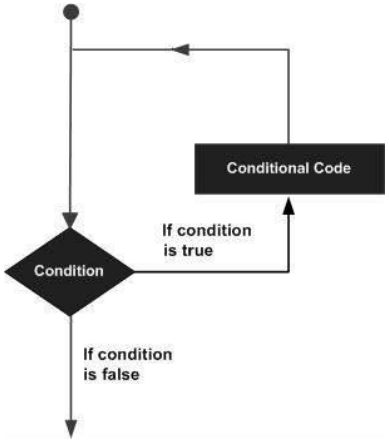
ကျွန်တော်တို့ Facebook တွေ Gmail တွေနဲ့တခြား Login ဝင်တဲ့ပုံစံမျိုး Program အသေးလေး ရေးပြ ပါမယ် ဘယ်လိုမျိုးလဲဆိုတော့ Username ရော Password ရော မှားရင် Login Fail ဖြစ်မယ် Username မှန်ပြီး Password မှားနေရင် Username မှားကြောင်းပြောမယ် Password မှားနေရင် Password မှားကြောင်းပြောမယ် Username ရော Password ရော မှန်ရင်တော့ Login Successful ဖြစ်မယ် အဲလို Program ရေးပါမယ်။

```
name = input("Enter Your Name:")
password = input("Enter Your Password:")

if (name!="AungMoeKyaw" and password =="1234"):
    print("Username Fail")

elif (name =="AungMoeKyaw" and password !="1234"):
    print("Password Fail")

elif (name =="AungMoeKyaw" and password == "1234"):
    print("Login Successful")

else:
    print("Login Fail")
```

Variable ၂ ခုတည်ဆောက်ပြီး User input ယူထားပါတယ်။အဲဒီ User Input မှာ အရင်ဆုံး Username != နဲ့ စစ်ဆေးထားပါတယ်။AungMoeKyaw နဲ့မညီဘဲ password ကတော့ 1234 နဲ့ညီနေမယ်ဆိုရင် Username အမှားလို့ပြပြီး password က 1234 နဲ့မညီဘဲ Username က AungMoeKyaw နဲ့ ညီနေရင် Password မှားတာလို့ဖော်ပြပါမယ် နှစ်ခုလုံးတူညီနေမယ်ဆိုရင်တော့ Login Successful ဖြစ်မယ်လို့ပြောပြီး 2 ခုလုံးမှားနေမယ်ဆိုရင် Else ထဲကိုရောက်ပြီး Login Fail မယ်လို့ပြောပါမယ်။

Do Something – Again and Again

အလုပ်တစ်ခုကို အကြိမ်ကြိမ်လုပ်ဖို့အတွက်ထပ်ခါထပ်ခါ ရေးတာ နည်းလမ်းမဟုတ်ပါဘူး တချို့ကိစ္စတွေမှာလည်း ထပ်ခါထပ်ခါရေးဖို့ဆိုတာမဖြစ်နိုင်ပါဘူး ဒါကြောင့်ကျွန်တော်တို့က Loop တွေကို အသုံးပြုရပါတယ်။Loop မှာ While Loop ရယ် For Loop ရယ်ဆိုပြီး Python နှစ်မျိုးအသုံးပြုပါတယ်။နောက် ထပ်ကတော့ Nested Loop ဖြစ်ပါတယ်။Loop ထဲမှာ Loop ထည့်ပတ်တာမျိုးပါ။အရင်ဆုံး While Loop အကြောင်းပြောပါမယ်။



အပေါ်ကဖော်ပြထားတဲ့ပုံကြည့်လိုက်ပါ။Loop ဆိုတာ စစ်ဆေးထားတဲ့ condition မှန်နေသရွေ့ Loop Condition Statement ထဲမှာလည်ပတ်နေပြီး condition မှားယွင်းသွားတဲ့အခါမှာတော့ Loop ထဲက ထွက် ပြီး တခြားအလုပ်ကိုလုပ်စေပါတယ်။

Whlie Loop – Forever Loop

Whlie Loop အကြောင်းကိုပြောပါမယ်။Python မှာတော့ တကယ်အသုံးဝင်ပြီး ကျွန်တော်တို့ Android Font Changer Script လိုဟာမျိုး၊Ubuntu Easy Script လိုဟာမျိုးရေးတဲ့အခါမှာ While Loop ကိုအကြိမ်ကြိမ် သုံး ရေးသွားမှာတွေ့ရပါလိမ့်မယ်။ဒါဆိုရင် While Loop မပြောခင်မှာ decrease and increase လုပ်ငန်းစဉ် ကို ပြောရမှာပါ။IDLE ကိုဖွင့်ပြီး Decrease and Increase အတွက်လက်တွေ့ပြပါမယ်။

```
>>> i = 0
>>> i +=1
>>> i
1
```

ဒါက Increase ရဲ့နာမူနာပါဘဲ။variable i ထဲမှာ 0 ဘဲသိုလျောင်ထားပါတယ်။ဒါပေမယ့် အောက်မှာ i +=1 ဆိုပြီးပေါင်းထည့်ရပါတယ်။i ကိုအဖြေထုတ်လိုက်တဲ့အခါမှာ 1 ကိုတွေ့ရမှာပါ ဒါက i ကို 1 တိုးလိုက်တာ ဖြစ်တဲ့ အတွက် Icrease လုပ်တယ်လို့ခေါ်ပါတယ်။ဒါဆိုနောက်တစ်ခုလုပ်ကြည့်ရအောင်။

```
>>> a = 0
>>> a += 5
>>> a
5
```

ဒါကတော့ variable a ကို increase လုပ်တဲ့နေရာမှာ 5 တိုးတဲ့သဘောပါ။ဒါဆိုရင်သိရမှာက ကျွန်တော်တို့ increase လုပ်တဲ့နေရာမှာကြိုက်သလောက်တန်ဖိုးတိုးလို့ရတယ်ဆိုတာပါဘဲ ပြီးတော့ ပေါင်းပြီးတိုးတာမှ မဟုတ်ပါဘူး။မြောက်တာ စားတာ အကုန်လုပ်လို့ရပါတယ်။

```
>>> i = 1
>>> i *= 2
>>> i
2
>>> y = 6
```

```
>>> y /=2
>>> y
3.0
>>> x = 6
>>> x %=2
>>> x
0
```

ဒါကတော့ အမြောက် အစား အကြွင်း ကိုအသုံးပြုထားတာပါ။ဒါဆိုရင် increase လုပ်ဖို့ ပေါင်းတာ မြောက်တာ ကိုလုပ်တယ်ဆိုရင် Decrease လုပ်ဖို့ဆိုရင်တော့ နှုတ်တာ နဲ့ စားတာ ကိုလုပ်ရမှာပေါ့။အိုကေ လက်တွေ့ကြည့်ရအောင်။

```
>>> z = 10
>>> z -=4
>>> z
6
```

ဒီမှာ z ထဲမှာ 10 ကိုသိုလှောင်ထားပြီးတော့ Decrease လုပ်တာကတော့ 4 ကိုလုပ်သွားပါတယ် z -=4 ပါ ဘဲ ဒီတော့ z က 4 လျော့သွားတော့ 6 ဖြစ်ပါတယ်။ဒါဆိုရင် While Loop ကိုလေ့လာကြည့်လို့ရပြီပေါ့။

While Loop ကို အောက်ပါအတိုင်းရေပါတယ်။expression ဆိုကတော့ Condition ပါဘဲ။အဲဒီ Condition မှန်နေသရွေ့ statement(s) ကအလုပ်လုပ်နေမှာဖြစ်ပါတယ်။Loop ပတ်နေတယ်လို့ပြောချင်တာပါ။Condition က False သွားရင်တော့ Loop ထဲက ထွက်သွားမှာဖြစ်ပါတယ်။

```
while expression:
    statement(s)
```

ဒါဆိုရင်ကျွန်တော်တို့လက်တွေ့လုပ်ကြည့်ရအောင်။

```
i = 0
while i<10:
    print("Number Increase : %d"%i)
    i +=1
```

ဒီ Program လေးမှာ i က 0 ဖြစ်ပါတယ် ဒါကို while ထဲမှာစစ်ဆေးထားတာက i < 10 ဆိုတော့ i က 10 မဖြစ်မခြင်း ဒီ loop က အလုပ်လုပ်နေရမှာပါ။အဲဒီ Loop ထဲမှာ လုပ်မယ့်အလုပ်ကတော့ Number Increase : I ဆိုပြီး i တိုးလာတာကို စာသားနဲ့တွဲပြီးဖော်ပြမယ်လို့လုပ်ထားပါတယ်။ပြီးရင်တော့ i +=1 လဲ Loop ထဲမှာ ဘဲ အလုပ်လုပ်ထားပါတယ်။ဒီတော့ ဘယ်လိုဖြစ်မလဲဆိုတော့။

i ကိုစတင်စစ်ဆေးပါတယ် i က 0 ဖြစ်တဲ့အတွက် 10 အောက်ငယ်တယ်ဒီတော့ အခြေနေက True ဖြစ်ပြီး Loop တစ်ကြိမ်အလုပ်လုပ်တယ်ဒီတော့ Number Increase 0 ဆိုပြီးစဖြစ်တယ် i +=1 ကိုတွေ့တော့ i ကို 1 ပေါင်းတယ် ဒီတော့ i က 1 ဖြစ်သွားတယ်။i က 1 ကလည်း 10 အောက်ငယ်သေးတော့ Loop ထပ်လုပ်ပါတယ် Number Icrease ဆိုတဲ့စာသားနဲ့လက်ရှိ i တန်ဖိုး 1 ကိုတွဲဖက်အဖြေထုတ်ပါတယ် ပြီးတော့ i +=1 လုပ် ငန်းစဉ်လုပ်လို့ i က 2 ဖြစ်ပါတယ်။ဒီလိုနဲ့ 9 အထိပြီးသွားပြီ i +=1 လုပ်တဲ့အခါမှာ i က 10 ဖြစ်ပါတယ်။ဒီတော့ while မှာစစ်ဆေးတဲ့အခါမှာ 10 က 10 အောက်မငယ်တဲ့အတွက် Loop လည်းရပ်သွားပြီး နောက်ထပ်အလုပ်လဲထပ်မလုပ်တော့ပါဘူး။

ဒါဆိုရင်နောက် Program တစ်ခုကိုသွားရအောင် ကျွန်တော်တို့ 1 ကနေ 20 အတွင်းရှိတဲ့ နာပတ်တွေထဲကမှ Odd Number လို့ခေါ်တဲ့ မကိန်းကိုရှာဖို့အတွက် While Loop ကိုသုံးပြီးရေးကြည့်ရအောင်။

```
i = 1
while i<=20:

    if(i%2!= 0):
        print("Odd Number : %d"%i)

    i +=1
```

1 ကနေ 20 အတွင်းလိုချင်တာဖြစ်လို့ i တန်ဖိုးကို 1 ကနေစထားပြီး 20 ထိဖြစ်ချင်တာမို့ <=20 ထားထားပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ Loop ပတ်လိုက်မယ်ဆိုရင်တော့ 1 ကနေ 20 ထိရပြီပေါ့။တစ်ခုစဉ်းစားကြည့်ရအောင် မကိန်း(odd) ဆိုတာ 2 နဲ့စားလို့မပြတ်တဲ့ ဂဏန်းတွေကိုခေါ်တာပါ။ဒါကြောင့် While ထဲမှာ IF နဲ့တစ်ခါထပ်စစ်ဆေးလိုက်ပါတယ် if (i%2 !=0) ဆိုတာ i ကို 2 နဲ့စားပြီးရတဲ့အကြွင်းက 0 နဲ့မညီဘူးဆိုရင် print နဲ့ Odd Numeber ဆိုပြီး i တန်ဖိုးကိုထုတ်ထားပါတယ် ဒီထဲမှာထုတ်တဲ့ i တန်ဖိုးကတော့ 2 နဲ့စားလို့ အကြွင်းသုည မဖြစ်တဲ့ i တန်ဖိုးတွေဘဲထွက်လာတာပါ။i +=1 ကတော့ increasement လုပ်ထားတာပါ။ဒါဆိုရင် စုံကိန်းလည်းရှာနိုင်ပြီဖြစ်ပါတယ်။မကိန်းမှာ i%2 !=0 ဆိုပြီးစစ်ထားရင် စုံကိန်းအတွက်က 2 နဲ့စားလို့ပြတ်တဲ့ကိန်း တွေကိုစစ်ဆေးရမှာပေါ့။

```
i = 1
while i<=20:

    if(i%2 == 0):
        print("Even Number : %d"%i)

    i +=1
```

အနည်းငယ်ဘဲပြောင်းလဲလိုက်ပါတယ် အခု i ကို 2 နဲ့စားပြီးအကြွင်း 0 ဖြစ်တဲ့ ကိန်းမှန်သမျှကို Even Number ဆိုတဲ့စာသားနဲ့တွဲထုတ်လိုက်ပါတယ်။အခုဆိုရင်အဖြေမှာ စုံကိန်းတွေဘဲတွေ့ရမှာဖြစ်ပါတယ်။

စုံကိန်းရော မကိန်းရောဖော်ပြပေးမယ့် Program လေးကိုလေ့လာကြည့်ရအောင်ပါ စုံကိန်း သို့မဟုတ် မကိန်းကြည့်သတ်သတ်စစ်ထုတ်ချင်တဲ့အခါမှာ IF တစ်ခုထဲကိုသုံးပြီးအဖြေထုတ်ပါတယ်။စုံရော မရောတွဲထုတ်ဖို့ဆိုရင်တော့ IF ရော ELSE ရောသုံးပြီးထုတ်နိုင်ပါတယ်။

```
i = 1
while i<=20:

    if(i%2 == 0):
        print("Even Number : %d"%i)
    else:
        print("Odd Number  : %d"%i)

    i +=1
```

ဒီ Program မှာ i%2 ==0 ဖြစ်တဲ့ အဖြေတွေက Even Number အဖြစ် အဖြေထွက်မှာဖြစ်ပါတယ် အဲဒီလို မဟုတ်ဘဲ IF နဲ့စစ်ဆေးတာမှာ မအောင်မြင်တဲ့ (မမှန်တဲ့အခြေနေတွေက) Else ထဲရောက်သွားပြီး Odd Number အဖြစ်အဖြေထွက်ပါတယ်။Forever Loop ဆိုတာဘာကိုပြောတာလဲဆိုတာကို ကျွန်တော် Program တခုနဲ့လက်တွေ့ပြပါမယ်။

Simple Calculator With Forever Loop

```
cho = ""
while (cho !="end"):
    print("""
      ####################################
      ##  Simple Calculator with Forever ##
      ##            Loop                  ##
      ##     Mcoder Python Tutorial       ##
      ##                                  ##
      ##                                  ##
      ####################################
    """)


    print("""
    Choice The Number =>
    1.Calculate Number:
    2.Exit Program
    """)

    inp = input("Input Number :")

    if (inp =="1"):
        number1 = int(input("Enter The First Number:"))
        number2 = int(input("Enter The Second Number:"))
        operator = input("Enter The Operators:")

        if(operator =="+"):
            sum = number1 + number2
            print("Answer is %d"%sum)
        elif(operator =="-"):
            sub = number1 - number2
            print("Answer is %d"%sub)
        elif(operator == "*"):
            mul = number1 * number2
```

```
print("Anser is %d" % mul)
elif (operator == "/"):
div = number1 / number2
print("Answer is %f" % div)
elif (operator == "%"):
mod = number1 % number2
print("Answer is %d" % mod)
else:
print("Input Error")

elif (inp == "2"):
print("GoodBye!!!!")
cho = "end"

input()
```

ဒီ Program ကလွယ်လွယ်လေးဘဲနော်ရှည်တယ်ဆိုပြီး စိတ်ရှုပ်မသွားနဲ့အုန်း Variable တစ်ခုတည်ဆောက်ပါ တယ်။အဲဒီ Variable နာမည်က cho = "" ဖြစ်ပါတယ်။သူ့အထဲမှာ ဘာတန်ဖိုးမှမပါပါဘူး ဒါကြောင့် အလွတ် ထားတာပါ။နောက်ပြီး while ထဲမှာ စစ်ဆေးထားတဲ့ condition ကတော့ cho !="end" ဖြစ်ပါတယ် ပြောချင် တဲ့သဘောက end ဆိုတာနဲ့မညီမချင်း Loop က အလုပ်လုပ်နေမှာပါ။Loop ထဲမှာပါတဲ့အလုပ်တွေကတော့ အများကြီးပါ print နဲ့ Blah Blah ဆိုတဲ့စာတွေလဲပါဝင်ပါတယ်။အဓိက အလုပ်ကတော့ input() နဲ့ အဝင်လက် ခံထားပြီးတော့ 1 ဝင်လာမယ်ဆိုရင် Calculator ထဲကိုရောက်သွားမယ်(အရှေ့က Calculator Program မှာ ကြည့်ပါ) 2 ကိုရွေးရင်တော့ cho ကို end ထည့်ပေးလိုက်ပါတယ် ဒီတော့ while ထဲမှာ !="end" နဲ့မညီရင် မှန် လို့ Loop ဖြစ်နေပေမယ့် cho က end ဖြစ်သွားတဲ့အခါမှာတော့ ညီသွားတဲ့အတွက် condition မမှန်တော့ ပါဘူး ဒါကြောင့် Program လည်းရပ်သွားပါလိမ့်မယ်။အဆုံးမှာ input() ဆိုပြီးထည့်ထားတာကတော့ Program ကို ချက်ချင်းဆိုသလိုပြီးဆုံးသွားမှာစိုးရိမ်တဲ့အတွက် Key ခနရပ်ပေးထားတဲ့သဘောပါ။ကီးဘုတ်က တစ်ခုခု ကိုနှိပ်မှ Program က ဆုံးခန်းတိုင်ပါတယ်။

Tuples,Lists with Loop

Tuples တွေ Lists တွေနဲ့ Loop နဲ့တွဲဖက်အသုံးပြုပုံတွေကို ကျွန်တော် ဖော်ပြပါမယ်။Tuples တွေ Lists တွေနဲ့တွဲဖက်ပြီး items တွေကိုဖော်ပြတာ၊items တန်ဖိုးတွေပေါင်းထည့်တာစတာတွေပြုလုပ်နိုင်ပါတယ် ။လက်တွေ့ Program တွေနဲ့အသုံးပြုတာက ပိုပြီးရှင်းလင်းမြင်စေပါတယ်။

ပထမဆုံး Tuples ထဲက items တွေကို အရင်လို index numbers တွေအများကြီးရေးပြီး အဖြေထုတ်မှာ မဟုတ်ဘဲ Loop နဲ့ဘဲအဖြေထုတ်သွားမှာဖြစ်ပါတယ်။

```
i = 0
tuple_1 = (1,2,4,5,6,7,8,9,10)
while (i<len(tuple_1)):
    print(tuple_1[i])
    i +=1
```

ဒီမှာ ပုံမှန် Loop တွေလိုမျိုးဘဲ variable တစ်ခုပေးပြီး တန်ဖိုးသုည‌သို့လျောင်ထားပါတယ်။နောက်တွေ့ရတာ ကတော့ တန်ဖိုးတွေသို့လျောင်ထားတဲ့ Tuple တစ်ခုကိုတွေ့ရပါတယ်။အဲဒီ Tuple ထဲက items တွေကို ကျွန်တော်တို့က အဖြေထုတ်ချင်တာဖြစ်ပါတယ်။အဲဒီတော့ while ထဲမှာ i<len(tuple_1) ဆိုပြီး သုံးထား တာ တွေ့ရပါလိမ့်မယ်။Loop တစ်ခုမှာ condition ဘဲပြောပြော Loop အရေအတွက်ဘဲပြောပြော အဆုံးသတ် ဖို့အတွက်ထားရပါတယ် i<10 ဆိုရင် i က 10 အောက်ထိဘဲအလုပ်လုပ်မယ်ပေါ့။ဒါပေမယ့် ဒီ Tuple မှာတော့ ကျွန်တော်တို့က ထင်သလိုပေးလို့မရပါဘူး 10 ပေးလိုက်လို့ items က 10 ခုမရှိရင် Error တတ်ပါလိမ့်မယ် ဒါမှမဟုတ် items က 10 ခုကျော်နေရင်လဲ မပြည့်မစုံအဖြေရပါလိမ့်မယ် ဒါပေမယ့် ဒီပြဿနာကိုဖြေရှင်းဖို့ အတွက် len() ကိုအသုံးပြုနိုင်ပါတယ်။len() ဆိုတာကတော့ Tuple ဘဲဖြစ်ဖြစ် List ဘဲဖြစ်ဖြစ် String ဘဲဖြစ်ဖြစ် သူတို့ရဲ့အရေတွက်တွေကိုဖော်ပြပေးနိုင်ပါတယ်ဒါကြောင့် i<len(tuple_1) လို့ရေးခြင်းဖြင့် Tuple ထဲမှာ အခု 100 ရှိရင် လည်း အခု 100 လုံးအလုပ်လုပ်ပါလိမ့်မယ်။items ၁၀ ခုဘဲရှိရင်လဲ ၁၀ ခုစာအလုပ်လုပ်ပါလိမ့်မယ်။ နောက်ပြီးမှ အဖြေထုတ်ဖို့ print(tuple_1[i]) ဆိုပြီးသုံးထားပါတယ်။Tuple တွေ ကို အဖြေထုတ်တဲ့အခါမှာ tuple[index] ဆိုပြီး index number နဲ့အဖြေထုတ်တယ်မလား ဒါကြောင့် i တန်ဖိုး 0 ကနေစပြီး tuple items ရှိသလောက် Run စေပြီးတော့ အဖြေထုတ်စေပါတယ်။i +=1 ကတော့ Increase လုပ်ထားတာဖြစ်ပါတယ်။

ဒါဆိုရင် List နဲ့တွဲဖက်အသုံးပြုကြည့်ရအောင်ဗျာ။ကျွန်တော်တို့ Lists ကို အသုံးပြုဖို့လဲ အပေါ်ကနည်း အတိုင်းဘဲ အသုံးပြုနိုင်ပါတယ် အထူးထွေရှင်းပြနေစရာလိုမယ်မထင်ပါ။

```
i = 0
list_1 =["AungAung","MaungMaung","KyawKyaw","ZawZaw","KhinKhin"]
while i<len(list_1):
    print(list_1[i])
    i +=1
```

Dictionary With Loop

Dictionary ကို Loop အသုံးပြုပြီး အောက်ပါ ပုံစံအတိုင်း ဖော်ပြတဲ့ Program လေးကို ကျွန်တော်တို့ ရေးပါ့မယ်။

```
name      aungmoekyaw
--------------------------
Height    5.9'
--------------------------
Blood     O
--------------------------
No.       137
--------------------------
Ph.       09799130782
```

```
--------------------------
```

Dictionary မှာ item တွေက Key and Value ဆိုပြီးတော့သွားတာဖြစ်ပါတယ်။ဒါကြောင့်ကျွန်တော်တို့ iIdex number နဲ့ခေါ်သုံးလို့မရပါဘူး။while မှာကလည်း loop ပတ်တဲ့အခါ ကိန်းဂဏန်းကိုသာအများဆုံးပတ် ကြပါတယ်။ဒါကြောင့် ကျွန်တော်တို့က Dictionary ကို သုံးမယ်ဆိုရင် ကျွန်တော်တို့Number သုံးလို့ရမယ့် Tuple တို့ List တို့နဲ့တွဲဖက်အသုံးပြုရပါတယ်။အလွယ်ကူဆုံးတွဲဖက်အသုံးပြုပုံကိုဖော်ပြထားပါတယ်။

```
dictionary = {"name":"aungmoekyaw","Height":"5.9'","Blood":"O","No.":"    137","Ph.":"
09799130782"}
v=tuple(dictionary.keys())
i =0
while i<len(v):
    print(v[i]+"\t"+dictionary[v[i]])
    print("--------------------------")
    i +=1
```

dictionary တစ်ခုတည်ဆောက်ထားပါတယ် အဲဒီထဲမှာ ကျွန်တော်ရဲ့ကိုယ်ရေးအချက်လက်တစ်ချို့ပါဂင်ပါတယ်။အဲဒီမှာ အရင်ဆုံး ကျွန်တော်က dictionary ရဲ့ keys() ကိုသုံးပြီး Key တွေကြည့်စွဲထုတ်လိုက်ပါတယ်။ရှေ့မှာ tuple() လို့ခေါ်လိုက်တာကတော့ အားလုံးကို tuple() ပြောင်းပြစ်လိုက်တာဖြစ်ပါတယ်။အဲဒီပြောင်းပြစ်လိုက်တဲ့ Tuple တွေကို v ဆိုတဲ့ variable ထဲကိုထည့်လိုက်ပါတယ်။ပြီးရင် while loop ကိုစတင်ပါတယ် len(v) ဆိုတော့ကတော့ dictionary ရဲ့ key အရေတွက်နဲ့တူတူပါဘဲ ဘာကြောင့်လဲဆိုတော့ ရှိသမျှ dictionary ရဲ့ Key တွေကို ကျွန်တော်တို့က tuple ပြောင်းလဲပြီးဖြစ်ပါတယ်။ပြီးတဲ့ အခါမှာတော့ print နဲ့ v[i] ကိုအဖြေထုတ်ပါတယ် v[i] ဆိုတာကတော့ Tuple ထဲမှာရှိနေတဲ့ Dictionary ရဲ့ Key တွေကိုဖော်ပြဖို့ဖြစ်ပါတယ်။ဒီနေရာမှာသုံးထားတဲ့ + က စာကြောင်းဆက်တာပါ ပေါင်းတာမဟုတ်ပါဘူး "\t" ကတော့ Tap ခုန်တာဖြစ်ပါတယ် dictionary[v[i]] ကတော့ dictionary ထဲကို Key တွေ Passing လုပ်ပြီး Value တွေကို ထုတ်ထားတာ ဖြစ်ပါတယ်။

Loop Control Statement

Loop မှာလည်း သူ့ရဲ့ စနစ်တချို့ကို Control လုပ်ဖို့ Loop Control Statemnt တွေရှိပါတယ်။အဲဒါတွေကတော့ break , continue , pass တို့ဘဲဖြစ်ပါတယ်။ဘယ်လိုမျိုးတွေသုံးလဲဆိုတာကိုလည်း လက်တွေ့ပြော ပြပါမယ်။

Break

Break ဆိုတဲ့အတိုင်းဘဲ ရပ်တန့်လိုတဲ့အချိန်တွေမှာသုံးပါတယ် ကျွန်တော်တို့ Loop ကို ကျွန်တော်တို့ အလုပ်တစ်ခုမှာရပ်တန့်မယ်ဆိုတာကို Break နဲ့အသုံးပြုရပ်တန့်လို့ရပါတယ်။

```
z = 0
i = 0
while i <10:
    print(i)


    z += i
    i += 1


    if i == 5:
        print("Total I :", z)
        break
```

ဒီ Program လေးကတော့ Loop ပတ်မယ် အဲဒီ ပတ်လို့တဲ့အဖြေအားလုံးကို တန်ဖိုးတစ်ခုထဲမှာပေါင်းပြီး ဖော်ပြမယ်ပြီးတော့ 5 ဖြစ်သွားတဲ့အချိန်မှာ Loop ရပ်ဖို့အတွက်ဖြစ်ပါတယ်။ဒီမှာ variable ၂ခုတည်ဆောက်ထားပါတယ်။z , i ပေါ့ နှစ်ခုစလုံး တန်ဖိုးက 0 ဖြစ်ပါတယ်။နောက်ပြီး while နဲ့ i ကို 10 အောက်ငယ်တဲ့ အထိ Loop လုပ်မယ်လို့ပြောထားပါတယ် ပြီးတော့ print (i) ဆိုပြီး i တန်ဖိုးကို အဖြေထုတ်တယ် ပြီးတော့ အဲဒီ i တွေကို z ထဲကိုထည့်ပေါင်းပါတယ်။ပြီးတဲ့အခါမှာ i ကို increase လုပ်ထားပါတယ်။နောက်ထပ် ထပ်ပြီးစစ်ဆေးထားတာကတော့ i တန်ဖိုး 5 ဖြစ်သွားရင်။print နဲ့ i အားလုံးပေါင်းလဒ် z ကို အဖြေထုတ်ပြီး break လို့ရေးထားပါတယ်။ဒီတော့ ပထမဆုံး Loop စလုပ်တော့ i တန်ဖိုး 0 ဖြစ်တယ် print က 0 ကိုအဖြေထုတ်တယ် z ထဲကို 0 ထည့်တယ်။ပြီးတော့ i ကို 1 ပေါင်းထည့်တယ်။အဲဒီနောက် if အခြေနေကိုရောက်တယ် ဒါပေမယ့် i တန်ဖိုးက အခုမှ 1 ဘဲရှိသေးတာ 5 နဲ့မညီသေးတဲ့အတွက် သူထဲမှာလုပ်မယ့်အလုပ်တွေက အရေးမပါသေးဘူး။ဒီလိုနဲ့ Loop ထပ်လုပ်ဖို့စစ်ဆေးတဲ့အခါမှာ i တန်ဖိုးက 1 ဖြစ်တဲ့အတွက် 10 အောက်ငယ်သေးလို့ထပ်ပတ်ပါတယ် စောစောကလုပ်တဲ့အတိုင်းထပ်လုပ်ပါတယ် z ထဲကို လည်း ၁ ပေါင်းပါတယ် ဒီလိုနဲ့ဘဲ i တန်ဖိုး 4 ဖြစ်ပြီး တစ်ခုတိုးလို့ 5 ဖြစ်တဲ့အခါမှာ if လုပ်ငယ်က ထလုပ်လို့ break နဲ့တွေ့ပြီး 10 ထိ Loop မလုပ်တော့ဘဲ 4 ကြိမ်မြောက်မှာဘဲ Loop ရပ်တန့်သွားပါတယ်။z တန်ဖိုးကတော့ 10 ရမှာပါ ဘာလို့လဲဆိုတော့ 0 + 1 + 2 + 3 + 4 ဖြစ်လို့ပါ။program က အစဉ်လိုက်တိုင်းတစ်ကြောင်းခြင်းအပေါ်ကနေအောက်ထိအလုပ်လုပ်ပါတယ် ဒါကြောင့် i က 4 ဖြစ်ပြီး i +=1 မှာတင် ရပ်တန့်သွားပါတယ်။တကယ်လို့ 5 ထိအလုပ်လုပ်စေချင်တယ်ဆိုရင် increase လုပ်ထားတာကို if i == 5 ရဲ့အောက်မှာထားမှရမှာပါ။

```
z = 0
i = 0
while i <10:
    print(i)
```

```
z += i

if i == 5:
    print("Total I :", z)
    break

i += 1
```

ဒီလိုမျိုးရေးတယ်ဆိုရင်တော့ z တန်ဖိုးသည်လည်း 15 ဖြစ်ပြီး i ကလည်း 5 ထိရောက်ပြီးမှ Loop ရပ်တန့် မှာ ဖြစ် ပါတယ်။

Continue

ဒါကတော့ သူထဲမှာရှိတဲ့ အလုပ်တွေကိုပယ်ချလိုက်ပြီးတော့ condition ကိုပြန်ရောက်သွားပြီး ပြန်ပြီးတော့ Loop ကိုပြန်လုပ်စေပါတယ် ဒါပေမယ့် continue အောက်က အလုပ်မှန်သမျှကတော့ လုပ်မှာ မဟုတ်ပါဘူး။ဒီမှာကြည့်လိုက်ပါ။

```
i = 0
while i < 10:
    i += 1
    if i == 5:
        continue

    print(i)
```

ဒီအဖြေမှာကြည့်လိုက်တဲ့အခါမှာ 5 ကိုကျော်ပြီး တခြား ဂဏန်းတွေမှာအလုပ်လုပ်သွားတာ တွေ့ရပါ လိမ့်မယ် ဘာကြောင့်လဲဆိုတော့ continue က သူ့အတွက်လုပ်မယ့်လုပ်ငန်းစဉ်တွေကို rejects လုပ်လို့ဖြစ်ပါတယ်။ပို ပြီးသိသာအောင်သိချင်တယ်ဆိုရင်။

```
i =0
while i < 10:
     i +=1
     if i == 5:

         continue
         print("Contiue")

     print(i)
```

continue အောက်က print ကလည်း အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ဒါကိုကြည့်ခြင်းအားဖြင့် continue က သူ့ အောက်က statement တွေကိုအလုပ်လုပ်စေမှာမဟုတ်ဘဲ ကျော်သွားစေနိုင်တယ်ဆိုတာကို သိနိုင် ပါတယ်။

Pass

နောက်ဆုံးတစ်ခုကတော့ Pass ဖြစ်ပါတယ်။Pass ကတော့ Loop ရဲ့မူလအလုပ်ကို မပျက်စေဘဲနဲ့တခြား အလုပ်တစ်ခုကို ကြားထဲမှာလုပ်စေချင်ရင် pass ကိုအသုံးပြုနိုင်ပါတယ်။

```
i =0
while i < 10:
     i +=1
```

```
    if i == 5:

        pass
        print("I am Pass")

    print(i)
```

ဒီ Program မှာဘဲကြည့်လိုက်ပါ Loop ရဲ့လုပ်ငန်းစဉ်တွေ ရပ်တန့်တာမရှိပါဘူး ဒါပေမယ့် 5 ကိုရောက်တဲ့အခါမှာတော့ I am Pass လို့ Program က ပေါ်လာပါတယ်။

For Loop

နောက်ထပ် Loop တမျိုးကတော့ အသုံးများဆုံးနဲ့မတွေ့ချင်အဆုံးတွေ့ရမယ့် For Looping ဖြစ်ပါတယ်။for Looping ကိုမပြောခင် range() ဆိုတဲ့ function အကြောင်းကို အရင်ဆုံးပြောဖို့လိုပါတယ်။range ဆိုတာကတော့ ကိန်းဂဏန်းတွေသူ့အလိုလျှောက်ထုတ်ပေးတဲ့ Function မျိုးဖြစ်ပါတယ်။စမ်းသက်ကြည့်ရ အောင်။IDLE ဖွင့်မယ်။

```
tuple(range(9))
```

```
(0, 1, 2, 3, 4, 5, 6, 7, 8)
```

range(9) လို့ရေးလိုက်တဲ့အခါမှာ 0 ကနေ 8 အထိ ကိန်းစဉ်ပေါ်လာပါတယ်။အရင် python 2.X.X တုန်းကတော့ range(9) လို့ရေးလိုက်တာနဲ့တိုက်ရိုက်ပေါ်ပါတတယ် အခု Python 3.X.X မှာတော့ tuple ပြောင်းပေးမှ ကိန်းစဉ်တန်းကြီးပေါ်လာမှာပါ။ဒီ range() ရေးနည်းက Slicing လုပ်တဲ့နည်းနဲ့ အတူတူပါဘဲ။နောက်ဆုံးအနေနဲ့ 9 လို့ရေးရင် 9 မပါဘဲ 8 ထိဘဲပါပါလိမ့်မယ်။

```
tuple(range(1,9))
```

```
(1, 2, 3, 4, 5, 6, 7, 8)
```

စမှတ် 1 လို့သတ်မှတ်ပေးပြီး 9 လို့သတ်မှတ်လိုက်တဲ့အခါမှာ 1 ပါဝင်လာပြီး ဆုံးမှတ် 9 ကတော့ပါမှာမဟုတ်ပါဘူး တစ်လုံးလျော့ပြီး အလုပ်လုပ်တာသူ့ရဲ့သဘာဝဘဲဖြစ်ပါတယ်။နောက်ထပ် range() ကို increase ပုံစံ မျိုးနဲ့သုံးလို့ရပါသေးတယ်။

```
tuple(range(1,11,2))
```

```
(1, 3, 5, 7, 9)
```

Range မှာ (start,end,increase) ဆိုတဲ့ပုံစံနဲ့သွားတာပါ increase လုပ်ရရင် decrease လည်းလုပ်ရတာပေါ့ဗျာ။1 ဆိုတော့ 1 ကနေစတယ် end မှာ 11 ဆိုတော့ 10 မှာဆုံးမယ်။increase ကတော့ 2 ဆိုတော့ နှစ်ခု နှစ်ခု

တိုးမှာဖြစ်ပါတယ် ဒီမှာ 10 မှာဆုံးရမယ်ဖြစ်ပြီး 9 မှာဘာလို့ဆုံးတာလဲဆိုတော့ 2 တိုးတာဆိုတော့ 9 မှတင်ရပ်မှ 10 မကျော်မှာဖြစ်ပါတယ်။ဒါဆိုရင် decrease လုပ်ကြည့်ရအောင်။

```
tuple(range(10,-1,-1))
```

```
(10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)
```

အခုဆိုရင် 0 မှာဆုံးချင်တဲ့အတွက် -1 မှာအဆုံးသတ်ထားတာတွေ့ရမှာပါ။လျော့သွားမယ်အရေတွက်ကိုလည်း -1 လို့ပေးထားပါတယ်။စမှတ်ကတော့ 10 ဖြစ်လို့ 10 ကနေစပြီးလျော့လာမှာဖြစ်ပါတယ်။

ဒါဆို For Looping ကိုသွားလို့ရပါပြီ၊For ကိုရေးဖို့အတွက် Loop လုပ်မယ့် variable တလုံးပါဂင်ပြီး in ဆိုတဲ့စာလုံးလေးနဲ့ sequence တွေကို pass လုပ်ပြီး အလုပ်လုပ်ရပါတယ်။

```
for iterating_var in sequence:
   statements(s)
```

For Looping ကို အထက်ပါပုံအတိုင်းရေးပါတယ်။လက်တွေ့ရေးကြည့်ရအောင်။

```
for var_x in range(1,6):
    print(var_x)
```

ဒီမှာကြည့်လိုက်ပါ var_x ထဲကို in နဲ့ range(1,6) ဆိုတော့ 1 ကနေ 5 ထိကိန်းစဉ်တွေကို var_x ထဲကိုထည့်လိုက်ပါတယ်။ပြီးတော့ အဖြေကို print နဲ့ထုတ်လိုက်တဲ့အခါမှာ Loop ပတ်တဲ့အတိုင်း အစဉ်လိုက် 1 ကနေ 5 ကိုပေါ်လာမှာဖြစ်ပါတယ်။

String နဲ့ For Loop ကိုလဲတွဲဖက်အသုံးပြုနိုင်ပါတယ်။

```
for var_x in "Python":
    print(var_x)
```

ဒါကတော့ String နဲ့ For Loop နဲ့တွဲဖက်အသုံးပြုပုံဖြစ်ပါတယ်။P y t h on ဆိုပြီး တလုံးချင်း Loop ပတ်ပြီးထွက်လာမှာဖြစ်ပါတယ်။

List သို့မဟုတ် Tuple တွေနဲ့ For Loop ကိုတွဲဖက်အသုံးပြုနိုင်ပါတယ်။

```
var_i = ["Apple","Banana","Pieapple","Orange","Lime"]

for var_x in var_i:
    print(var_x)
```

ဒါကတော့ List နဲ့တွဲဖက်အသုံးပြုထားတာဖြစ်ပါတယ်။

```
var_i = ("Apple","Banana","Pieapple","Orange","Lime")

for var_x in var_i:
    print(var_x)
```

ဒါကတော့ Tuple နဲ့တွဲဖက်အသုံးပြုပြထားတာဖြစ်ပါတယ်။List က ထောင့်ကွင်း square bracket နဲ့ ရေးပြီး တော့ Tuple ကတော့ Round Bracket လက်သည်းကွင်းနဲ့ရေးပါတယ်။အရှေ့က List နဲ့ Tuple တွေ အကြောင်းမမှတ်မိရင်ပြန်ကြည့်ပါ။

နောက်တစ်ခုကတော့ len() နဲ့ range() ကိုတွဲဖက်အသုံးပြုတာ အနည်းငယ်ပြောချင်ပါတယ်။ဒီအကြောင်း ကို ပြောပြီးတဲ့အချိန်မှ ကျွန်တော် တို့ dictionary တွေနဲ့ For နဲ့တွဲဖက်အသုံးပြုပြမှာဖြစ်ပါတယ်။

```
food_info
={"Food_1":"Apple","Food_2":"Banana","Food_3":"Pieapple","Food_4":"Mango","Food_5":"Or
ange","Food_6":"Lime"}
index_num1 = range(len(dict.keys(food_info)))

for var_x in index_num1:
    print(var_x)
```

ဒီ Program မှာ Dictionary တစ်ခုတည်ဆောက်ထားပြီးအဲဒီ Dictionary ထဲက Key တွေရဲ့ အရေအတွက် ကို ရယူဖို့မှာဖြစ်ပါတယ်။ဒါကြောင့် variable တစ်ခုတည်ဆောက်လိုက်ပါတယ် index_num1 ဆိုပြီးတော့ နောက် ပြီး dict.keys() ဆိုတဲ့ Function က dictionary ထဲက keys တွေကိုထုတ်ယူတဲ့ function ဖြစ်ပါတယ်။ဒီ function နဲ့ keys တွေကိုထုတ်ယူလိုက်ပြီး len() ထဲမှာထည့်ပြီး အရေအတွက်ထုတ်ယူလိုက်ပါတယ်။ပြီးတဲ့ အခါမှာ range() ထဲထပ်ထည့်ပြီး ဘယ် အရေတွက်ကိုထုတ်ယူလိုက်ပါတယ်။ပြီးတဲ့အခါမှာ for နဲ့ loop ပတ် ပြီးထုတ်လိုက်ပါတယ်ဒီတော့ ကိန်းစဉ်အစဉ်လိုက်တိုင်းပေါ်ပြီပေါ့။ဒါဆိုရင် food_info ဆိုတာကို အစဉ်လိုက်လေး Loop လုပ်ပြီး အဖြေထုတ်တဲ့ Program ရေးပါမယ်။

```
food_info
={"Food_1":"Apple","Food_2":"Banana","Food_3":"Pieapple","Food_4":"Mango","Food_5":"Or
ange","Food_6":"Lime"}
index_num1 = range(len(dict.keys(food_info)))
key_list =list(dict.keys(food_info))

print("Food Name"+"\t"+"Food List")

for var_x in index_num1:
    print(key_list[var_x]+"\t\t"+food_info[key_list[var_x]])
```

ဒီ Program မှာ food_info ဆိုတဲ့ dictionary တစ်ခုပါဝင်ပါတယ် အဲဒီထဲမှာကတော့ Keys နဲ့ Values တွေ ပါဝင်ပါတယ်။အဲဒီ Keys နဲ့ Values တွေကိုတွဲဖက်ပြီး အဖြေပြချင်တာဖြစ်ပါတယ်။အဲဒီအတွက် အရင်ဆုံး keys အရေတွက်ကို ထုတ်ယူပါမယ် ဒါကြောင့် variable တလုံးတည်ဆောက်ပြီး range(len(dict.keys(food_info)) ကိုအသုံးပြုပြီး အရေတွက်ထုတ်ယူပါတယ်။ပြီးတဲ့အခါမှာ dictionary ရဲ့ Key တွေကို Lists အဖြစ် ထုတ်ယူ ဖို့ အတွက် list() ကိုအသုံးပြုပြီးထုတ်ယူပါတယ်။ဒါဆိုရင်အားလုံးပြင်ဆင်ပြီးပြီလို့ဆိုရမှာ နောက်ပြီး for နဲ့ var_x ထဲကို index_num1 ရဲ့ dictionary keys အရေအတွက်တွေကို pass လုပ်လိုက်ပါတယ်။ပြီးတဲ့အခါမှာ အဖြေ ထုတ်တဲ့အခါမှာ keys အတွက် keys_list[var_x] ကိုအဖြေထုတ်ပြီး values အတွက်တော့ food_info[keys_list[var_x]] နဲ့အဖြေထုတ်ပါတယ်။\t \t ကတော့ Tab နှစ်ချက်ခုန်တာပါ။နည်းနည်းရှုပ်တယ် ထင်ရပေမယ့် အရှေ့က လုပ်ဆောင်ခဲ့တာတွေအကုန်နားလည်ရင်တော့ အလွယ်တကူရှင်းလင်းမှာပါ။သေချာ အချိန်ယူပြီးစဉ်းစားရင်နားလည်မယ်လို့မျှော်လင့်ပါတယ်။

Nested Loop

Loop ထဲမှာ Loop ထပ်ပတ်တာကို Nested Loop လို့ခေါ်ပါတယ် တချို့သော ကိစ္စရပ်တွေမှာ Loop ထဲမှာ Loop ပြန်ပတ်မှသာ အဆင်ပြေတာမျိုးရှိပါတယ်။ဒါကြောင့် Nested Loop ကိုအသုံးပြုရပါတယ်။ဒါဆို Nested Loop ပုံစံကြည့်ရအောင်။

```
for num in range(10,20):
    for i in range(2,num):
        if num%i ==0:
            j = num/i
            print("%d * %d equal %d"%(i,j,num))
            break
    else:
        print(num,"is a prime number")
```

ဒီပုဒ်စာမှာ Nested Loop သုံးထားပါတယ်။အလုပ်လုပ်ပုံကိုပြောရမယ်ဆိုရင် num ဆိုတဲ့ variable ထဲကို for loop နဲ့ range(10,20) ဆိုတော့ 10 ကနေ 19 ထိ ပတ်ခိုင်းလိုက်တာပေါ့။နောက် Loop ထဲမှာတော့ 2 ကနေ စောစောကနေ 10 ကနေ 19 ထိ passing လုပ်မယ့် num ကိုထည့်ထားပါတယ်။နောက်ထပ်လုပ်ထားတဲ့ လုပ်ဆောင်ချက်ကတော့ if နဲ့ num%i ဆိုတော့ num ကို i နဲ့စားလို့အကြွင်း 0 ဖြစ်ရင် numကို i နဲ့စားပြီး j ထဲကိုသိမ်းဆည်းပါတယ်။ပြီးတော့ print နဲ့ i * j equal num ဆိုပြီးအဖြေထုတ်ထားပါတယ်။ပြီးရင်တော့ break ဆိုပြီး Loop ကို break လုပ်လိုက်ပါတယ်။ဒါကြောင့်ဘာအလုပ်မှဆက်မလုပ်ဘဲ ဒီ Loop ဒီမှာဘဲ ရပ်တန့်ပါတယ်။တစ်ခုသိရမှာက else က ဒုတိယ for loop နဲ့ဆိုင်တာဖြစ်ပါတယ်။if နဲ့စစ်ဆေးတဲ့ num%i == 0 အခြေနေမမှန်ဘူးဆိုရင် else ထဲကိုသွားပြီး num ကို prime အဖြစ်အဖြေထုတ်မှာဖြစ်ပါတယ်။ဒါဆိုအလုပ်လုပ်ပုံလေးနည်းနည်းပြောရအောင်။num ထဲကို 10 ကနေ 19 ထိ ထည့် Loop လုပ်တယ် နောက်ထပ် အဲဒီ 10 က နေ 19 ကိုတစ်ခုခြင်းနောက် Loop တစ်ခုနဲ့ပတ်ထားတယ် 2 ကနေ အဲဒီ 10 ကနေ 19 ထိ num ကိုပေါ့။အခု program စစခြင်းမှာ num က 10 ဂင်လာပါတယ် အဲဒီတော့ ဒုတိယ Loop မှာ 2 ကနေ 10 ထိ Loop လုပ်မှာပေါ့ ဒါပေမယ့် 10%2 က 0 ရတော့ ပြတ်တယ်ပေါ့ဗျာ အဲဒီတော့ j = 10/2 အလုပ်ကိုလည်းလုပ်တယ် j က 5 ရတာ ပေါ့။အဲဒီတော့ print ထဲမှာ ရှိတဲ့အတိုင်းဘဲ i * j equal num ဆိုတော့ i က 2 j က 5 num က 10 ပေါ့ဗျာ အဲဒီတော့ 2 * 5 equal 10 ဖြစ်ပါတယ် ပြီးတော့ break ဖြစ်သွားလို့ အဲဒီ Loop တစ်ခုလုံးရပ်တန့်သွားပါတယ်။ဒီလိုနဲ့ num က 11 ဖြစ်လာတဲ့အခါမှာ if အခြေနေနဲ့ num%i ==0 ဖြစ်ဖို့ကို 2 ကနေ 10 ထိ စစ်ဆေး ပေမယ့် မရှိတဲ့အတွက် ဘာနဲ့မှမပြတ်လို့ else ထဲကိုရောက်သွားပြီး prime ဖြစ်ကြောင်းပြောပါတယ် ဒီလိုနဲ့ဘဲ program က 19 ထိအလုပ်လုပ်တာဖြစ်ပါတယ်။

Loop Control Statement

For Loop အတွက် Loop Control Statement တွေအသုံးပြုလို့ရပါတယ်။break , continue , pass တို့ဘဲဖြစ်ပါတယ် အသုံးပြုပုံကလည်း အတူတူဘဲဖြစ်ပါတယ်။While Loop မှာလုပ်တဲ့လုပ်ဆောင်ချက် နဲ့ ထပ်တူညီပါတယ်။

Break with For Loop

Break နဲ့For Loop ကိုတွဲဖက်အသုံးပြုကြည့်ရအောင်။Break ဆိုတာ Loop တစ်ခုလုံးရဲ့အလုပ်ကို ရပ်စေတယ်ဆိုတာ သိထားရမှာပါတယ်။

```
for x in "Python":

    if x == "h":
        break

    print(x)
```

ဒီ program မှာ variable x ထဲကို Python ဆိုတဲ့ squences တစ်ခုကို Loop လုပ်ထားပါတယ် ဒါပေမယ့် ကျွန်တော်က x == "h" နဲ့ h ကိုရောက်တဲ့အခါမှာ break လို့ရပ်တန့်ထားပါတယ် ဒါကြောင့် P y t ပြီးရင် ဆက် အလုပ်တွေလုပ်တော့မှာမဟုတ်တော့ပါဘူး။

Continue With For Loop

Continue ကတော့ သူ့မှာရှိတဲ့အလုပ်ကို reject လုပ်သွားစေပါတယ်။ဒါကြောင့် continue ထည့်လိုက်ရင် သူ့ရဲ့အလုပ်ကမလုပ်စေတော့ပါဘူး ဒါပေမယ့် Looping ရပ်တန့်သွားတာတော့မဟုတ်ပါဘူး အခြေနေတစ်ခုကို ကျော်လိုက်တဲ့သဘောပါဘဲ။

```
for x in "Python":

    if x == "h":
        continue

    print(x)
```

ဒီ program မှာဆိုရင် x က h နဲ့ညီတဲ့အခါမှာ continue လုပ်မယ်လို့ပြောလိုက်တဲ့အခါမှာ h ကိုကျော်သွားပြီးတော့ P y t o n ဆိုပြီးအဖြေထုတ်သွားပါတယ် ဒါကိုကြည့်ခြင်းအားဖြင့် Loop ရပ်သွားတာမဟုတ်ဘဲ အခြေနေတစ်ခုကိုဘဲ ဖယ်ထုတ်လိုက်တာဖြစ်ပါတယ်။

Pass With For Loop

Pass ကတော့သိတဲ့အတိုင်းဘဲ ကြားထဲမှာ အလုပ်တစ်ခုခုကိုလုပ်စေချင်ရင် pass နဲ့သုံးတာ ပါ ဒါပေမယ့် သူက null operation ဖြစ်ပါတယ် run နေတဲ့ code ကို ဘာအနှောက်ယှက်မှမဖြစ်စေပါဘူး။

```
for x in "Python":
    print(x)
    if(x == "o"):
        pass
        print("Hello I am Pass")
```

ဒီမှာဆိုရင် P y t h o Hello I am Pass n ဆိုပြီးအဖြေထုတ်ပါလိမ့်မယ် သူ့ရဲ့ လုပ်ငန်းစဉ်ကိုလည်း မထိခိုက်စေဘဲ ကျွန်တော်တို့လုပ်ချင်တဲ့အလုပ်ကိုလဲလုပ်ပါတယ်။

Functions

Function ဆိုတာ Block of code ဘဲဖြစ်ပါတယ်။သူထဲမှာ ကျွန်တော်တို့လုပ်ချင်တဲ့ အလုပ်တွေကိုစုစည်းရေးသားထားတဲ့အရာမျိုးဖြစ်ပါတယ်။functions တွေကိုတော့ အကျွမ်းတဝင်ရှိခဲ့ဖူးပါတယ်။ဥပမာ- print() , len() ,range() လိုဟာမျိုးပေါ့။ဒါကို python မှာ built in ပါတဲ့ နဂိုမူလ ပါတဲ့ functions တွေလို့ခေါ်ပြီး ကျွန်တော်တို့ကိုယ်တိုင်လုပ်ချင်တဲ့အလုပ်တွေကို လုပ်ဖို့အတွက် တည်ဆောက်မယ့် Functions တွေကိုတော့ User-define Function လို့ခေါ်ပါတယ်။ဘာကြောင့် Functions တွေသုံးရလဲဆိုတော့ ထပ်ခါထပ်ခါလုပ်ရမယ့်အလုပ်တွေ ခေါ်သုံးရမယ့် အရာတွေကို function လေးခေါ်သုံးရုံနဲ့အလုပ်လုပ်နိုင်တဲ့အတွက် ထပ်ခါထပ်ခါပြန်ပြီးရေးနေစရာမလိုတော့ဘူးပေါ့ဗျာ။

Functions တွေကို ဘယ်လိုတည်ဆောက်လဲဆိုတဲ့အကြောင်းပြောရအောင် Functions တွေကို တည်ဆောက်တဲ့အခါမှာ def ဆိုတဲ့ စာလုံးကိုအရင်ရေးရမယ်ပြီးရင် ကိုယ်ကြိုက်တဲ့ Function Name ပေးနိုင်တယ် ပြီးတဲ့အခါမှာတော့ Round Bracket ခေါ် လက်သည်းကွင်း ဒီမှာတော့ parenthese () ပါရမယ် ပြီးရင် colon (:) ဆိုတဲ့ အစက်လေးနှစ်စက်ပါရမယ် ဒါက Function တည်ဆောက်ပုံဖြစ်ပါတယ်။နောက်တမျိုးကတော့ parenthese () ထဲမှာ parameters or arguments ပါတဲ့ Functions အမျိုးအစားရှိတယ် မပါတဲ့ အမျိုးအစား တွေလဲရှိတယ်။နောက်တစ်ခုက Function ရဲ့ပထမ statement ကို docstring အဖြစ်မှတ်ချက် တွေ ဘာညာရေးဖို့အသုံးပြုလို့ရတယ်။နောက်တစ်ခုက Functions တွေက return ပြန်နိုင်တယ်။ဒါက အခြေခံအချက်တွေပါ တည်ဆောက်ပြီးသေချာပြောပြတဲ့အခါမှ ဘာက return ဘာက parameters ဘာက docstring ဆိုတာ သေချာသိရမှာပါ။

```
def fun():
    print("Hello I am Functions")
```

ဒါက fun ဆိုတဲ့ နာမည်နဲ့ တည်ဆောက်ထားတဲ့ Function တစ်ခုဘဲဖြစ်ပါတယ် သူ့မှာ parameters မပါပါဘူး ဘာကိုကြည့်ပြီးပြောတာလဲဆိုရင် parenthese () ထဲမှာ ဘာမှမပါဘဲ အလွတ်ဖြစ်နေလို့ပါ။ဒီထဲမှာ လုပ်ထားတဲ့လုပ်ဆောင်ချက်က တော့ ရိုးစင်းပါတယ် print နဲ့ Hello I am Functions ဆိုတဲ့စာသားကို အဖြေထုတ်တာဘဲဖြစ်ပါတယ်။Run ကြည့်လိုက်ပါ။ဘာမှပေါ်မှာ မဟုတ်ပါဘူး :D ဘာလို့လဲဆိုရင် Functions တွေက ကြိုက်သလောက်တည်ဆောက်ပါစေ သူတို့ကိုမခေါ်ရင် အလုပ်လုပ်မှာမဟုတ်ပါဘူး ဒါကြောင့် အလုပ်လုပ်ဖို့ ခေါ်ပေးရပါတယ်။

```
def fun():
    print("Hello I am Functions")


fun()
```

function name ကို parethese နဲ့တွဲပြီး fun() ဆိုတာ က function ကိုခေါ်လိုက်တာပါဘဲ။ဒီလို ခေါ်လိုက် တဲ့ အတွက် သူ့ထဲမှာပါတဲ့ အလုပ်ဖြစ်တဲ့ print() က အလုပ်လုပ်သွားပါတယ်။နောက်ထပ် parameters တွေပါ တဲ့ Functions တွေတည်ဆောက်ကြည့်ပါမယ်။

```
def calc(num1,num2):
    "function with Parameter"
    total = num1 + num2
    print(total)
```

ဒီမှာ num1 နဲ့ num2 ဆိုပြီး parameters နှစ်ခုကို Function မှာထည့်သွင်းတည်ဆောက်ထားတာတွေ့ ရမှာ ဖြစ်ပါတယ်။အဲဒီထဲမှာပါတဲ့ function with parameter ဆိုတာကတော့ docstring ဘဲဖြစ်ပါတယ် သူက program အလုပ်လုပ်တဲ့နေရာမှာ ပြသာနာ မပေးနိုင်ပါဘူး။ဒီထဲမှာလုပ်တဲ့အလုပ်က parameters နှစ်ခု ထဲကို passing လုပ်ပြီးပေါင်းပါမယ် ပြီးတော့ total ဆိုတဲ့ variable ထဲကိုထည့်ပြီး အဖြေပြန်ထုတ်တဲ့ အခါမှာ လည်း total ကိုဘဲ အဖြေထုတ်ပို့မယ်။ဒီတော့ ဒီတစ်ခေါက် function က parameters ပါဝင်တာမို့ ခေါ်သုံးတဲ့ အခါမှာ လည်း parameter ပါဝင်ပါတယ်။

```
def calc(num1,num2):
    "function with Parameter"
    total = num1 + num2
    print(total)


calc(3,4)
```

ဒီလိုဆိုရင် 3 နဲ့ 4 နဲ့ပေါင်းတဲ့ 7 ကို အဖြေအဖြစ်ရမှာဖြစ်ပါတယ်။နောက်ထပ် return ပြန်တယ်ဆိုတာ ဘယ်လို လဲဆိုတာကိုဖော်ပြပို့မယ်။

```
def calc():
    total = 1000 +2000
    return total
```

ဒီ Function မှာ return ပြန်ထားတာတွေ့ရမှာပါ 1000 + 2000 ကို total ထဲ ထည့်ပြီး return ပြန်ထား ပါတယ် ဒီ function မှာ print ထုတ်ထားတာမပါပါဘူး။ဒါပေမယ့် return total လို့ရေးထားတဲ့အတွက် total ရဲ့ အဖြေ က calc() ဆိုတဲ့ function ထဲကိုဝင်နေပါပြီ။လက်တွေ့စမ်းကြည့်ရအောင်။

```
def calc():
    total = 1000 +2000
    return total

print(calc())
```

ဒီလိုရေးလိုက်တော့ 3000 လို့အဖြေထွက်ပါတယ် ဒါကဘာကိုပြောတာလဲဆိုတော့ calc() က return ပြန် လုပ် ထားလို့ total ရဲ့အဖြေကိုပြန်ရနေတယ်ဆိုတာပါဘဲ return က အရမ်းအသုံးဝင်ပါတယ် control statement တွေနဲ့သုံးတဲ့အခါမှာ တကယ်အသုံးဝင်ပါတယ်။

```
def cont():
    return True
```

```
if cont():
    print("That is True")

else:
    print("That is False")
```

ဒီမှာကြည့်လိုက်တဲ့ အခါမှာ cont() ဆိုတဲ့ function က true ကို return ပြန်ထားတဲ့အတွက် cont() က လည်း true ဖြစ်နေပြီပေါ့။အဲဒီတော့ if နဲ့ cont(): ဆိုပြီး စစ်ဆေးတော့ if အခြေနေမှာ True ဖြစ်လို့ That is True လို့ပေါ်မှာဖြစ်ပါတယ်။

Setting a Default Parameter

ကျွန်တော်တို့ parameters တွေပါဝင်တဲ့ function တွေကို ခေါ်သုံးတဲ့အခါမှာ parameters value တွေ passing မလုပ်ပေးဘူးဆိုရင် error တတ်ပါလိမ့်မယ်။

```
def add(name,passwod):
    print(name,passwod)


add()
```

ဒီမှာဆိုရင် parameters တွေထည့်ရမှာဖြစ်ပါတယ် မထည့်ဘဲ ဒီတိုင်း Function ကိုခေါ်မိတဲ့အတွက် error တွေတတ်ပါလိမ့်မယ်။ဒီအတွက် default Parameter ထည့်ပေးဖို့ သို့မဟုတ် အသုံးပြုပေးဖို့လိုပါတယ် ဒီလိုသုံးကြည့်ရအောင်။

```
def add(name="Admin",passwod="admin"):
    print(name,passwod)


add()
```

ဒီလို Default Parameters ထည့်ပေးတော့ function ခေါ်တဲ့အခါဘာမှမပါရင်လဲ Error မတတ်တော့ပါဘူး value တွေ pass လုပ်ရင်လဲ ရပါတယ်။

```
def add(name="Admin",passwod="admin"):
    print(name,passwod)


add("AungMoeKyaw","1234")
```

ဒီလို pass လုပ်တဲ့အခါမှာ လည်း pass လုပ်တဲ့ value အတိုင်းရပါတယ်။If နဲ့ functions နဲ့ တွဲဖက်အသုံး ပြုကြည့်ရအောင်ဗျာ။

```
def color(colour):
    if colour =="red":
        return "That is red"
    elif colour == "blue":
        return "That is Blue"
    elif colour == "Yellow":
        return "That is Yellow"
    elif colour == "Green":
        return "That is Green"
    else:
        return "Error Input"
```

```
user_des = input("Enter Colour:")
print(color(user_des))
```

ဒီ program မှာ function တစ်ခုတည်ဆောက်ထားပြီးတော့ if နဲ့ passing လုပ်ထားတဲ့ colour ကို စစ်ဆေးပါတယ် red ဆို ရင် That is red ဆိုတာကို return ပြန်ပါမယ်။အဲဒါကို user_des ဆိုတဲ့ variable တလုံးကိုတည်ဆောက်ပြီး input() ယူထားပါတယ် ပြီးတဲ့ အခါမှာ print နဲ့ color(user_des) နဲ့passing လုပ်တယ် ပြီးတော့ return ပြန်လာတဲ့အဖြေကို print ထုတ်တယ်ပေါ့ဗျာ။

```
def var(list_d):
    list_1 = list_d

    for x in list_1:
        print(x)


vari = ["a","b","c","d","e","f","g"]

var(vari)
```

ဒါကတော့ For Loop ရယ် List ရယ်နဲ့ Function ရယ်ပေါင်းသုံးတဲ့ program ပါ။function မှာ parameter ထည့်ထားပါတယ် အဲဒီ parameter ကို variable ထဲထည့်လိုက်ပါတယ်။တကယ်တော့မလိုအပ်ဘူးနော် မြင်သာ အောင်ပြတာဘဲ တိုက်ရိုက်သုံးလို့ရတယ်။ပြီးတော့ for နဲ့ x ထဲကို အဲဒီ variable ထည့်ပြီး Looping ပတ်ပါတယ်ပြီးတဲ့အခါမှာ x ကို အဖြေပြန်ထုတ်ထားပါတယ်။function ကိုခေါ်သုံးတဲ့အခါမှာတော့ ကျွန်တော်တို့ တည်ဆောက်ထားတဲ့ List လေးကို passing လုပ်ပေးလိုက်ခြင်းအားဖြင့် ပုံမှန် Looping တွေလိုဘဲ Loop လုပ်လေသည်။

နောက်ထပ် ထပ်ပြောမှာကတော့ Special Argument လို့ခေါ်တဲ့ Tuple Argument နဲ့ Dictionary Argument နှစ်မျိုးကိုဖော်ပြပါမယ်။passing လုပ်ပေးတဲ့ data တွေကို Tuple တွေအဖြစ်ရယူချင်တဲ့ အခါမှာ Special Argument ဖြစ်တဲ့ *args ကိုအသုံးပြုရပါတယ်။လက်တွေ့ကြည့်ရအောင်။

```
def sa(*args):
    print(args)


sa(1,2,"air","car",0.11,"A")
```

ဒီမှာ Passing လုပ်တဲ့ Data တွေက အများကြီးဖြစ်ပါတယ်။လက်ခံဖို့ဆိုရင် passing လုပ်လိုက်တဲ့ တန်ဖိုးအရေအတွက်တိုင်းလက်ခံရမှာပါ ဒါပေမယ့်အခု special argument လို့ခေါ်တဲ့ ရှေ့မှာ asterisk (*) ခံထားတဲ့ argument လေးကြောင့် value တွေဘယ်နှစ်ခု passing လုပ်လုပ် အဆင်ပြေပြေလက်ခံနိုင်မှာပါ Tuple Argument လို့ဘာကြောင့်ခေါ်ရလဲဆိုရင် Tuple လိုသိမ်းဆည်းလိုက်လို့ဖြစ်ပါတယ်။ဒါခုသိထားဖို့ လိုပါတယ် asterisk (*) နောက်က နာမည်မှာ args မဟုတ်ဘဲ တခြားနာမည်လဲဖြစ်လို့ရပါတယ် Function မှာ ပြန်ခေါ်တဲ့အချိန်ကြရင်တော့ asterisk(*) ဖြုတ်ပြီးနောက်ကနာမည်ကိုပြန်ရေးပေးရင်ရပါတယ်။

```
def sa(*var):
    for x in var:
        print(x)


sa(1,2,"air","car",0.11,"A")
```

ဒါကတော့ Tuple Argument နဲ့ For နဲ့ပေါင်းသုံးထားတဲ့ ပုံစံဖြစ်ပါတယ်။ဒီမှာ asterisk(*) နောက်မှာ ကြိုက်တဲ့ နာမည်ပေးပြီးသုံးထားတာတွေ့ရမှာပါ။ဒါဆိုရင်ဒီ special argument က တခြားသော normal argument or parameter တွေနဲ့ရောသုံးနိုင်လား၊သုံးနိုင်ပါတယ် လက်တွေ့တွဲသုံးကြည့်ရအောင်။

```
def sa(a,b,c,*var):
    print("This is Parameter a:",a)
    print("This is Parameter b:",b)
    print("This is Parameter c:",c)
    print("Tuple Argument:",var)

sa(1,2,"air","car",0.11,"A")
```

ဒီ program မှာဆိုရင် parameter တွေအနေနဲ့ a , b , c နဲ့ tuple argument *var ပါဝင်ပါတယ် passing လုပ် လိုက်တဲ့ data တွေအနေနဲ့ ၆ ခုရှိပါတယ် အဲဒီ ၆ ခုမှာ ရှေ့သုံးခုက parameter ဖြစ်တဲ့ a , b ,c အတွက် ဖြစ်သွားပြီး ကျန်တဲ့ ၃ လုံးကတော့ tuple ဖြစ်သွားပါတယ် ဘာကြောင့်လဲဆိုတော့ tuple argument က ရယူသွားလို့ဖြစ်ပါတယ်။ဒါဆိုကျန်တဲ့ Dictionary Argument ကိုဆက်သွားပါတယ် နောက်တစ်နည်းသူ့ကို Keyword Argument လို့ခေါ်ပါသေးတယ်။ဒီမှာတော့ asterisk နှစ်လုံးနဲ့တည်ဆောက်ပါတယ် (**) ပေါ့ အများ ဆုံးရေးတာကတော့ **kwargs လို့ရေးပါတယ် Keyword argument ရဲ့အတိုကောက်ပေါ့ဗျာ။ဒါကို ဘယ်လိုသုံးလဲကြည့်ရအောင်။

```
def sa(**kwargs):

   print(kwargs['one'],kwargs['two'],kwargs['three'])

sa(one=1,two=2,three=5)
```

တန်ဖိုးတွေကို သိုလှောင်မှုပုံစံနဲ့ နှစ်ခု passing လုပ်တဲ့အခါမှာ kwargs ကို အသုံးပြုခြင်းအားဖြင့် dictionary ပုံစံနဲ့သိမ်းဆည်းနိုင်ပါတယ်။ဒီမှာမြင်ရတဲ့ပုံစံအရဆိုရင်တော့ passing လုပ်လိုက်တဲ့ one က key အဖြစ် kwargs ထဲကိုဝင်သွားပြီး 1 ကတော့ value ဖြစ်သွားပါတယ် kwargs ကတော့ dictionary ဖြစ်သွားပြီး ဒါကြောင့် အဖြေထုတ်တဲ့နေရာမှာ one ဆိုတဲ့ key ကိုထည့်ရင် 1 ဆိုတဲ့ value ကိုပြန်ရတာပေါ့။နောက်တစ်ခုလုပ်ကြည့်ရအောင်။

```
def sa(**kwargs):
   for x in kwargs:
      print(x)

sa(one=1,two=2,three=5)
```

ဒီ kwargs ကို x ထဲထည့်ပြီး for နဲ့ ပတ်ထားတဲ့အခါမှာ အဖြေထုတ်တဲ့အခါမှာ one , two , three ဆိုတဲ့ key တွေကိုတွေ့ရမှာပါ။ဒါဆိုရင် dictionary ကို Loop ပတ်ကြည့်ရအောင်။

```
dic={"one":1,"two":2,"three":3}

for y in dic:
   print(y)
```

ဒီ dic ဆိုတဲ့ dictionary ကို loop ပတ်တဲ့အခါမှာလည်း key တွေကိုဖော်ပြတာတွေ့ရမှာ ဖြစ်ပါတယ်။ဒါဆိုရင် ** နဲ့ kwargs က dictionary နဲ့အတူတူဘဲပေါ့။ကျွန်တော်တို့က key တွေမလိုချင်ဘဲ value တွေဘဲ လိုချင်တယ် ဆိုရင် ဘယ်လိုလုပ်မလဲ။

```
dic={"one":1,"two":2,"three":3}

for y in dic:
   print(dic[y])
```

dictionary မှာဆိုရင်တော့ အထက်ပါ အတိုင်းဘဲ dic ဆိုတဲ့ dictionary ကိုရေးပြီး [] ထဲမှာ key တွေ pass လုပ်ပေးလိုက်ရင် value ရမှာပေါ့။အပေါ်က kwargs မှာလည်း အတူတူဘဲဖြစ်ပါတယ်။

```
def sa(**kwargs):
   for x in kwargs:
      print(kwargs[x])

sa(one=1,two=2,three=5)
```

နောက်ထပ် Function ကိုတည်ဆောက်အသုံးပြုတဲ့အခါမှာ သိထားရမယ့်အချက်တစ်ချက်ပြောပါမယ်။ဒါ က ဘာလဲဆိုတော့။function ကိုခေါ်သုံးတဲ့နေရာမှာ မတည်ဆောက်ခင်ခေါ်သုံးလို့မရပါဘူးပြောချင်တာ ဒီလိုဗျာ function တည်ဆောက်ထားတဲ့အောက်မှာဘဲခေါ်သုံးလို့ရမယ် မဟုတ်ရင်သူက မတွေ့ဘူးဘဲပြောလိမ့်မယ်။

```
fun()
def fun():
   print("Hello I am Function")
```

တည်ဆောက်ထားတဲ့ function အပေါ်မှာ ခေါ်ထားတော့ error တတ်ပါတယ်။ဒီတော့ function ရဲ့ အောက်မှာဘဲ ခေါ်သုံးမှရမှာပါ။

```
def fun():
   print("Hello I am Function")

fun()
```

ဒီလိုခေါ်သုံးမှဘဲ အလုပ်လုပ်မှာပါ။ဒါပေမယ့် တစ်နည်းတော့ရှိပါတယ်။

```
def main():
   fun()

def fun():
   print("Hello I am Function")


main()
```

အထက်က အတိုင်းပါဘဲ ပင်တိုင် Function တစ်ခုကိုတည်ဆောက်ပြီး အဲဒီ Function ထဲမှာလှမ်းခေါ်ထားရင် ရပါတယ်။ဒါဟာဘာကြောင့်လဲဆိုရင် Function ကိုတွေ့တာနဲ့အလုပ်မလုပ်သေးဘဲ အောက်ဆုံးထိ program က ဖတ်သွားပါတယ်။ဒါကြောင့် Function ချင်းမှာ အပေါ်အောက်သိပ်မလိုဘဲ ရေးလိုရတာဖြစ်ပါတယ်။

The Anonymous Functions

Def ကိုမသုံးဘဲ သူ့ရဲ့ တည်ဆောက်နေကြပုံစံမဟုတ်ဘဲ lambda ကိုအသုံးပြုပြီး Function ပုံစံ တည်ဆောက်ခြင်းကို anonymous functions လို့ခေါ်ပါတယ်။

Lambda ကိုတည်ဆောက်မယ်ဆိုရင် lambda ဆိုတဲ့ စာလုံးပါရမယ်၊ပြီးတော့ ဘယ် Argument မဆို Function နဲ့အတူတူဘဲထည့်လို့ရတယ် ဒါပေမယ့် expression လို့ခေါ်တဲ့ လုပ်မယ့်အလုပ်ကတော့ အများကြီးလုပ်လို့မရဘူး အလုပ်တစ်ခုဘဲလုပ်လို့ရမယ်။

```
sum =lambda args,args1,*argg:args+args1

print(sum(110,20))
```

variable တစ်ခုတည်ဆောက်ပြီးတည်ဆောက်ပြီး lambda စာလုံးကိုရေးတယ် ပြီးတော့ argument တွေထည့်က လို့ရတယ်။ပြီးတော့ coloum (:) အနောက်က expression ခေါ် အလုပ်လုပ်မယ့် statement ကိုထည့်ရတယ်။ပြီးတော့ ခေါ်သုံးတဲ့အခါမှာ variable ကိုခေါ်ပြီး print(sum(110,20)) လို့ခေါ်လိုက်ပါတယ်။130 အဖြေပြန်ထွက်လာတဲ့အတွက် lambda က return သူ့အလိုလိုပြန်တယ်လို့ သဘောရပါတယ်။

Global and Local Variable

Functions အတွင်းထဲမှာတည်ဆောက်ထားတဲ့ variable တွေကို local variable လို့ခေါ်ပြီး Functions တွေရဲ့ အပြင်ဘက်မှာတည်ဆောက်ထားတဲ့ Variable တွေကိုတော့ global လို့ခေါ်ပါတယ်။

```
num = 0

def calc(fnum=0,snum=0):
    num = fnum + snum
    text = "Calculator"
    print(text,num)

def tex():
    print(num)

calc(20,49)
```

ဒီမှာကြည့်လိုက်ရင် Global ဖြစ်တဲ့ num ဆိုတဲ့ variable တစ်ခုကိုတည်ဆောက်ပါတယ်။ပြီးတဲ့အခါမှာ ကျွန်တော်တို့ Function 2 ခုတည်ဆောက်တယ် calc() ထဲမှာ text ဆိုတဲ့ variable ရယ် num ကို Local အနေနဲ့တည်ဆောက်ပါတယ်။calc(20,49) လို့ အဖြေထုတ်လိုက်တဲ့အခါမှာ Local variable ဖြစ်တဲ့ text နဲ့ num က ပေါင်းပြီး calculator 69 ဆိုတဲ့အဖြေထုတ်လာပြီးတော့ အောက်က function မှာ အပေါ်က num ဆိုတဲ့ global ကိုအဖြေထုတ်တော့ 0 ဘဲဖြစ်ပါတယ်။ဒီတော့ Local ဆိုတာ သူ့ Function အတွင်းမှာဘဲ ခေါ်လို့ရတယ် Function ပြင်ပ သို့မဟုတ် class ပြည်ပ မှာအသုံးပြုလို့မရနိုင်ပါဘူး။ဒါပေမယ့် Global ကတော့ ဘယ်က လှမ်းခေါ်ခေါ်အသုံးပြုလို့ရပါတယ်။

Modules

Module ဆိုတာကတော့ python object တစ်ခုပါဘဲ သူ့အထဲမှာ class တွေ functions တွေ expression တွေ statements တွေပါဝင်ပါတယ်။ပြောရရင်သူက ရိုးရိုး python program လိုမျိုးဘဲဖြစ်ပါတယ်။ module တွေက ထပ်ခါထပ်ခါ သုံးရမယ့် program မျိုးကို module အဖြစ်ရေးဆွဲထားပြီးတော့ program တွေ ရေးတဲ့အခါ အသင့်ခေါ်သုံးနိုင်အောင် ရေးဆွဲထားတဲ့သဘောပါ။module တွေကို နာမည် အမျိုးမျိုးနဲ့ပေးနိုင်ပါတယ်။python မှာ အသင့် တည်ဆောက်ပြီးသား modules တွေလည်းရှိပါတယ်။

ကျွန်တော်တို့ ကိုယ်ပိုင် Module တစ်ခုဖန်တီးကြည့်ရအောင်။အခု module လေးက Temperature ပြောင်းတဲ့ module လေးဖြစ်မယ် နာမည်ကို easytemp လို့ပေးမယ်။Fahrenheit to Celsius နဲ့ Celsius to Fahrenheit ကို အလွယ်တကူပြောင်းနိုင်တဲ့ Formula ပါတဲ့ Module ရေးပါ့မယ်။

ဒါဆိုရင် ပုံသေနည်းနည်းနည်းပြောရအောင် Celsius ကနေ Fahrenheit ပြောင်းတဲ့ အခါမှာ ပုံသေ နည်း အရ

F = 9/5(C)+32 ဖြစ်ပါတယ်။F ဆိုတာ Fahrenheit ဖြစ်ပြီး C ဆိုတာ Celsius ဖြစ်ပါတယ်။ဒီတော့ ကျွန်တော်တို့ တွက်ရမှာပေါ့ 9/5 = 1.8 အစားထိုးတွက်လို့ရပါတယ်။

```
def converte_f(cvalue):
    F = (1.8 * cvalue) + 32
    return F

def converte_c(fvalue):
    C = 0.5*(fvalue - 32)
    return C
```

ဒီမှာရေးထားတာကတော့ Functions ၂ ခုဖြစ်ပါတယ်။converte_f နဲ့ converte_c ပါ။converte_f မှာ fvalue လို့ခေါ်တဲ့ Fahrenheit တန်ဖိုးကိုလက်ခံယူထားပြီး variable F ကိုတည်ဆောက်ပြီး ပုံသေနည်း အတိုင်း တွက်ယူထားပါတယ် (1.8 * cvalue)+ 32 နောက်ပြီး ဒီ Function ကိုခေါ်ပြီး အသုံးပြုနိုင်အောင် return F လုပ်ထားပါတယ်။နောက်ထပ် function ကလည်း အတူတူပါဘဲ ဒါကတော့ fvalue ဆိုတဲ့ argument ကနေ Fahrenheit တန်ဖိုးကိုရယူပြီး Celsius ကိုပြောင်းပေးဖို့ ပုံသေနည်းသုံးထားတာဘဲဖြစ်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့က temperature ပြောင်းလဲတွက်ချက်ဖို့ easytemp.py ဆိုတဲ့ module ကိုရေးပြီးပါပြီ ခေါ်သုံးတော့မှာဖြစ်ပါတယ်။ခေါ်သုံးဖို့အတွက် လိုက်နာရမယ့်အချက်တွေရှိတယ်။ကြည့်ရအောင်ဘာတွေလဲဆိုတာ။

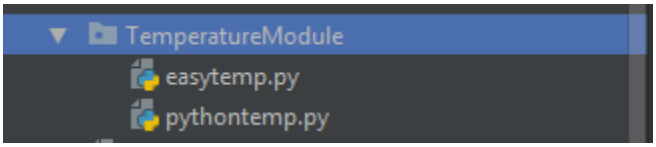
၁.Python Module File က သုံးမယ့် Program ဖိုင်နဲ့ Directory တူတူဖြစ်ရမယ်။ပြောချင်တာ Folder တူတူဖြစ်ရမယ်။

၂.ဒါမှမဟုတ်ရင် C: ထဲက C:\Users\user\AppData\Local\Programs\Python\Python36 ထဲက Lib ဆိုတဲ့ Folder ထဲမှာထည့်ပေးရမယ်။

၃.ဒီလိုမှမဟုတ်ဘဲခေါ်သုံးချင်တယ်ဆိုရင် From ဆိုတဲ့ Keyword သုံးပြီး နေရာကိုဖော်ပြပြီးအသုံးပြုနိုင်ပါတယ်။

```
import easytemp

ce = easytemp.converte_c(95)

print ("Fahrenite 95 converted Celsius is:",ce)

fe = easytemp.converte_f(33)
print ("Celsius 33 converted Fahrenite is:",fe)
```

ဒီ Program ကတော့ Temperature တွက်ဖို့ဖြစ်ပါတယ် ဒါကြောင့် easytemp ကို import လုပ်ပါတယ် ဒီ program ကိုတည်ဆောက်ဖို့ File ယူတဲ့အခါမှာ easytemp.py ရှိတဲ့ Folder ထဲမှာဘဲယူဖို့ သတိပြုရပါမယ်။ ဒါမှ import ခေါ်သုံးနိုင်မှာပါ။



ကျွန်တော်က Folder တစ်ခုထဲမှာ တည်ဆောက်ထားတာကိုတွေ့ရပါလိမ့်မယ်။ဒါက ကျွန်တော် ဒီ module လေးကိုခေါ်သုံးမှာဖြစ်လို့ပါ။စစခြင်း import လုပ်တဲ့အခါမှာ module မတွေ့တာမျိုးဖြစ်နိုင်ပါတယ် ဒါဆိုရင် File ထဲကနေ ပြန် save ပြီး ထပ် Run ပေးပါ ဒါဆိုရင် အဆင်ပြေပါလိမ့်မယ် အခု ဆိုရင် ce ဆိုတဲ့ variable တစ်ခုကို တည်ဆောက်ပြီး easytemp.converte_c(95) ဆိုပြီး Fahrenheit 95 ကနေ Celsius ပြောင်းဖို့အတွက် ခေါ်သုံးထားပါတယ် နောက်ပြီး ce ကို print ထုတ်ထားတယ် ဒီတော့ 95 ရဲ့ Celsius တန်ဖိုးကို ရမယ် နောက်တစ်ခု fe ကတော့ Celsius ရဲ့တန်ဖိုးကနေ Fahrenheit ပြောင်းဖို့အသုံးပြုထားတာကိုတွေ့ရမှာပါ။ဒီ module တစ်ခုကို သုံးပြီးထပ်ခါထပ်ခါ program တွေမှာ C to F , F to C ပြောင်းချင်းကို လုပ်နေလို့ရတယ်။ပုံသေနည်းတွေပြန်ရေးနေစရာမလိုဘဲပေါ့။

ကျွန်တော်တို့ Module တစ်ခုထဲကို မဟုတ်ဘဲ namespace ပေးထားတဲ့ နေရာတစ်ခုလုံးမှာရှိတဲ့ modules အားလုံးကိုယူသုံးမယ်ဆိုရင် အများကြီးရေးစရာမလိုဘဲခေါ်သုံးနိုးရှိပါတယ်။

```
from tkinter import *
```

ဒီလိုရေးတာဟာ tkinter ထဲမှာရှိတဲ့ Modules အားလုံးကို import လုပ်လိုက်တာဘဲ ဖြစ်ပါတယ်။tkinter ထဲမှာ module ပေါင်းများစွာပါဝင်ပါတယ်။ဒီ Module ဘာ အလုပ်လုပ်လဲဆိုတာကတော့ နောက်ပိုင်းသင်ခန်းစာ ဖြစ်တဲ့ GUI မှာပါဝင်မှာပါ အခုတော့ အားလုံးကို import လုပ်နည်းပြတာဘဲဖြစ်ပါတယ်။နောက်တစ်ခု သိရမှာတော့ dir() function ပါ ဒီ function ကတော့ import လုပ်ထားတဲ့ namespace ထဲမှာ modules ဘယ်နခု ပါလဲဆိုတာကိုဖော်ပြပေးပါတယ်။ကြည့်ရအောင်

```
import tkinter

ce = dir(tkinter)
print(ce)
```

Run ကြည့်လိုက်တဲ့အခါမှာ tkinter ထဲက modules တွေအကုန်လုံးကို sorted List လုပ်ပေးထားတာ တွေ့ရမှာဖြစ်ပါတယ်။

Python Exceptions Handling

Python မှာ ပါလာတဲ့ လုပ်ဆောင်ချက် နှစ်ခုက အတော်လေးကောင်းပါတယ် အဲဒါကတော့ exception handling နဲ့ assertions ဖြစ်ပါတယ်။

Assertions in Python

Assertions ဆိုတာ sanity-check တစ်ခုပါဘဲ။ဒါကို program ထဲထည့်ချင်လဲရတယ် ပြီးတော့ ဖွင့်လို့ပိတ်လို့ရအောင်လဲလုပ်ထားနိုင်တယ်။ဒါက အခြေနေကို စစ်ဆေးတဲ့ လုပ်ဆောင်ချက်တစ်ခုပါဘဲ။အဲဒီအခြေနေမှန်နေသရွေ့ပြဿနာမရှိပေမယ့်အခြေနေမှားကိုရောက်သွားတဲ့အခါမှာexpression လုပ်ဆောင်ချက်နဲ့ Error ထွက်လာမှာဖြစ်ပါတယ်။

```
v = int(input("Enter Number:"))
assert v > 0,print("V is 0 Bad Bad")
if v >= 1:
    print("hello",v)
```

ဒီ Program မှာ v ကို user input ကို integer အနေနဲ့ယူထားပါတယ်။ပြီးတော့ assert နဲ့ Assertions Check လုပ်ထားတယ် v >0 နောက်က print() ကတော့ expression ဖြစ်ပါတယ်။user က သွင်းလိုက်တဲ့ ကိန်းတွေက 0 ထက်ကြီးတာတွေဖြစ်နေသရွေ့ဘာပြဿနာမှ မဖြစ်ပါဘူး။ဒါပေမယ့် 0 ဖြစ်တဲ့အခါမှာတော့ Error ကိုတွေ့ရမှာဖြစ်သလို v is 0 Bad Bad ဆိုတဲ့စာသားပေါ်လာတာလည်းတွေ့ရမှာပါ။

```
Enter Number:0
Traceback (most recent call last):
  File "C:/Users/Aung Moe Kyaw/Documents/Python Program/Lesson1/lesson.py", line 2, in <module>
    assert v > 0,print("V is 0 Bad Bad")
AssertionError: None
V is 0 Bad Bad
```

ဒါဆိုရင် နောက်ထပ် တစ်ခုဖြစ်တဲ့ Exception Handling ကိုဆက်သွားရအောင်။Exception Error Handling ဆိုတာကတော့ Error တွေတတ်နိုင်တဲ့ကနေရာတွေကို Error တတ်ပြီး program ကြီးရပ်တန့်မသွားအောင် အသုံးပြုတာဖြစ်ပါတယ်။ဘယ်လိုလဲဆိုတော့ လက်တွေ့တည်ဆောက်ကြည့်ရအောင်။

```
v = int(input("Enter The Number:"))
total = v +100
print(total)
```

ဒီ program မှာ int ကို userinput လက်ခံထားပြီး total ထဲမှာ v +100 ဆိုပြီး ကျွန်တော်တို့ ထည့် လိုက်တဲ့ကိန်းကို 100 နဲ့ပေါင်း အဖြေထုတ်မှာဖြစ်ပါတယ်။ကျွန်တော်တို့က user input မှာ ကိန်းဂဏန်းတွေ ရိုက်ထည့်

နေရင် အဆင်ပြေနေပေမယ့် စာသားတွေနဲ့ တခြားသော ပေါက်ကရတွေရိုက်ထည့်ရင် Error တတ်ပြီး Program လည်း run တာ အခက်ခဲဖြစ်ပြီးရပ်တန့်သွားမှာဖြစ်ပါတယ်။လက်တွေ့စမ်းကြည့်ရအောင်။

```
Enter The Number:hey

Traceback (most recent call last):

  File "C:/Users/Aung Moe Kyaw/Documents/Python Program/Lesson1/lesson.py", line 1, in <module>

    v = int(input("Enter The Number:"))

ValueError: invalid literal for int() with base 10: 'hey'
```

ဒါက input ကို hey လို့ရိုက်ထည့်လိုက်လို့ ဖြစ်ပေါ်လာတဲ့ error ပါ ValueError လို့ရေးထားပါတယ် ဒါမျိုးကတကယ့် ပြဿနာပါဘဲ ကျွန်တော် တို့ program ကို သုံးမယ့် user က အမှားမလုပ်ဘူးလို့ပြောမရပါဘူး။သူအမှားလုပ်တဲ့အခါ program ကြီးရပ်တန့်သွားရင် တကယ့်ပြဿနာကြီးပါဘဲ။ဒီလိုမဖြစ်အောင် Exception လုပ်ဖို့လိုပါတယ်။ဒါဆိုလုပ်ကြည့်ရအောင်။

```
try:
    v = int(input("Enter The Number:"))

except:
    print("Please Input Only Number!!!!")

else:
    total = 1000 + v
    print("Answer:",total)
```

try: ထဲမှာ မှားနိုင်လောက်တဲ့ မှားတတ်တဲ့ လုပ်ငန်းကိုထည့်ရပါတယ် ပြီးရင် except: ထဲမှာတော့ မှားရင် လုပ်မယ့် အလုပ်တွေကိုထည့်ပါတယ်။နောက်တစ်ခု else ထဲမှာကတော့ မမှားခဲ့ရင် except: ကိုကျော်ပြီး ပုံမှန်အဖြေထုတ်ချင်တဲ့လုပ်ငန်းတွေကိုထည့်ရပါတယ်။ဒီတော့ပြန် Run ကြည့်ရအောင်။

```
Enter The Number:hey

Please Input Only Number!!!!
```

Exception လုပ်ပြီးတဲ့အခါမှာတော့ hey လို့ရိုက်ထည့်တဲ့အခါမှာ ဘာပြဿနာမှမတတ်ဘဲ Error နိနီကြီးတွေ မတတ်ဘဲနဲ့ကျွန်တော်တို့မှားရင်လုပ်ချင်တဲ့အလုပ်ကိုသာလုပ်သွားတာတွေ့ရမှာပါ။print ထုတ် ထားတဲ့ သတိ ပေးစာတမ်းကိုတွေ့ရမှာပါ။ဒါက ဘာ Error ရယ်လို့အတိကျ မဖမ်းထားဘဲနဲ့ exception လုပ်ထားတာ ပေါ့ Error အတိအကျဖမ်းဖို့လိုတာတွေအတွက် အတိကျဖမ်းနည်းပြောပါမယ်။

```
num1 = 100
num2 = 0

ans = num1/num2
```

ကျွန်တော်တို့ calculator program ရေးတဲ့အခါမှာ ဒီလိုအမှားမျိုးဖြစ်တတ်ပါတယ် ဘယ်ဂဏန်းကိုမှ 0 နဲ့ စားလို့မရပါဘူး Error ဖြစ်ပါလိမ့်မယ်။

windows calculater မှာတောင် 0 နဲ့စားမိရင် သတိပေးစာပေါ်ပါတယ် မဟုတ်ရင် Program က ရပ်တန့်သွားမှာဖြစ်ပါတယ် Exception လိုအရာမျိုးတွေလုပ်ထားတဲ့အတွက် သာ ရပ်တန့်မသွားတာဖြစ်ပါတယ်။ဘာ error လဲ ကြည့်ရအောင်။

```
Traceback (most recent call last):
  File "C:/Users/Aung Moe Kyaw/Documents/Python Program/exception/exceptionZeroDivision.py", line 4, in <module>
    ans = num1/num2
ZeroDivisionError: division by zero
```

ZeroDivisionError လို့ပြနေပါတယ်။အဲဒီ Error ကိုဖမ်းကြတာပေါ့။

```
try:
    num1 = int(input("Enter Number:"))
    num2 = int(input("Enter Number:"))
    ans = num1 / num2
except ZeroDivisionError:
    print("Cannot divided by zero")
else:

    print(ans)
```

ဒီ program မှာ Error နေရာမှာ except အနေနဲ့ ZeroDivisionError ဆိုပြီးထည့်ထားပါတယ် ဒီတော့ စားကိန်းက 0 ဖြစ်နေရင် Cannot Division Zero ဆိုတာပေါ်မယ်။မဟုတ်ရင် အလုပ်ဆက်လုပ်မယ် ဒါပေမယ့် ပြဿနာတစ်ခုကရှိနေတယ် ZeroDivsionError ကိုဘဲ ဖမ်းတာဖြစ်တဲ့အတွက် တခြား Error တွေမဖမ်းဘူး ဥပမာ input မှာ int တွေမဟုတ်ဘဲ စာသားတွေထည့်မိရင်တောင် အနီရောင် Error ကြီးပြပြီး ရပ်တန့်သွားမှာဖြစ်ပြီး error ဖမ်းမှာမဟုတ်ပါဘူး။ဒီပြဿနာကိုဖြေရှင်းဖို့အတွက်က except နေရာမှာ Error တွေအကုန်ဖမ်းပို့မယ်။

```
try:
    num1 = int(input("Enter Number:"))
    num2 = int(input("Enter Number:"))
    ans = num1 / num2
except (ZeroDivisionError,ValueError):
```

```
    print("""
    something wrong!!!
    Please Check Your Input Only Number:
    Please Check Your Division Number is Zero:
         """)


else:

    print(ans)
```

ဒီမှာ Error ၂ မျိုးလုံးဖမ်းထားပါတယ်။ZeroDivisionError နဲ့ ValueError ဖြစ်ပါတယ် ဒီတော့ ပြဿနာတစ်စုံတစ်ရာဖြစ်လာရင် Message ကြလာမယ် Please ဘာညာနဲ့ပေါ့ ဒီမှာထုတ်ထားတဲ့ Message တွေက Grammer မှန်ချင်မှမှန်လိမ့်မယ် ကျွန်တော် ဂရုမစိုက်ဘူးကျွန်တော်က Python သင်နေတာ အင်္ဂလိပ်သင်တာ မဟုတ်ဘူး OK။Exception ထဲမှာ Exception ထပ်ထည့်ပြီး Error ဖမ်းတာလုပ်နိုင်ပါတယ်။ဒီ Program ကိုဘဲ ကျွန်တော်တို့ Input Error ဆို Input Error ပြမယ် ZeroDivisionError ဆို ZeroDivisionError ပြမယ်ပေါ့။

```
try:
    num1 = int(input("Enter Number:"))
    num2 = int(input("Enter Number:"))
except:
    print("Input Error")

else:
    try:
        ans = num1/num2

    except ZeroDivisionError:
        print("Cannot Divided by Zero")

    else:
        print(ans)
```

ဒီမှာ try: နဲ့အရင်ဆုံး Input Error ဖမ်းထားတယ် Input Error မရှိဘူးဆိုရင် နောက်ထပ် try: နဲ့ except: ထပ်လုပ်ပြီး ZeroDivisionError ဖမ်းတယ် ပြီးမှာ print နဲ့ အဖြေထုတ်ပါတယ်။နောက်တစ်ခု try: finally: အကြောင်းလေးပြောရအောင်။finally: keyword ကတော့ try: နဲ့တွဲသုံးနိုင်ပြီးတော့ Error ရှိသည်ဖြစ်စေ Error မရှိသည်ဖြစ်စေ သူ့ထဲကအလုပ်ကိုလုပ်စေပါတယ်။အသုံးပြုပုံကတော့ဒီလိုပါ။

```
try:
    num1 = int(input("Enter Number:"))
    num2 = int(input("Enter Number:"))
except:
    print("Input Error")

else:
    try:
      try:
          ans = num1/num2
      finally:
          print("Hello I am Finally Keyword")

    except ZeroDivisionError:
        print("Cannot Divided by Zero")

    else:
        print(ans)
```

ဒီမှာ Run လိုက်တဲ့အခါမှာ Finally ထဲက စာသားကို အလုပ်လုပ်စေပါတယ်။အားလုံးအဆင်ပြေတာကို တွေ့ရ မှာဖြစ်ပါတယ်။

File Read/Write with Python

File Read or Write ဆိုတာ system အတွက် app တွေရေးတဲ့အခါမှာ အရေးပါပါတယ်။Text ဖိုင်တွေထဲက စာတွေကိုဖတ်မယ်၊ဒါမှမဟုတ် Text ဖိုင်တွေထဲကိုစာတွေရေးမယ်ဆိုရင် Read/Write အသုံးပြု နိုင်ပါတယ်။အရင်ဆုံး File Read မယ် Write မယ်ဆိုရင် ဖိုင်ကို Open လုပ်ရမှာဖြစ်ပါတယ်။open() ဆိုတဲ့ Function ကိုအသုံးပြုနိုင်ပါတယ်။open() function မှာ open(file , access mode,buffering) ဆိုပြီး သုံး မျိုး နဲ့အသုံးပြုနိုင်ပါတယ်။Buffering မပါရင်ရပေမယ့် access mode ကတော့မပါမဖြစ်ပါရမှာဖြစ်ပါတယ်။

File နေရာမှာတော့ Text File တွေ နဲ့တခြား .dat လို စာရေးလို့ဖတ်လို့ရတဲ့ဖိုင်တွေကို အသုံးပြု နိုင်တယ် access mode ကတော့ File Open တဲ့နေရာမှာ Read လုပ်မှာလား Write လုပ်မှာလား ဆိုတာကို ရေးပေးရတာပါ။Line Buffering လုပ်ချင်တဲ့အခါမှာ buffering တန်ဖိုးထည့်ပေးရပါတယ်။ဒါဆိုရင် ရေးကြည့်ရအောင်။

```
file_open = open("example.txt","w")
```

ဒါကတော့ file_open ဆိုပြီး File Object တစ်ခုတည်ဆောက်လိုက်ပြီးရင် open() function နဲ့ "example.txt" ဆိုတဲ့ဖိုင်ကိုဖွင့်လိုက်ပါတယ်နောက်က access ကတော့ "w" ဆိုတော့ write ဖြစ်ပါတယ်။ဒါကို run လိုက်မယ် ဆိုရင် Python က သူ့အလိုလို example.txt ကိုတည်ဆောက်သွားမှာပါ ဘာကြောင့်လဲဆိုတော့ w ဖြစ်လို့ပါ write နေရာမှာ read " r " သာဖြစ်မယ်ဆိုရင်တော့ example.txt ဆိုတဲ့ ဖိုင်မရှိရင် သေချာပေါက် error တတ်မှာ ဖြစ်ပါတယ်။w မှာတော့မရှိရင်တည်ဆောက်သွားမယ် error မတတ်ပါဘူး။ဒါဆိုရင် စာရေးပို့မယ်။

```
file_open = open("example.txt","w")
file_open.write("Python Tutorial")
file_open.close()
```

ဒါဆိုရင် ကျွန်တော်က စာရေးလိုက်တာဖြစ်ပါတယ် Python Tutorial ဆိုတဲ့စာကို example.txt ထဲကို ထည့်ရေးလိုက်တာပါ ထည့်ရေးလိုက်တာနဲ့မပြီးသေးပါဘူး save တဲ့ပုံစံမျိုးနဲ့ file_open.close() ဆိုတာ ကို ထည့်ရေးရပါတယ်။example.txt ကိုဖွင့်ကြည့်လိုက်ပါ ဒါဆိုရင် ကျွန်တော်တို့ရေးထားတဲ့စာကိုတွေ့ရမှာပါ။ဒါ ဆိုရင် File Read တာကြည့်ရအောင်။

```
file_read = open("example.txt","r")
text = file_read.read()
print(text)
file_read.close()
```

ဒီဟာကတော့ Python မှာဖိုင်ဖတ်ဖို့အတွက် အသုံးပြုတာဘဲဖြစ်ပါတယ်။ဒါဆိုရင် example.txt ထဲမှာ ရေး ထားတဲ့ စာတွေကိုဖတ်နိုင်ပါတယ်။

စာလုံးအရေတွက်ကို ဖော်ပြချင်သလောက်ဘဲ ဖော်ပြလို့လဲရပါတယ် read() ထဲမှာ စာလုံး အရေတွက် ထည့်ပြီး အသုံးပြုနိုင်ပါတယ်။

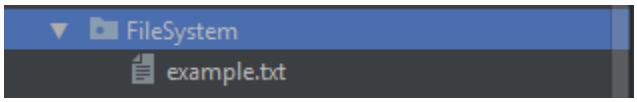
```
file_read = open("example.txt","r")
text = file_read.read(5)
print(text)
file_read.close()
```

ဒါဆိုရင် Pytho ဆိုတဲ့စာလုံး ၅ လုံးကိုဘဲဖော်ပြမှာပါ။

Rename and Remove File

Files တွေကို နာမည်ပြောင်းဖို့ ဖျက်ပစ်ဖို့အတွက် Module တစ်ခုကို import လက်ခံရယူပြီး ပြုလုပ်နိုင်ပါတယ်။အဲဒီ module နာမည်ကတော့ os ဖြစ်ပါတယ်။rename() နဲ့ remove() method နှစ်ခုက os ထဲက ဖြစ်ပါတယ် အဲဒီ os module ကို import ရယူပြီးမှသာ rename() နဲ့ remove() ကိုအသုံးပြုနိုင်မှာဖြစ်ပါတယ်။

ပထမဆုံး ကျွန်တော်တို့ pyCharm IDE ထဲမှာ New Folder ယူပြီး အဲဒီထဲမှာ example.txt ဆိုတဲ့ Text ဖိုင်တစ်ဖိုင်တည်ဆောက်ထားပါ့မယ်။



အဲဒီထဲမှာ ဘဲ Python File တစ်ဖိုင်ယူလိုက်ပါမယ်။စိတ်ကြိုက်နာမည်ပေးလို့ရပါတယ်။ကျွန်တော်ကတော့ ostext.py ပေးထားပါတယ်။

```
import os

os.rename("example.txt","namelist.txt")
```

ဒီမှာ အရင်ဆုံး os ကို import လုပ်တယ်ပြီးတော့ os.rename() နဲ့ example.txt ဆိုတာ နဂိုရှိတဲ့ဖိုင်ပြီးတော့ ပြောင်းချင်တဲ့ဖိုင်နာမည်ကိုထည့်ပေးရပါတယ်။namelist.txt ဆိုတာ ကျွန်တော်ကပြောင်းချင်နဲ့ နာမည် ဖြစ်ပါတယ် Run လိုက်ပါ။Run လိုက်တဲ့အခါမှာ namelist.txt ဆိုတဲ့နာမည်ကိုပြောင်းသွားမှာ တွေ့ရမှာ ဖြစ်ပါတယ်။

နောက်တစ်ခုကတော့ os.remove() ပါ ဖိုင်တွေကိုဖျက်ပစ်ချင်တဲ့အခါမှာ သုံးရပါတယ်။လက်တွေ့သုံးကြည့်ရ အောင်ဗျာ။

```
import os

os.remove("namelist.txt")
```

စောစောကတည်ဆောက်ထားတဲ့ namelist.txt ကိုဘဲပေးထားတဲ့အတွက် အဲဒီဖိုင် delete ဖြစ် သွား တာကို တွေ့ရပါလိမ့်မယ်။os ထဲမှာက method တွေအများကြီးရှိပါတယ်ဒါပေမယ့် အခြေခံသင်ခန်းစာမှာကတော့ ဒီနှစ်ခုကိုဘဲထည့်ရေးပေးလိုက်ပါတယ်နောက်ထပ် Advance သင်ခန်းစာတွေအနေနဲ့ Python OOP, Python Database, Python GUI တို့ထပ်ပါဖင်လာမှာဖြစ်ပါတယ်။

Python Object-Oriented Programming

Python Programming မှာ Object Oriented Programming OOP ကို အထောက်ပံ့ပေးတဲ့ Language တစ်ခုဖြစ်ပါတယ်။OOP ဆိုတာ Programming ရေးထုံး နည်းပညာတစ်ခုဘဲဖြစ်ပါတယ်။OOP မပါဘဲနဲ့ Program တွေရေးနိုင်တယ်ဆိုပေမယ့် တကယ့် Project အကြီးကြီးတွေမှာ OOP မပါဘဲရေးတဲ့ အခါမှာ Code တွေထပ်ခါထပ်ခါရေးရတဲ့အတွက် မလိုအပ်ဘဲများတာမျိုးဖြစ်လာတာရယ် အမှားတွေ့တဲ့အခါ ရှုပ်ထွေးတဲ့ Code ရေးစနစ်တွေကြောင့်ပြင်ဆင်ဖို့ခက်ခဲတာမျိုးဖြစ်လာပါတယ်။ဒါတွေကို ဖြေရှင်းဖို့နည်းလမ်းကို ကြံစည်ကြတဲ့အခါမှာ OOP စနစ်ကိုတည်ထွင်ခဲ့ကြပြီး 1967 မှာ Simula 67 လို့ခေါ်တဲ့ Programming က ပထမ ဆုံး Object ကိုစသုံးတဲ့ Language ဖြစ်လာပါတယ်။1970 လောက်မှာတော့ OOP ကိုစတင်အသုံးပြုခဲ့ကြတယ်လို့ ဆိုပါတယ်။ယခုအခါမှာ Computer Scientists တွေနဲ့ Programmer တွေက OOP ကို modern Programming Paradigm အဖြစ်သတ်မှတ်ထားကြပါတယ်။OOP က တကယ့်ကို Powerful ဖြစ်တဲ့ နည်းပညာ တစ်ခုဖြစ်ပါတယ် အသုံးလည်းဝင်ပါတယ် ဒါပေမယ့် Python မှာ OOP မသုံးဘဲလည်း Powerfulဖြစ်တဲ့ ကောင်းမွန်တဲ့ Program တွေရေးနိုင်တယ်ဆိုတာကိုတော့မမေ့ပါနဲ့။

OOP ဆိုတဲ့ Object တွေ Class တွေနဲ့ Program ကို ထပ်ခါထပ်ခါရေးရခြင်းကို လျော့ချခြင်းဖြင့် သော်လည်းကောင်း၊နည်းပညာတစ်ခုဖြစ်တဲ့ Encapsulation ကိုအသုံးပြုခြင်းဖြင့် လုံခြုံသော Code များကိုအသုံးပြုနိုင်ခြင်း စတဲ့ အကျိုးကျေးဇူးများစွာကိုရစေပါတယ်။OOP အကြောင်းကို တခြားသော ဆရာများနဲ့ တခြားသော Programmer များက အဓိပ္ပါယ်ဖွင့်ဆိုမှုများစွာရှိပါတယ် Object အကြောင်းရော Class အကြောင်းရောပေါ့။ဒါပေမယ့် ကျွန်တော်ကတော့ ကျွန်တော်နားလည်ထားတဲ့ ပုံစံ နဲ့ရေးသွားမှာဖြစ်ပါတယ် ဒါကြောင့် အမှားယွင်းရှိရင် ထောက်ပြကြပါလို့မေတ္တာရပ်ခံရင်း အမှားယွင်းဖြစ်သွားရင်လဲ ကြိုတင်တောင်း ပန်ပရစေ။

OOP ကိုအကြောင်းကိုပြောမယ်ဆိုရင် အဓိက ကျတဲ့ ရေးနည်း နည်းပညာ ၄ ခုအကြောင်းကို မဖြစ်မနေ ပြောရမှာဖြစ်ပါတယ် OOP ရဲ့ အဓိက လုပ်ဆောင်ချက်ကလည်း ဒီ ၄ ခုဖြစ်ပါတယ်။ဒါတွေကတော့ Encapsulation, Data Abstraction, Polymorphism, Inheritance တို့ဖြစ်ပါတယ်ဒါပေမယ့် ဒါတွေ မပြောခင် Object နဲ့ Class တွေအကြောင်းအရင်ပြောရအောင်။Class ဆိုတာ Object တည်ဆောက်ဖို့အတွက် အုပ်မြစ်ချတဲ့အရာဘဲဖြစ်ပါတယ်။Class ကိုတည်ဆောက်ပြီးမှ Object ကိုတည်ဆောက်နိုင်ပါတယ်။Object ဆိုတာကတော့ ခံစားလို့ရတဲ့အရာအားလုံးကိုဆိုလို့ပါတယ် ဥပမာ variable တွေ လုပ်ဆောင်နိုင်တဲ့ လုပ်ဆောင် ချက်တွေက Object တွေပါဘဲ။ဒီလိုပြောတော့ ရေးတေးတေးဖြစ်နေမှာပါ ဒါကြောင့် လက်တွေ့ တည်ဆောက်ကြည့်ရအောင်။

```
class name_info:
    name = "AungMoeKyaw"
    age = "22"
```

```
obj_name = name_info()
p_name = obj_name.name
p_age  = obj_name.age
print(p_name)
print(p_age)
```

ဒီ program မှာကြည့်လိုက်ပါ class ဆိုတဲ့ keyword ကို အသုံးပြုပြီးတော့ name_info ဆိုတဲ့ class တစ်ခုကိုတည်ဆောက်လိုက်ပါတယ် ဒါက objectမဖြစ်သေးပါဘူး။ဒီထဲမှာ variable ၂ ခုပါတယ် name နဲ့ age ဖြစ်ပါတယ်။class ထဲက variable တွေကိုခေါ်လို့မရပါဘူး ဘာကြောင့်လဲဆိုတော့ သူက global variable လဲမဟုတ်သလို တိုက်ရိုက်ခေါ်နိုင်ဖို့မသက်ဆိုင်ပါဘူး ဒါဆိုရင် ဒီထဲက variable တွေကို ခေါ်ချင်ရင် object ပြောင်းပြီးမှခေါ်လို့ရပါတယ်။ဒီတော့အောက်မှာ obj_name ဆိုတဲ့နာမည်နဲ့ name_info() ဆိုတဲ့ class ကို object အဖြစ်ပြောင်းတည်ဆောက်လိုက်ပါတယ်။ဒီကနေပြီးတော့ obj_name.name ဆိုတဲ့ object တစ်ခု obj_name.age ဆိုတဲ့ object တစ်ခုရပြီး သုံးစွဲနိုင်ပါတယ်။ဒါက ရိုးရှင်းတဲ့ Object တည်ဆောက်ပုံဖြစ်ပါတယ်။ class ထဲမှာက variable တွေ method တွေ function တွေနဲ့ တခြားအလုပ်များစွာပါနိုင်ပါတယ်။ဒါဆို နောက်တစ်ခုတည်ဆောက်ကြည့်ရအောင်။

```
class car:
    car_hp = 0
    car_name =""

    def __init__(self):
        print("*******Car Infomation********")
        self.car_hp =0
        self.car_name = "Unknow"
        self.show_info()

    def set_name(self,name):
        self.car_name = name

    def set_hp(self,hp):
        self.car_hp = hp

    def show_info(self):
        print("Car Name :" + str(self.car_name))
        print("Car Hp :" + str(self.car_hp))

first_obj = car()

print("_____________________________________")

n_name = input("Enter Car Name:")
first_obj.car_name = n_name

hp_unit = input("Enter Car Hp:")
first_obj.car_hp = hp_unit

first_obj.show_info()

print("_____________________________________")

sec_obj = car()
print("_____________________________________")
sec_obj.car_name ="BMW"
sec_obj.car_hp = 2500
sec_obj.show_info()
```

class တစ်ခုတည်ဆောက်ထားပါတယ် အဲဒီ class နာမည်က car ဆိုတာဖြစ်ပါတယ် အဲဒီထဲမှာ variable ၂ ခု တည်ဆောက်ထားပါတယ် car_hp နဲ့ car_name ဖြစ်ပါတယ်။နောက်ထပ် method တွေလဲ ပါဝင်ပါတယ်။ပထမဆုံးတွေ့ရမယ့် __init__(self) က constructor ဖြစ်ပါတယ်။constructor ဆိုတာ အဲဒီ class ကို object ပြောင်းလိုက်တာနဲ့ခေါ်စရာမလိုဘဲ constructor ထဲမှာရှိတဲ့အလုပ်တွေကလုပ်ပါတယ်။self ဆိုတာကတော့ class တည်ဆောက်ပြီး method ထည့်ရင် သုံးရမယ့် argument ဖြစ်ပါတယ်။နောက်ထပ် method ထဲမှာသုံးမယ့် variable တွေ လုပ်ဆောင်ချက်တွေကိုလည်း self နဲ့ဘဲခေါ်သုံးရပါမယ်။အခု constructor ထဲမှာ ဘာအလုပ်တွေလုပ်ထားလဲကြည့်ရအောင် print() နဲ့စာတစ်ကြောင်းထုတ်ထားတယ် ပြီးတော့ self.car_hp =0 နဲ့ self.car_name = "Unknow" ဆိုပြီး ပေးထားပါတယ် ပြောရရင် construtor ကြည့်ဘဲခေါ်ထားရင် car_hp က 0 car_name က unknow ဖြစ်ပါတယ်။ပြောရရင် ပထမဆုံး class ကို object ဆောက်လိုက်တဲ့အခါမှာ တန်ဖိုးတွေက 0 နဲ့ unknow ထွက်မှာပါ။self ကတော့ class တွေထဲက ဘယ် အရာမဆိုခေါ်မယ်ဆိုရင် self ထည့်ရပါတယ်။self.show_info() ဆိုပြီး show_info() method ကိုလဲလှမ်းခေါ် ထားပါတယ်။show_info() က မတည်ဆောက်ရသေးဘူးနော်။နောက်ထပ် ကျွန်တော်က method တစ်ခုတည် ဆောက်ထားပါတယ် set_name(self,name) ဖြစ်ပါတယ်။self argument အပြင်နောက်ထပ် argument တစ်ခုပါ ပါလာပါတယ်။ဒါကတော့ pass လုပ်ပြီး တန်ဖိုးထည့်ဖို့ဖြစ်ပါတယ်။အဲဒီမှာက self.name ထဲကို name ထည့်ထားပါတယ်။ပြီးတော့ set_hp() ထဲမှာလဲအဲလိုပါဘဲ show_info() ကတော့ print နဲ့ car_name နဲ့ car_hp ကိုအဖြေထုတ်ထားပါတယ် တစ်ခုသိထားရမှာက str() နဲ့ string ပြန်ပြောင်းရပါတယ် ဒီလိုမှမဟုတ်ရင် အဖြေထွက်မှာမဟုတ်ပါဘူး။ပြီးရင် class ကို object ပြောင်းတယ် first_obj ဆိုပြီးတော့ ပြောင်းတာပါ။အဲဒီလိုတည်ဆောက်လိုက်တာနဲ့ construtor ကအလုပ်လုပ်တာဖြစ်လို့ constructor ထဲမှာရှိတဲ့ အလုပ်တွေအကုန်လုပ်ပါလိမ့်မယ်။ပြီးရင် ကျွန်တော်က input() နဲ့တန်ဖိုးနှစ်ခု ယူတယ်ပြီးတော့ first_obj.car_name နဲ့ first_obj.car_hp ထဲကိုထည့်ပါတယ်ပြီးတော့ အဖြေထုတ်ဖို့ first_obj.show_info() ကိုခေါ်ပြီးအဖြေထုတ်ပါတယ်။နောက်ထပ် sec_obj ဆိုပြီး sec_obj တည်ဆောက်ပြီး တန်ဖိုးကို အသေပေး အဖြေထုတ်တဲ့အခါမှာလဲ အဖြေထွက်ပါတယ်။ဒါကိုသိစေချင်တာက class တစ်ခုမှာ object တွေအများကြီးတည်ဆောက်နိုင်ပြီးတော့ သူ့ object နဲ့သူ သီးသန့်အလုပ်တွေကိုလုပ်စေတယ်ဆိုတာ ကိုနားလည်ထားဖို့ဖြစ်ပါတယ်။ဒါဆိုရင် ဒါက class တွေ object တွေအကြောင်းဘဲဖြစ်ပါတယ် ဒါဆိုရင် Encapsulation ကိုထပ်သွားရအောင်။

Encapsulation

Object Oriented Programming မှာ method နဲ့ variable တွေကို အကန့်သတ် လုပ်ပြီး ကန့်သတ်ချက်ထားနိုင်ပါတယ်။ဒါက ဘာအတွက်လဲဆိုရင် data တွေရဲ့တန်ဖိုးတွေကို မတော်တစ ပြောင်းလဲ မိတာမျိုးကိုကာကွယ်ပေးပါတယ်။ဒါအပြင် method နဲ့ variable တွေကို တိုက်ရိုက် လက်ခံခြင်းပြုလုပ်တာ ကို

ကန့်သတ်ထားတဲ့အတွက် လုံခြုံမှုလည်းပိုရှိသွားပါတယ်။ဒါကို Encapsulation လို့ခေါ်ပါတယ် ဆေးတွေမှာ capsul လုပ်သလိုမျိုးအခွံတစ်ထပ်အုပ်ထားတဲ့သဘောနဲ့ program ကိုလဲ ပြုလုပ်ထားလို့ပါ။အရင်ဆုံး ကျွန်တော်တို့ Ecapsulation လုပ်မယ်ဆိုရင် access properties တွေကိုသိထားဖို့လိုပါတယ် ဒါတွေက ဘာတွေလဲဆိုတော့ public , protected , private တို့ဘဲဖြစ်ပါတယ်။

Public	Example : a = 100 def a(self)	Public သည် ဘယ်နေရာမှ မဆို Access လုပ်နိုင်သည်။
Protected	Example :_a = 100 def_a(self)	Protected သည် Public နှင့်တူသည် သို့သော် သူ့၏ class ရှိသော နေရာတွင်သာ Access လုပ်နိုင်သည်။
Private	Example :__a =100 def __a(self)	Private သည် သူ့၏ class အတွင်းမှ လွဲ၍အခြားနေရာမှ Access မလုပ်နိုင်ပါ။

ရေးနည်းရေးဟန်ကတော့ အထက်ပါအတိုင်းဖြစ်ပါတယ်။ပထမ Program မှာပြထားတဲ့အတိုင်းပါဘဲ။Public variable , Public method တွေရေးဖို့အတွက်က သာမန် variable သာမန် method တွေလိုပါဘဲ။ဒါပေမယ့် protected ကိုတော့ _ ခံရေးပါတယ် _ တစ်ချက်ခံရေးတာ protected ဖြစ်ပါတယ်။နောက်ထပ် Private ကတော့ __ နှစ်ချက်ခံရေးရပါတယ်။ဒါဆိုလက်တွေ့ကြည့်ရအောင်။

```
class letter(object):
    def __init__(self, a, b, c):
        self.a = a
        self._b = b
        self.__c = c


new_obj = letter("I am Public", "I am Protected", "I am Private")
print(new_obj.a)
print(new_obj._b)
print(new_obj.__c)
```

ဒီမှာ class တစ်ခုတည်ဆောက်ထားပြီး constructor တစ်ခုပါဂင်တယ် အဲဒီ constructor မှာ self argument အပြင် a , b ,c ဆိုတဲ့ argument တွေပါဂင်တယ်။အဲထဲမှာ variables သုံးခုရှိပါတယ် အဲဒီ သုံးခုမှာ a က public အနေနဲ့ လည်းကောင်း ၊ b က protected အနေနဲ့လည်းကောင်း c က private အနေနဲ့လည်းကောင်း သုံးခုတည်ဆောက်ထားပါတယ် ပြီးတော့ ကျွန်တော်တို့ object အနေနဲ့တည်ဆောက်ပြီး value passing လုပ်ပါတယ် အဲဒီမှာ ပြန်ခေါ်သုံးတဲ့အခါမှာ public နဲ့ protected ကိုခေါ်အသုံးပြုနိုင်ပြီး private ကတော့ error တတ်ပါတယ်။ဒါက private public protected ပုံစံကိုသိစေချင်လို့ဖြစ်ပါတယ် Encapsulation ရေးပုံ မဟုတ် သေးပါဘူး။

```
class letter(object):
    def __init__(self, a, b, c):
        self.a = a
        self._b = b
        self.__c = c
```

```
    def call_value(self):
        var = self.__c
        return var


new_obj = letter("I am Public", "I am Protected", "I am Private")
print(new_obj.a)
print(new_obj._b)
print(new_obj.call_value())
```

ဒီ program က စောစောက private ကိုယူဖို့အတွက် တည်ဆောက်ထားတဲ့ program ဖြစ်ပါတယ်။private က သူ့ class ထဲက သူဆိုရင်တော့ခေါ်ပြီးအသုံးပြုနိုင်တယ်လို့သိရပါတယ် ဒါကြောင့် method တစ်ခုကို class အတွင်းမှာဘဲတည်ဆောက်လိုက်ပါတယ်။ပြီးတော့ အဲဒီထဲမှာ var ဆိုတဲ့ variable တစ်လုံးကိုတည် ဆောက်တယ် ပြီးတော့ self.__c ကိုထည့်ပြီးတော့ var ကို return လုပ်လိုက်ပါတယ်။အဲဒီမှာဘဲ ဒီ တစ်ဆင့်ခံ method ကိုခေါ်ယူခြင်းနဲ့ ကျွန်တော်တို့က private variable ကို access လုပ်လို့ရစေပါတယ်။ဒါကို encapsulation ရေးထုံးလို့ခေါ်တာပါဘဲ။နောက်တစ်ခုလုပ်ကြည့်ရအောင်။

```
class letter:
    __num = 100
    __num1 = 200

    def set_value(self, a, b):
        self.__num = a
        self.__num1 = b

    def show(self):
        print(str(self.__num))
        print(str(self.__num1))


new_obj = letter()
new_obj.show()
new_obj.set_value(5000, 100000)
new_obj.show()
```

အခု Program မှာတော့ Encapsulation လုပ်ပြီးတန်ဖိုးထည့်တဲ့နည်းသုံးပါမယ်။private variable ၂ ခုတည် ဆောက်ထားပြီး တန်ဖိုးကိုအသေထည့်ထားပါတယ်။သာမာန် public နဲ့သာဆိုရင် တန်ဖိုးကို တိုက်ရိုက် ပြောင်းလဲနိုင်မှာပါ သို့သော ဒီမှာ private နဲ့ဖြစ်တဲ့အတွက် ပြောင်းလဲလို့မရပါဘူး။ဒီတော့ တန်ဖိုး ပြောင်းဖို့အတွက် method တစ်ခုတည်ဆောက်ရပါတယ်။ဒီမှာ constructor မပါဘဲတည်ဆောက် ထားတာတွေ့ရမှာပါ။private variable တွေကို တန်ဖိုးပြောင်းလဲဖို့ အတွက် set_value() ဆိုတဲ့ method တစ်ခု ကိုတည်ဆောက်ပြီးတော့ self argument အပြင် Argument နှစ်လုံးကိုလက်ခံထားတယ်။အဲဒီ passing ဖြစ်လာတဲ့ Argument တွေကို private variable တွေထဲကိုထည့်ပါမယ်။နောက် method တစ်ခုကတော့ show() ဆိုတာဖြစ်ပါတယ်။အဲဒီဟာကတော့ __num နဲ့ __num1 ကိုအဖြေထုတ်ဖို့ဖြစ်ပါတယ်။ဒါဆိုရင် object တည်ဆောက်တယ် ပြီးတော့ object ကိုခေါ်ပြီး set_value() နဲ့ private variable တွေကို တန်ဖိုးထည့်ပါတယ် ပြီးတော့ show method ကိုခေါ်ပြီး အဖြေထုတ်ကြည့်တဲ့အခါမှာ private တန်ဖိုးတွေပြောင်းနေမှာဖြစ်ပါတယ် ဒါပေမယ့် မူရင်းတန်ဖိုးကလဲ မပျက်ဘဲရှိပါတယ် ဘာကြောင့်လဲဆိုတော့ Encapsulation လုပ်ထားတဲ့ အတွက် တိုက်ရိုက်သွားမပြောင်းလဲလို့ဖြစ်ပါတယ်။

Inheritance

OOP ရဲ့နောက်ထပ် အဓိက လုပ်ဆောင်ချက်တစ်ခုကတော့ Inheritance ဖြစ်ပါတယ်။သူကတော့ class တစ်ခုကနေ တစ်ခုကို လုပ်ဆောင်ချက် ဆောင်ရွက်ချက်နဲ့ object method အားလုံးကို အမွေဆက်ခံနိုင်တဲ့ လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။အမွေဆက်ခံဖို့ မူရင်း class သို့မဟုတ် အမွေပေးတဲ့ class ကို မိဘ class parent class လို့ခေါ်သလို super class လို့လဲခေါ်ပါတယ်။အမွေဆက်ခံတဲ့ class သို့မဟုတ် အမွေယူ တဲ့ class ကိုတော့ sub class ခေါ် child class လို့လဲခေါ်ပါတယ်။

```
class super:
    name =""
    age  =""

    def set_user(self,a,b):
        self.name = a
        self.age  = b

    def show(self):
        print(str(self.name))
        print(str(self.age))


class sub(super):
      def pp(self):
          print("Python")

obj1 = sub()
obj1.set_user("AungMoeKyaw","22")
obj1.show()
```

ဒီမှာတော့ super ဆိုတဲ့ class တစ်ခုရယ် sub ဆိုတဲ့ class တစ်ခုရယ်တည်ဆောက်ထားပါတယ် super ထဲမှာ variable ၂ ခု method နှစ်ခုပါတယ်။sub class ထဲမှာကတော့ pp ဆိုတဲ့ method တစ်ခုဘဲပါပါတယ်။ဒါပေမယ့် sub(super) ဆိုပြီးရေးထားတာက Inheritance လုပ်ထားတာပါ။ပြောရရင် အမွေဆက်ခံထားပါတယ် အဲဒီတော့ sub class ထဲမှာ super ထဲမှာရှိသမျှ variable တွေ method တွေရောက်သွားပြီလို့ မြင်ရမှာ ဖြစ်ပါတယ်။ဒါကြောင့် sub() ကိုဘဲ object တည်ဆောက်ပြီး အသုံးပြုလိုက်တဲ့အခါမှာအသုံးပြုလို့ရမှာဖြစ်ပါတယ်။sub(super) ဆိုတဲ့ super ကိုဖြုတ်ကြည့်လိုက်ပါ error တွေတတ်လာမှာဖြစ်ပါတယ်။

Multiple Inheritance

Class တစ်ခုက မြောက်များစွာသော Class တွေစီကနေ အမွေဆက်ခံနိုင်ပါတယ် ဒီစွမ်းရည်က ကျွန်တော်တို့ ဆော့ဝဲလ်ရေးတဲ့အခါမှာ အများကြီးအထောက်ကူဖြစ်စေပါတယ်။

```
class sub(super_one,super_two,super_three):
```

ဒီလိုမျိုး super class အမြောက်အများကို ထည့်သွင်းအမွေဆက်ခံခြင်းကို Multiple Inheriatnce လို့ခေါ်ပါတယ်။

```
class super_one:
    def print_one(self):
        print("Hello I am Super_one")

class super_two:
     def print_two(self):
         print("Hello I am Super_Two")

class super_three:
    def print_three(self):
        print("Hello I am Super_Three")


class sub(super_one,super_two,super_three):
    def __init__(self):
        print("Hello I am Sub_Class")


objx = sub()
objx.print_one()
objx.print_two()
objx.print_three()
```

ဒီမှာ super class ၃ ခုတည်ဆောက်ပြီးအသုံးပြုထားပါတယ်။ပြီးတော့ ကျွန်တော်တို့ sub class တစ်ခုကနေ အမွေဆက်ခံထားပါတယ်။sub class ထဲမှာ constructor တစ်ခုတည်ဆောက်ထားပြီး တခြား class တွေမှာတော့ method တစ်ခုစီပါဂင်ပါတယ် ဒီ class တွေအားလုံးကို object ဆောက်ပြီးအသုံးပြုစရာ မလိုဘဲနဲ့ multiple inheritance လုပ်ပေးထားတဲ့ sub class တစ်ခုထဲကနေဘဲ အားလုံးကို အသုံးပြုစေပါတယ်။

Polymorphism of Python

Polymorphism ဆိုတဲ့ စကားလုံးရဲ့ အဓိပ္ပါယ်က ပုံစံအမြောက်အများဆိုတာဘဲဖြစ်ပါတယ်။တခါတရံ objects တွေက ပုံစံအမျိုးမျိုး ကွဲပြားတဲ့ Types အမျိုးမျိုးနဲ့ရှိနေတတ်ပါတယ်။ထားပါတော့မယ် ကျွန်တော်တို့ ရဲ့ software မှာ Button ၃ခုရှိတယ်ပေါ့။အဲဒီ Button သုံးခုလုပ်မယ့် အလုပ်ကအတူတူဘဲဗျာ onclick() ဆိုတဲ့ method ကိုဘဲလုပ်မှာ ဒါပေမယ့် အဖြေလေးတွေက နည်းနည်းဘဲကွဲပြားသွားမယ့် ဟာမျိုးဆို Polymorphism ကိုအသုံးပြုပါတယ်။ပြောရရင် တူညီတဲ့ method တစ်ခုထဲအပေါ်မှာ ကွဲပြားတဲ့အလုပ်တွေလုပ်နိုင်ခြင်းကို polymorphism လို့ခေါ်ပါတယ်။

```
class app:
    def show(self):
        print("Hello")

class app2:
    def show(self):
```

```
        print("Hello I am App2's Show")

ob = app()
ob.show()
obx = app2()
obx.show()
```

ဒီ program က အရှင်းဆုံး method overloading လုပ်သွားတဲ့ပုံစံဖြစ်ပါတယ်။နောက် Program တစ် စမ်းသက်ကြည့်ရအောင် class ၂ ခုဖန်တီးမယ် အဲဒီ class ၂ ခုမှာ တူညီတဲ့ method တစ်ခုတည်ဆောက်ပြီး မတည်ညီတဲ့လုပ်ဆောင်ချက် နှစ်ခုလုပ်ပါမယ်။

```
class bear:
    def sound(self):
        print("Groarrr")

class dog:
    def sound(self):
        print("Woof Woof")

def makeSound(animalType):
    animalType.sound()

bearobj = bear()
dogobj = dog()

makeSound(bearobj)
makeSound(dogobj)
```

ဒီမှာ bear နဲ့ dog ဆိုတဲ့ class နှစ်ခုတည်ဆောက်ထားပါတယ် ပြီးတော့ sound() ဆိုတဲ့ method ကတော့ class နှစ်ခုလုံးမှာ တူတူတည်ဆောက်ထားပါတယ် လုပ်ဆောင်ချက်ကတော့ မတူပါဘူး bear က Groarr လို့ ဖြစ်ပြီးတော့ dog ကတော့ Woof Woof ဖြစ်ပါတယ်။အဲဒါတွေကိုခေါ်သုံးကြည့်ရအောင် ဒီအတွက် method တစ် ခုတည်ဆောက်မယ် makeSound(animalType) ဆိုတဲ့ method ပေါ့ Argument ကတော့ animalType ကို pass လုပ်ထားပါတယ်။animalType.sound() ဆိုပြီး sound() method ကိုခေါ်ထားပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ bearobj နဲ့ bear() class ကို object ပြောင်းပါတယ်။dogobj ဆိုပြီး dog() class ကို object ပြောင်းပါတယ်။ပြီးတော့ makeSound() ထဲက Argument animalType ထဲကို object တွေ့ကို pass လုပ်ပေးပြီး ခေါ်လိုက်ပါတယ်။Same method ဆိုပေမယ့် object ပေါ်လိုက်ပြီးလုပ်ဆောင်ချက်ပြောင်းလဲ သွားတာတွေ့ရမှာပါ ဒါကလဲ Polymaphism ဖြစ်ပါတယ်။

```
class Document:
    def __init__(self,name):
        self.name = name


    def show(self):
        raise NotImplementedError("Subclass must implement abstract method")


class pdf(Document):
    def show(self):
        return "Show PDF FIle"

class word(Document):
    def show(self):
        return "Show Word File"
```

```
print(pdf("Document").show())
print(pdf("Document").name)
```

ဒီမှာ Document ဆိုတဲ့ class ကိုတည်ဆောက်ထားပါတယ် အဲဒီထဲမှာတော့ constructor ပါဝင်ပါတယ်။အဲ ဒီ constructor မှာ Argument တစ်လုံးထည့်ထားပါတယ် Document ဆိုတာကတော့ abstract class ပေါ့ဗျာ။ class နှစ်ခုထပ်တည်ဆောက်ပါတယ် pdf နဲ့ word ပါ နောက်ပြီး Document ကို Inheritance လုပ်ထားပါတယ် အဲဒီတော့ Document Class ထဲမှာပါတဲ့ method နဲ့ constructor ကလဲ pdf နဲ့ word class ထဲမှာ ရှိနေ တယ်လို့သိရမှာပါ။show() method ကတော့ class ၃ခုလုံးမှာတူညီမှုရှိပါတယ် ဒါပေမယ့် လုပ်ဆောင် ချက် တွေသာကွဲပြားပါတယ်။pdf မှာ return string အနေနဲ့ show pdf file ဆိုတာကို return လုပ်တယ် ပြီး တော့ word မှာတော့ return string က show word file ဆိုတာပြမယ်။pdf("document") ဆိုတာ ကိုခေါ်ပြီး name ကိုခေါ်ရင် ကျွန်တော်တို့ passing လုပ်ထားတဲ့ document ဆိုတဲ့နာမည်ကိုဖော်ပြမယ်။ပြီး တော့ pdf("document").show() ဆိုတာကိုခေါ်ရင်တော့ return string က show pdf file ဆိုတာကိုပြမှာဖြစ်ပါတယ်။raise ကတော့ exception တစ်ခုဘဲဖြစ်ပါတယ်။ကျွန်တော် တစ်ခုခြင်း တစ်ခုအလိုက် exception လုပ်ချင်တဲ့အခါမှာ raise ကိုသုံးလို့ရပါတယ်။show ကို ဒီမှာကတော့ တခြား class ကမဟုတ် ဘဲ Document ကိုခေါ်မယ်ဆိုရင် raise exception လုပ်မှာဖြစ်ပါတယ်။

```
class car:
    def __init__(self,name):
        self.name = name

    def drive(self):
        raise NotImplementedError ("Sub Class Must Implement abstract method")
    def stop(self):
        raise NotImplementedError ("Sub Class Must Implement abstract method")


class Truck(car):
    def drive(self):
        return "Drive Slowly"
    def stop(self):
        return "Truck Breaking"

class sportcar(car):
    def drive(self):
        return "Drive Fast"
    def stop(self):
        return "sportcar breaking"

obj_truck = Truck("Truck Car")
print(obj_truck.drive())
print(obj_truck.stop())

obj_spot = sportcar("Spot Car")
print(obj_spot.drive())
print(obj_spot.stop())
```

ဒီမှာလဲ car ဆိုတဲ့ abstract class ထဲမှာ name ထည့်ဖို့ paramenter ပါတယ် ပြီးတော့ constructor လဲ ပါတယ်။အဲဒီ class ကို Truck နဲ့ sportcar ဆိုတဲ့ class တွေက Inheriatance လုပ်တယ် ပြီးတော့ အဲဒီ class တွေထဲမှာ drive နဲ့ stop ဆိုတာ ပါတယ် ဒါပေမယ့် အခေါ်ခံရတဲ့ class ပေါ်မူတည်ပြီး တန်ဖိုးတွေကွာ ပါတယ်။မူလ abstract class ထဲက drive နဲ့ stop ကတော့ raise နဲ့ except လုပ်ထားပါတယ် အဲဒီ except

error ကတော့ NoImplementedError ဖြစ်ပါတယ် ဒီအမျိုးစား Error ကို except လုပ်မှာပါ။ဒါဆိုရင် ကျွန်တော်တို့ အပေါ်က တစ်ခုမှာလို object မဆောက်ဘဲ တိုက်ရိုက်မသုံးဘဲ အခု obj_truck နဲ့ obj_spot ဆိုတဲ့ object နှစ်ခုတည်ဆောက်ပြီး ခေါ်သုံးပါတယ်။မတူညီတဲ့ အဖြေတွေထွက်လာမှာပါ ဒါက polymorphism နည်းစနစ်ဘဲ ဖြစ်ပါတယ်။

Inner Class In Python

Class တစ်ခုတည်ဆောက်ထားပြီး အဲဒီ Class ထဲမှာ အခြား class တစ်ခုထပ်မံတည်ဆောက်အသုံး ပြုတာ ကို inner class လို့ခေါ်ပါတယ်။အဲဒီ inner class က လည်း OOP ရေးနည်းစနစ် တစ်မျိုး ဘဲ ဖြစ်ပါတယ်။

```
class Ai:
    def __init__(self):
        print("I am Ai")

    class Brain:
        def __init__(self):
            print("I am Brain")
        def mywork(self):
            print("I am For calculation")


obj_ai = Ai()
Ai.Brain().mywork()
```

ဒီမှာ Ai ဆိုတဲ့ class တစ်ခုရှိပြီး __init__ constructor တစ်ခုပါဝင်ပါတယ် အဲဒီ constructor ထဲမှာ print တစ်ခု ထုတ်ထားပါတယ်။နောက်ပြီး အဲဒီ Class ထဲမှာဘဲ နောက် Class တစ်ခုကို Inner Class အဖြစ် တည်ဆောက်ပြီးတော့ constructor နဲ့ mywork လို့ခေါ်တဲ့ method တစ်ခုပါဝင်ပါတယ်။အဲဒီတော့ ခေါ်သုံးဖို့အတွက် Ai.Brain().mywork() လို့ခေါ်မှ အလုပ်လုပ်ပါတယ်။ဒါက Inner Class ရဲ့ Example တစ်ခုဖြစ်ပါတယ်။

Magic Methods in Python

Magic Methods ဆိုတာ သာမန် တည်ဆောက်ထားတဲ့ method ပုံစံမဟုတ်တာတွေပေါ့။Constructor မှာဆိုရင် __init__ ဆိုတာမျိုးပေါ့ ထူးထူးဆန်းဆန်းပုံစံမျိုးနဲ့ပေါ့ဗျာ ခေါ်တဲ့အခါမှာလဲ underscore underscore init underscore underscore ဆိုတော့ ဒီဟာလေးတစ်ခါပြောဖို့ကိုအတော်အချိန်ယူရမှာဖြစ်ပါတယ်။ဒါကြောင့် double underscore init doucble underscore လို့လဲတချို့ကခေါ်တယ်။ဒါပေမယ့် အကောင်းဆုံးအခေါ်ဝေါ်ကတော့ dunder init duner လို့ခေါ်တာပါဘဲ။အခေါ်အဝေါ်တွေက အရေးမကြီးပေမယ့် dunder method လို့ပြောလိုက်ရင် underscore တွေပါတဲ့ method တွေလိုပြေးမြင်ရမှာပါ။ဒါတွေ ကို megic method လို့လဲခေါ်ပါတယ် ဘာကြောင့်လဲဆိုတော့ ထူးခြင်းတဲ့ လုပ် ဆောင် ချက် တွေကို လုပ်ဆောင် ပေးနိုင်လို့ဖြစ်ပါတယ်။ကျွန်တော်တို့ class တစ်ခုကို object တည်ဆောက်တဲ့ အခါမှာ နောက်ကွယ် မှာလုပ်ဆောင်ချက်တွေရှိပါတယ်။ဥပမာ class A ကို x object တည်ဆောက်လိုက်တဲ့အခါမှာ x = A()

လို့လုပ်ဆောင်လိုက်တဲ့အခါမှာ Python က နောက်ကွယ်မှာ __new__ နဲ့ __init__ ကိုခေါ်ဖို့လိုအပ်ပါတယ်။ OOP မှာတော့ Operator Overloading လုပ်ဖို့အတွက် magic methods တွေလိုအပ်ပါတယ်။ဒါဆိုရင် Operator Overloading ဆိုတာဘာလဲကြည့်ရအောင်။အရင်ဆုံးတော့ magic methods တွေကိုကြည့်ရအောင်။

Binary Operators Magic Methods	
Operator	Methods
+	object.__add__(self,other)
_	object.__sub__(self,other)
*.	Object.__mul__(self,other)
//	object.__floordiv__(self,other)
/	object.__truediv__(self,other)
%	object.__mod__(self,other)
**	object.__pow__(self,other)
<<	object.__ishift__(self,other)
>>	object.__rshift__(self,other)
&	object.__and__(self,other)
^	object.__xor__(self,other)
|	object.__or__(self,other)

Extended Assignments	
Operator	Method
+=	object.__iadd__(self,other)
-=	object.__isub__(self,other)
*=	object.__imul__(self,other)

/=	object.__idiv__(self,other)
//=	object.__ifloordiv__(self,other)
%=	object.__imod__(self,other)
**=	object.__ipow__(self,other)
<<=	object.__ilshift__(self,other)
>>=	object.__irshift__(self,other)
&=	object.__iand__(self,other)
^=	object.__ixor__(self,other)
|=	object.__ior__(self,other)

Unary Operators	
Operators	Method
-.	object.__neg(self)
+	object.__pos__(self)
abs()	object.__abs__(self)
~	object.__invert__(self)
complex()	object.__complex__(self)
int()	object.__int__(self)
long()	object.__long__(self)
float()	object.__float__(self)
oct()	object.__oct__(self)
hex()	object.__hex__(self)

Comparison Operators	
Operators	Method
<	object.__it__(self,other)
<==.	Object.__le__(self,other)
==	object.__eq__(self,other)
!=	object.__ne__(self,other)
>=	object.__ge__(self,other)
>	object.__gt__(self,other)

Magic methods တွေက အများကြီးဖြစ်ပါတယ်။ဒါတွေအားလုံးအသုံးပြုဆိုတာ တကယ်တော့လဲ မလွယ်ပါဘူး ဒါတွေမှမဟုတ်ဘဲ သချာ်ဆိုင်ရာ methods တွေလဲရှိပါသေးတယ်။ဒါဆိုရင် အရင်ဆုံး __add__ ကို အသုံး ပြုကြည့်ပါမယ်။

```
class add:
    def __init__(self,x):
        self.x = x



    def __add__(self, other):
        return add (self.x+other)


fi = add(8)+8
print(fi.x)
```

ဒီ program မှာ class တစ်ခုတည်ဆောက်ထားပြီး constructor တစ်ခုတည်ဆောက်ထားတယ် ပြီးတော့ paramenter တစ်လုံးလက်ခံထားပါတယ်။နောက်ထပ် method ကတော့ __add__ ဆိုတဲ့ magic method ဘဲဖြစ်ပါတယ်။အဲဒီထဲမှာလုပ်တဲ့ အလုပ်က return add() ဆိုပြီး class စီကို return ပြန်ထားတယ် လုပ်ထား တဲ့ အလုပ်က self.x + other ဖြစ်ပါတယ်။other ဆိုတာ dunder method ရဲ့ argunment ဖြစ်ပါတယ်။ကျွန်တော်တို့က class ကိုခေါ်သုံးတော့ class ကို object အဖြစ်ပြောင်းပါတယ် ပြီးတော့ ခေါ်တဲ့နေရာမှာ add(8)+8 ဆိုပြီးခေါ်ထားပါတယ် မူရင်း class ကို 8 ထပ်ထည့်ပြီး overloading လုပ်သွားတဲ့သဘောပါ operator overloading ပေါ့။ဒီတော့ self.x ထဲကို 8 ရောက်ပြီး other ထဲကို 8 ရောက်ပါတယ်။အဖြေကိုခေါ်သုံးတဲ့ အခါမှာ object ကိုခေါ်ပါတယ် fi.x ဆိုပြီး ပြန်ခေါ်တဲ့ အတွက်မှာ overload ဖြစ်သွားတဲ့ x ကိုရမှာဖြစ်ပါတယ်။အခြား သော အနှုတ် အမြောက် ထပ်တိုး တွေလဲ အတူတူပါဘဲ program ဥပမာ တချို့ တင်ပေးလိုက်ပါတယ်။

```
class sub:
    def __init__(self,y):
        self.y = y
```

```
    def __sub__(self, other):
        return sub (self.y - other)

su = sub(9)-7
print(su.y)

class mul:
    def __init__(self,z):
        self.z = z

    def __mul__(self, other):
        return mul(self.z * other)

mul = mul(9)*9
print(mul.z)


class pow:
    def __init__(self,sq):
        self.sq = sq

    def __pow__(self, power, modulo=None):

        return pow(self.sq**power)

pow = pow(6)**3
print(pow.sq)
```

ဒါဆိုရင် နောက်ထပ် Program တစ်ခုလောက်တည်ဆောက်ကြည့်ရအောင်။ဒီမှာတော့ Built-in Method တွေ အနေနဲ့ __str__ နဲ့ __repr__ ဆိုတဲ့ method နှစ်ခုကို သုံးပဲ့မယ်။

```
class Length:
    __metric =
{"mm":0.001,"cm":0.01,"m":1,"km":1000,"in":0.0254,"ft":0.3048,"yd":0.9144,"mi":1609.34
4}

    def __init__(self,value,unit="m"):
        self.value = value
        self.unit = unit
    def conver2metre(self):
        return self.value * Length.__metric[self.unit]
    def __add__(self, other):
        l = self.conver2metre() +other.conver2metre()
        return Length(l/Length.__metric[self.unit],self.unit)
    def __str__(self):
        return str(self.conver2metre())
    def __repr__(self):
        return "Length(" + str(self.value)+",'" + self.unit +"')"
if __name__ == "__main__":
        x = Length(4)
        print(x)
        z = Length(4.5, "yd") + Length(1)
        print(repr(z))
        print(z)
```

ဒီ Program မှာ Length ဆိုတဲ့ class တစ်ခုကိုတည်ဆောက်ထားပြီးတော့ __metric ဆိုတဲ့ dictionary တစ်ခုကို private အဖြစ်တည်ဆောက်ထားပါတယ်။အဲဒီထဲမှာ Key နဲ့ Value တွေပါဝင်ပါတယ်။အဲဒီထဲမှာ unit တွေနဲ့ သူတို့ စံစနစ်တွေအတွက် ကိန်းဂဏန်းတွေတည်ဆောက်ထားပါတယ်။ပြီးတဲ့အခါမှာ __init__ method ကိုတည်ဆောက်တယ်။အဲဒီထဲမှာ value နဲ့ unit ဆိုတဲ့ Argunments နှစ်ခုပါဝင်ပါတယ်။unit ကတော့ defult

value အနေနဲ့ "m" ကိုထည့်သွင်းပေးထားပါတယ်။အရှေ့မှာလည်း ကျွန်တော်ပြောခဲ့ပြီးပါပြီး defult value ထည့်သွင်းခြင်းအားဖြင့် passing လုပ်လာတဲ့အခါမှာ ဘာ value မှ passing မဖြစ်ရင်တောင် သူ့ရဲ့ defult value က အလုပ်လုပ်သွားမှာဖြစ်ပါတယ်။value နဲ့ unit ကို self.value နဲ့ self.unit ထဲကို pass လုပ်ထားပြီးပါတယ်။နောက်ထပ် method ကတော့ conver2metre() ဖြစ်ပါတယ်။အဲဒီထဲမှာတော့ အပေါ်က self.value နဲ့ Length.__metric[self.unit] နဲ့ကို မြောက်ထားတဲ့ တန်ဖိုးကို return ယူထားပါတယ်။Length.__metric[self.unit] ဆိုတာ အပေါ်မှာတည်ဆောက်ထားတဲ့ private dictionary နော် အဲဒီထဲမှာ Key ကို self.unit အဖြစ် passing လုပ်ထားတာ။အဲဒီတော့ self.unit ထဲကို mm လို့ ဂင်လာရင် self.value နဲ့ 0.001 နဲ့ မြောက်မယ် cm ဂင်လာရင် self.value နဲ့ 0.02 မြောက်လိမ့်မယ် သူ့ရဲ့ Key အလိုက် value တွေရလာပြီး မြောက်မှာဖြစ်ပါတယ်။နောက်ထပ် method တစ်ခုကတော့ operator overloading လုပ်ခဲ့ရင် လုပ်မယ့် method __add__ ဖြစ်ပါတယ်။အဲဒီထဲမှာကတော့ l ဆိုတဲ့ variable ထဲကို self.conver2metre() + other.conver2metre() ဆိုပြီး ဒီ နှစ်ခုပေါင်းတာကိုထပ်ထည့်ထားပါတယ်။ပြီးရင်တော့ (l/Length.__metric[self.unit],self.unit) l ကို Length.__metric[self.unit] နဲ့စားထားတဲ့ တန်ဖိုးကို class Length ထဲကို return ပြန်ထားပါတယ်။နောက် __str__ ကတော့ retrun str(self.conver2metre()) အလုပ်ကိုလုပ်ထားတယ် __str__ ဆိုတဲ့ method အကြောင်းကို အနည်းငယ်ပြောရအောင်။__str__ ဆိုတာ ဂင်လာတဲ့ တန်ဖိုးတွေကို String ပြောင်းပြီး အဖြေထုတ်ပေးပါတယ်။နောက်တစ်ခုက __repr__ ကတော့ String ပြောင်းတဲ့နေရာမှာ printable ပုံစံပေါ့ ကျွန်တော်တို့ Format ကြကြထုတ်ချင်တဲ့ အခါမှာ သုံး နိုင်တဲ့ အမျိုးစားဖြစ်ပါတယ်။if__name__ == "__Main__"ဆိုတာကတော့ Program စတင်တဲ့ Main အဖြစ် သတ်မှတ်ထားတာပါပြီးတော့ x ဆိုပြီး object တည်ဆောက်ပြီး Length(4) ဆိုပြီး Passing ကို 4 လုပ်လိုက်ပါတယ်။ပြီးတော့ print(x) ကိုထုတ်ပါတယ်။နောက်တစ်ခုကတော့ z ဆိုတဲ့ object ကိုတည်ဆောက်ပြီး operator overloading လုပ်ပါတယ်။ပြီးတဲ့အခါမှာ repr ကိုခေါ်ပြီး အဖြေထုတ်တာနဲ့ ဒီအတိုင်း z ကိုထုတ်တာ လဲမတူညီတာတွေ့ရမှာပါ။

Accessing Databases

ကျွန်တော်တို့ အခုနောက်ပိုင်း software တွေ computing system တွေမှာ data တွေ သိမ်းဆည်းဖို့မဖြစ်မနေ Database တွေပါဂင်လာပါတယ်။ကျွန်တော်တို့အရှေ့မှာ Dictionary တွေ Tuple တွေ List တွေမှာသိလှောင်တဲ့ data တွေဟာ software ပိတ်လိုက်တဲ့အခါမှာ ကျွန်တော်တို့ သိမ်းဆည်းမှတ်သားသမျှ တွေ ပျောက်သွားပါတယ်။ဒီပြဿနာတွေဖြေရှင်းဖို့အတွက်က data တွေကိုမပျက်စေဘဲ မှတ်ထားပေးမယ့် database တွေလိုအပ်လာတာပါ။Website တွေ www.amazon.com , www.facebook.com စတဲ့ website တွေမှာလဲ database တွေနဲ့အလုပ်တွဲလုပ်ပြီး data တွေကိုသိမ်းပါတယ်။ကျွန်တော်တို့လဲ Python နဲ့ program တွေရေးတဲ့အခါမှာ Data တွေသိမ်းဆည်းဖို့လိုလာရင် Database တွေကိုအသုံးပြုရပါတော့မယ် ကံ ကောင်းတာ တစ်ခုကတော့ Python မှာ Database တွေနဲ့ ချိတ်ဆက်ဖို့ API လို့ခေါ်တဲ့ အထောက်ပံ့ တွေ

ရှိနေပါတယ်။ဒါကြောင့် လွယ်လွယ်ကူကူဘဲ Database တွေနဲ့ ကျွန်တော်တို့ Program ကိုချိတ်ဆက်နိုင် မှာ ဖြစ်ပါတယ်။Database ကတော့ အမျိုးမျိုးရှိပါတယ် ကျွန်တော်ကတော့ sqlite , MySQL , DBM တို့ကို ဘဲ အသုံး ပြုသွားမှာဖြစ်ပါတယ်။

DBM with Python Program

DBM module ကို အသုံးပြုပြီးတော့ Key နဲ့ Value အတွဲလိုက် Data တွေကို သိမ်းဆည်းနိုင် ပါတယ်။ပြောရရင် အရှေ့မှာ ပြောခဲ့တဲ့ Dictionary အမျိုးစား Data တွေဖြစ်ပါတယ် ဒီတော့ DBM ကိုလဲ Dictionary Database လို့တွေခေါ်နိုင်ပါတယ်။DBM ကို C library နဲ့တည်ဆောက်ထားတာဖြစ်ပြီး UNIX အတွက် တည်ဆောက်ထားတာဖြစ်ပါတယ်။ဒီမှာတော့ Python မှာ DBM module ကိုအသုံးပြုနိုင်ပါတယ်။ Python အတွက်ကို dbm module ရယ်၊dbm.dumb ရယ် dbm.gnu ဆိုပြီးဖြစ်ပါတယ်။dbm ကတော့ အကောင်းဆုံးရွေးချယ်ရမှာဖြစ်ပါတယ်။dbm.dumb ကတော့ portable ဖြစ်ပါတယ်။dbm module ကို implementation လုပ်ထားတာဖြစ်ပြီးတော့ dbm.gnu ကတော့ GNU library အမျိုးစားဖြစ်ပါတယ်။ဒါဆိုရင် အသုံးပြုကြည့်ရအောင်။

```
import dbm

db = dbm.open("dbm_database",'c')

db["AungMoeKyaw"] ="09XXXXXXXX"
db["PhoeThar"] ="09XXXXXXXX"
db["SanWin"] = "09XXXXXXXX"
db["A"] ="09XXXXXXX"
db["Phoe"] ="09XXXXXXX"
db["San"] = "09XXXXXXX"
db["MoeKyaw"] ="09XXXXXXX"
db["Thar"] ="09XXXXXXX"
db["Win"] = "09XXXXXXX"
db["Moe"] ="09XXXXXXX"
db["P"] ="09XXXXXXX"
db["S"] = "09XXXXXXX"
print(db["AungMoeKyaw"])
print(db["PhoeThar"])
db.close()
```

dbm module က open ဆိုတဲ့ method နဲ့တွဲသုံးပါတယ် အရင်အခန်းမှာလဲ File Read / File Write အခန်း မှာပြောခဲ့ပြီးပါပြီ open method အကြောင်းကို ဒီမှာ အရင်ဆုံး import dbm ဆိုပြီး dbm ကို import လုပ် ပါ တယ်ပြီးတဲ့အခါမှာ dbm.open နဲ့ ဖိုင်ကိုဖွင့်ရပါတယ်() ဒီလက်သည်းကွင်းထဲမှာ database အနေနဲ့ချိတ် ဆက်မယ့် ဖိုင်ရယ် access လုပ်မယ့် command ဖြစ်ပါတယ်။ဒီမှာတော့ c ကို အသုံးပြုထားပါတယ် c ဆိုတာ တည်ဆောက်ထားတဲ့ဖိုင်မရှိနေရင် တည်ဆောက်မယ် File Read ရော Write ရောလုပ်လို့ရပါတယ်။နောက် တစ်မျိုးကတော့ N ဖြစ်ပါတယ် ဒီကောင်က တည်ဆောက်ပြီးသားဖိုင်ရှိနေလဲ overwrite လုပ်ပြီး ထပ်ခါ ထပ်ခါ တည်ဆောက်လို့ ဖိုင်အသစ်ယူသလိုဖြစ်ပြီး data တွေကျန်မှာမဟုတ်ပါဘူး။W ကတော့ ဖိုင်မရှိနေဘူးဆိုရင် အသစ်မတည်ဆောက်နိုင်ပါဘူး။ဒီမှာ dictionary တည်ဆောက်သလိုဘဲ db[key] = value ကိုထည့်ရေး

တည်ဆောက်ပါတယ်။ပြီးတော့ အဖြေထုတ်ချင်တဲ့အခါမှာ db[key] ကိုရေးလိုက်တဲ့ အခါမှာ အဖြေကို ရစေပါတယ်။ပြီးရင်တော့ မဖြစ်မနေ db.close() ဆိုပြီး ပိတ်ရပါတယ် ဒါက File Save ခြင်း အလုပ်နဲ့ db ကို အလုပ်လုပ်စေဖို့အတွက် အသုံးပြုရတာဖြစ်ပါတယ်။

```
import dbm

db = dbm.open("dbm_database",'c')

db["AungMoeKyaw"] ="09XXXXXXXX"
db["PhoeThar"] ="09XXXXXXXX"
db["SanWin"] = "09XXXXXXXX"
db["A"] ="09XXXXXXXX"
db["Phoe"] ="09XXXXXXXX"
db["San"] = "09XXXXXXXX"
db["MoeKyaw"] ="09XXXXXXXX"
db["Thar"] ="09XXXXXXXX"
db["Win"] = "09XXXXXXXX"
db["Moe"] ="09XXXXXXXX"
db["P"] ="09XXXXXXXX"
db["S"] = "09XXXXXXXX"
print(db["AungMoeKyaw"])
print(db["PhoeThar"])

for x in db.keys():
    print(x)
for y in db.values():
    print(y)
db.close()
```

Key တွေ Value တွေအကုန်လုံးကို အဖြေထုတ်စေဖို့အတွက် db.keys() သို့မဟုတ် db.values() ဆိုတဲ့ functions နှစ်ခုကို အသုံးပြုပြီး Key တွေ Value တွေကို အဖြေထုတ်လို့ရပါတယ်။

```
import dbm

db = dbm.open("dbm_database",'c')

db["AungMoeKyaw"] ="09XXXXXXXX"
db["PhoeThar"] ="09XXXXXXXX"
db["SanWin"] = "09XXXXXXXX"

print(db["AungMoeKyaw"])
print(db["PhoeThar"])

for x in db.keys():
    print(x)
for y in db.values():
    print(y)

del db["AungMoeKyaw"]
print("After Deleting Key AungMoeKyaw")
for x in db.keys():
    print(x)

db.close()
```

တန်ဖိုးတွေဖျက်ခြင်းအလုပ်ကို Dictionary မှာလိုဘဲ del ကိုအသုံးပြုပြီး ဖျက် လို့ရပါတယ်။ဒီ အပေါ်က Program မှာဆိုရင် del နဲ့AungMoeKyaw ကိုဖျက်ပြီးတဲ့အခါမှာ keys() တွေကိုထပ်အဖြေထုတ်တဲ့ အခါ AungMoeKyaw ဆိုတဲ့ Key မပါလာတော့တာတွေ့ရမှာပါ။

```
import dbm

db = dbm.open("dbm_database",'c')

db["AungMoeKyaw"] ="mcoder's founder"
db["PhoeThar"] ="AungMoeKyaw's Friend"
db["SanWin"] = "AungMoeKyaw's brother"

if db["AungMoeKyaw"] != None:
    print("Found",db["AungMoeKyaw"])
else:
    print("That is Missing!!")

del db["AungMoeKyaw"]

print("After Deleting Key AungMoeKyaw")
for x in db.keys():
    print(x)

db.close()
```

ဒီ Program မှာတော့ if နဲ့ db[“AungMoeKyaw”] != None: ဆိုပြီးစစ်ဆေးထားတာပါ AungMoeKyaw ဆိုတဲ့ Key ထဲမှာ ဘာ value မှမပါရင် ပြောရရင် None ဆိုတာ ဘာတန်ဖိုးမှမပါဘဲ return အမျိုးစားဖြစ်ပါတယ် ဒီဟာကို not equal ! ဆိုတော့ none မဖြစ်ရင် Found ဆိုတဲ့စာသားနဲ့ value နဲ့ကိုတွဲအဖြေပြမယ် မဟုတ်ဘူးဆိုရင်တော့ That is Missing လို့ပြမယ်လို့ရေးထားတာဖြစ်ပါတယ်။

```
import dbm

db = dbm.open("dbm_database",'c')

db["AungMoeKyaw"] ="mcoder's founder"
db["PhoeThar"] ="AungMoeKyaw's Friend"
db["SanWin"] = "AungMoeKyaw's brother"

if db["AungMoeKyaw"] != None:
    print("Found",db["AungMoeKyaw"])
else:
    print("That is Missing!!")

del db["AungMoeKyaw"]

print("After Deleting Key AungMoeKyaw")

try:
    print(db["AungMoeKyaw"])
except KeyError:
    print("Not Found")
else:
    print(db["AungMoeKyaw"])

db.close()
```

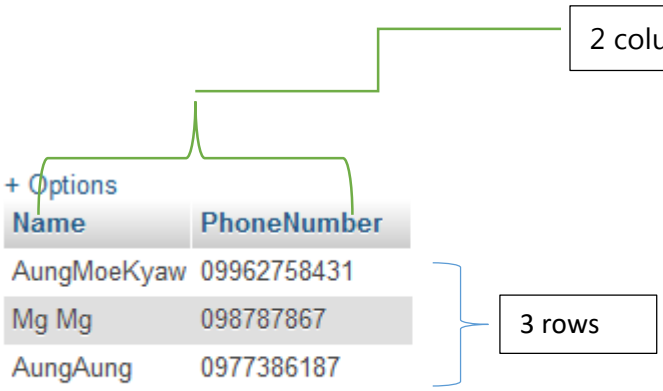
မရှိတဲ့ Key ကိုဆွဲထုတ်မိတဲ့အခါမှာ သို့မဟုတ် del လုပ်ထားတဲ့ Key ကိုအဖြေထုတ်ဖို့ သို့မဟုတ် ယူ သုံး ဖို့ကြိုးစားတဲ့အခါမှာ Program Error တတ်ပြီးရပ်မသွားအောင် try: နဲ့ except လုပ်ထားပါတယ် တကယ်လို့ Key မရှိဘူးဆိုရင် တတ်မယ့် Error က KeyError ဖြစ်ပါတယ် KeyError တတ်ခဲ့ရင် Not Found ကိုပြမယ် မဟုတ်ရင်တော့ အလုပ်ဆက်လုပ်သွားမယ်လို့ရေးထားပါတယ်။AungMoeKyaw ဆိုတဲ့ Key က del လုပ်ထားပြီးသားဖြစ်လို့ Not Found ဖြစ်ပါလိမ့်မယ်။

ဒါဆိုရင် dbm သုံးပြီး data တွေသိုလှောင်မှု့ကိုသိရပြီဖြစ်ပါတယ်။ဒါပေမယ့်စိတ်မကောင်းစွာဘဲ ပြဿနာ တွေရှိနေပါတယ် တချို့သော dbm database တွေက 1024byte ကိုသာသိမ်းဆည်းနိုင်ပါတယ် 1KB ပေါ့ဗျာ ဒါက တကယ်ကိုသေးငယ်ပြီးဘာမှလုပ်လို့မဖြစ်တဲ့ ပမာဏဖြစ်ပါတယ်။နောက်တစ်ခုက ရှုပ်ထွေးပြီး အဆမတန်များတဲ့ Data တွေအတွက်လဲ ဒီ dbm ကသုံးလို့မဖြစ်ပါဘူး။dbm database file တွေက .dat နဲ့ ဖြစ်ပါတယ်။နောက်တစ်ခုပြောချင်တာက ပမာဏနည်းပြီး လွယ်လွယ်ကူကူရိုးရိုးရှင်းရှင်းသိမ်းဆည်းရမယ့် data တွေကိုသိမ်းဆည်းဖို့အတွက် dbm ကိုအသုံးပြုနိုင်ပါတယ်ခက်ခက်ခဲခဲများများပြားပြားတွေ သုံးဖို့ကတော့ relational database တွေကိုအသုံးပြုရမှာဖြစ်ပါတယ် ဒီတော့ Relational Database တွေ အကြောင်းကိုဆက်ပြောသွားရအောင်။

Relational Database with Python

Relational Database တွေ နည်းပညာလောကမှာနေရာယူနေတာ ဆယ်စုနှစ်တွေကြာနေပါပြီ ဒါကြောင့် နည်းပညာလောကနဲ့မစိမ်းဘဲ အရွယ်ရောက်နေပြီလို့တောင်ဆိုနိုင်ပါတယ်။လူတွေကလဲ သိနေပါ ပြီ ဒီ Relational Database တွေ ဘယ်လိုအလုပ်လုပ်လဲ ဘယ်လိုအထောက်ကူပေးလဲ ဆိုတာ ကို မသိရင်လဲ အခုသိရတော့မှာပါ ဒါပေမယ့် တစ်ခုမှတ်ထားရမှာပါက ရှုပ်ထွေးပြီးများပြားတဲ့ Data သိမ်းဆည်းမှု့တွေ အတွက် ရွေးချယ်ရာကတော့ Relational Database နည်းပညာဘဲဖြစ်ပါတယ်။website တွေ android apk တွေ Desktop software တွေ Web app တွေအကုန်လုံးက ဒီ Relational Database ကို အသုံးပြုကြပါတယ် Data တွေသိုလှောင်ဖို့ပေါ့။

Relational Database တွေမှာ data တွေကို table တည်ဆောက်ပြီးတော့ Two-dimensional data structure ပုံစံနဲ့သိမ်းဆည်းပါတယ်။vertical (ထောင်လိုက်) Two-dimensional matrix ကို Columns လို့ ခေါ်ပြီးတော့ horizontal (အလျားလိုက်) ကိုတော့ row လို့ခေါ်ပါတယ်။ဒါတွေကို records တွေလို့လဲခေါ်ပါ တယ်။Columns တွေရဲ့ Data Type တစ်ခုခြင်းစီက တူညီတဲ့ DataType တွေဖြစ်ပါတယ်။Data Type တွေ အနေနဲ့တော့ integer INT, String TEXT, Char VARCHAR, DATE စတာတွေကိုသိမ်းဆည်းနိုင် ပါတယ်။



ဒီမှာဆိုရင် Table တစ်ခုတည်ဆောက်ထားပါတယ် Columns 2 ခုဖြစ်တဲ့ Name ရယ် PhoneNumber ရယ် နဲ့ row ကတော့ သုံးခုဖြစ်ပါတယ် နာမည်တွေနဲ့ ဖုန်းနာပတ်တွေပါဂင်ပါတယ်။ကြီးမားတဲ့ လုပ်ငန်းတွေမှာ ဆိုရင် တော့ Data သိမ်းဆည်းရတာလဲများပြားပြီးတော့ Columns တွေ Rows တွေလဲ သိန်းချီ သောင်းချီ ရှိနိုင်ပါတယ်။

ဒီ Database တွေကို အသုံးပြုဖို့အတွက် Database နားလည်တဲ့ ဘာသာစကားတစ်ခုကို သုံးရ ပါမယ် အဲဒါကတော့ SQL Language လို့ခေါ်ပါတယ် အသံထွက်ကို squel(စီးကွဲ) လို့ထွက်နိုင်သလို sql လို့ လဲ အလွယ် တကူထွက်နိုင်ပါတယ် အင်းတကယ်တော့ အခေါ်ဂေါ်က အရေးမကြီးပါဘူး။အခြေခံအကျဆုံး အသုံး အများဆုံး SQL Statement တွေကြည့်ရအောင်။

Operation	Usage
Select	Perform a query to search the database
Update	Modify a row or rows,usually based on certain condition
Insert	Create new rows in the database
Delete	Remove a row or rows from the database

Select ကတော့ database ထဲက data တွေကိုရွေးချယ်ထုတ်ယူဖို့အတွက် အသုံးပြုနိုင်သလို data တွေ ရှာတဲ့ အခါမှာလဲ အသုံးပြုနိုင်ပါတယ်။Update ကတော့ row ဒါမှမဟုတ် rows တွေကို အသစ်ပြောင်း Update လုပ်တဲ့ အခါမှာအသုံးပြုပါတယ်။Insert ကတော့ rows အသစ်ဖန်တီးပြီး data အသစ်တွေထည့်တဲ့ အခါမှာ အသုံး ပြုသလို Delete ကတော့ row or rows တွေကိုဖျက်တယ်။ဒါတွေကိုယေဘုယ အားဖြင့် QUID လို့ ခေါ်ပါတယ်။Query , Update , Insert ,Delete ပေါ့ ဒါမှမဟုတ်ရင် CRUD လို့လဲ ခေါ်ပါသေးတယ် Create , Read, Update, Delete ပေါ့။SQL မှာလုပ်ဆောင်ချက်တွေအများကြီးရှိပါတယ် ဒါပေမယ့် အရေးကြီးဆုံး ကတော့ ဒါတွေပေါ့။တကယ်လို့ ခင်ဗျားတို့ဟာ SQL နဲ့ စိမ်းနေတယ်ဆိုလဲမပူပါနဲ့ ကျွန်တော် ရဲ့ Blog မှာ မကြာခင် SQL နဲ့ပတ်သတ်တဲ့ အကြောင်းအရာတွေ အခန်းဆက်တင်သွားပေးပါ့မယ်။SQL Statements တွေက အရေးကြီးပါတယ် Python က API ကိုသုံးပြီး database ကို create လုပ်မယ် သို့မဟုတ် ချိတ်မယ် ပြီးရင် ဒီ database ထဲကို data တွေ insert လုပ်တာ delete လုပ်တာ update လုပ်တာ columns တည်ဆောက်တာ rows တည်ဆောက်တာ table တည်ဆောက်တာ အားလုံးကို SQL ကိုသုံးပြီးလုပ်ရမှာပါ Python API ထဲမှာ SQL ကိုထည့်ရေးပြီးအားလုံးလုပ်ဆောင်ရမှာဖြစ်ပါတယ်။Python မှမဟုတ်ဘူး Java ရော C# ရော C++ ရော PHP ရော Android မှာရော SQL support database ကိုသုံးတဲ့ အခါမှာ ဒီ SQL Language Statements တွေကိုဘဲအသုံးပြုရမှာဖြစ်ပါတယ်။ဒါဆိုရင် Program တစ်ခု ရေးကြည့်ရအောင်။အရင်ဆုံး ကျွန်တော်တို့ sqlite database ကိုအသုံးပြုရအောင်။

```
import sqlite3

sql_create_table ="""CREATE TABLE IF NOT EXISTS student_name(id INTEGER PRIMARY
```

```
KEY,name TEXT NOT NULL,
                    phone TEXT);"""
insert_data  = """INSERT INTO student_name(name,phone) VALUES(?,?)"""

select_data = """SELECT * FROM student_name"""

con = sqlite3.connect('student.db')
c   = con.cursor()
c.execute(sql_create_table)

insert = con.cursor()
insert.execute(insert_data,("AungMoeKyaw","09962758431"))
insert.execute(insert_data,("HtikeSanWin","09969340672"))

select = con.cursor()
all_data = select.execute(select_data)

show = all_data.fetchall()

for x in show:
    print(x)
```

sqlite ကိုအသုံးပြုမှာဖြစ်တဲ့အတွက် sqlite3 module ကို import လုပ်ပါတယ်။sqlite က အရမ်းကြီးတဲ့ website တွေ မှာသုံးလို့အဆင်မပြေပေမယ့် အားသာချက်တွေအများကြီးရှိတဲ့ database အမျိုးစားဖြစ်ပါတယ်။ပေါ့ပါးတယ်၊ဆာဗာမလိုဘူး၊အမှီခိုကင်းကင်းနဲ့ဘယ် OS မှာမဆိုတည်ဆောက်နိုင်တယ် programming တော်တော်များများနဲ့တွဲဖက်အသုံးပြုနိုင်တဲ့ စတဲ့ အားသာချက်တွေကြောင့် sqlite3 က နာမည်ကျော်ကြားလာပါတယ် အခုတော့ ထိတွေ့ကြည့်ရအောင်။အပေါ်ဆုံးက variable တွေကို ခန မေ့ထားပါ။အခုအောက်က command တွေကိုလေ့လာကြည့်ရအောင်။

object တစ်ခုကိုတည်ဆောက်လိုက်ပါတယ် conn ဆိုပြီး ပြီးတော့ sqlite3.connect('student.db') ဆိုပြီး database နဲ့ချိတ်ဆက်လိုက်ပါတယ် database ဖိုင်မရှိသေးရင်လဲပြသာနာမရှိပါဘူး အလိုလိုတည်ဆောက်သွားပါလိမ့်မယ် database နာမည်ကတော့ student.db ဖြစ်ပါလိမ့်မယ် .db ကတော့ extension ဖြစ်ပါတယ်။ဒါဆိုဆက်ကြည့်ရအောင် နောက်ထပ် object တစ်ခုတည်ဆောက်ပါတယ် c ဆိုတဲ့နာမည်နဲ့ပေါ့ အဲဒီထဲမှာတော့ con.cursor() ဆိုပြီးသုံးထားပါတယ် con ဆိုတာ database object ဖြစ်ပါတယ်။cursor() တော့ connection object လို့ခေါ်ပါတယ် database ချိတ်ဆက်မှုအတွက်ဖြစ်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့မှာ Database တစ်ခုရပြီ ချိတ်ဆက်ဖို့အတွက်လဲပြင်ဆင်ပြီးပြီ လိုအပ်တာ နောက်တစ်ခုကတော့ Table ဘဲ ဖြစ်ပါတယ် ကျွန်တော်တို့က Database ထဲမှာ Table တည်ဆောက်ပြီး data တွေသိမ်းတာဖြစ်တဲ့ အတွက် table တွေလိုအပ်ပါတယ်။ဒါဆိုရင် Table တည်ဆောက်ဖို့ connection object ဖြစ်တဲ့ c ကို ခေါ်ပြီး c.execute() ကိုခေါ်သုံးပါတယ် execute() ဆိုတာ ကျွန်တော်တို့ database ကနားလည်တဲ့ SQL ဘာသာစကားကိုရေးနိုင်အောင်ဖန်တီးထားတဲ့ method ပါ။execute(sql_create_table) ဆိုပြီး argument တစ်ခု passing လုပ်ထားပါတယ် အဲဒီ argument က ဘာတွေပါလဲဆိုတော့ SQL ဘာသာစကားနဲ့ရေးထားတဲ့ SQL Statement တွေဖြစ်ပါတယ်။sql_create_table ထဲမှာ CREATE TABLE IF NOT EXISTS ဆိုပြီးရေးထားပါတယ်။အဲဒီ အကြီးတွေနဲ့ရေးထားတာ SQL Statement တွေဖြစ်ပြီး အသေးတွေက နာမည်တွေ ဖြစ်ပါတယ်။

```
"""CREATE TABLE IF NOT EXISTS student_name(id INTEGER PRIMARY KEY,name TEXT NOT NULL,
                    phone TEXT);"""
```

ဒါက ဘာကိုပြောထားတာလဲဆိုတော့ student_name ဆိုတဲ့ နာမည်နဲ့ table မရှိဘူးဆိုရင် တည်ဆောက်မယ်လို့ပြောထားတာပါ အဲဒီ table တည်ဆောက်ပြီးသွားပြီဆိုရင် နောက်က round brackets ထဲမှာ id,name,phone ဆိုတဲ့ colunms 3 ခုပါဂင်ပါတယ် အဲဒီ column တစ်ခုခြင်းစီရဲ့နောက်မှာပါတဲ့ SQL Statement တွေကတော့ DataType တွေဖြစ်ပါတယ် id မှာ INTEGER PRIMARY KEY ဆိုတာ id ထဲထည့်မယ့် data ဟာ ကိန်းပြည့်ဖြစ်မယ် PRIMARY KEY ဆိုတာ AutoIncrement ဖြစ်တဲ့အမျိုးစားဖြစ်တယ်လို့ ပြောပါတယ်။ပြီးတော့ name က TEXT NOT NULL ဆိုပြီး name ထဲမှာထည့်မယ့်data က String စာသားအမျိုးစားဖြစ်ရမယ် ဘာမှမပါတဲ့ဗလာ မဖြစ်ရဘူး နောက်တစ်ခုက phone ထဲမှာတော့ TEXT ဆိုတော့ စာသားတန်ဖိုးဘဲသိုလှောင်မယ်လို့ပြောလိုက်တာပါဘဲ။ဒါဆိုရင် ကျွန်တော်တို့က columns 3 ခုပါတဲ့ table တစ်ခုကိုတည်ဆောက်ပြီးပါပြီ ဒါဆိုရင် data တွေထည့်ဖို့ဘဲကျန်ပါတော့တယ်။အဲဒီတော့ နောက်ထပ် connection object တစ်ခုတည်ဆောက်ပြီး ထပ်ထည့်ဖို့လုပ်မယ် (တကယ်တော့မလိုအပ်ပါဘူး အပေါ်က connection object ကိုဘဲခေါ်သုံးရင်ရပါတယ် ကျွန်တော်က အကျင့်ပါနေလို့ ဖျောက်ဖို့ကြိုးစားနေပါတယ် :D)ထားပါတော့ ဒီမှာလဲ execute() နဲ့ဘဲ အလုပ်လုပ်ပါမယ်။insert_data ထည့်ထားပါတယ် ဒီအထဲမှာ SQL က ဘာလဲဆိုတော့ INSERT INTO student_name(name,phone) VALUES(?,?) ဆိုပြီးရေးထားပါတယ် ဘာလဲဆိုတော့ INSERT INTO နဲ့ student_name ဆိုတဲ့ table ထဲက name နဲ့ phone ဆိုတဲ့ columns ထဲကိုထည့် data ထည့်မှာပါ VALUES(?,?) ဆိုတာကတော့ name နဲ့ phone ဆိုတဲ့ အထဲကို ထည့်မယ်လို့ ပြောတာပါ။

```
insert.execute(insert_data,("AungMoeKyaw","09962758431"))
insert.execute(insert_data,("HtikeSanWin","09969340672"))
```

VALUES(?,?) အနောက်မှာ , ထားပြီး (1,2) ထည့်တာဟာ။VALUES() ထဲကို Passing လုပ်လိုက်တာပါဘဲ၊အခုလိုရေးလိုက်တာဟာ (“AungMoeKyaw”,”09962758431”) ကို name နဲ့ phone ဆိုတဲ့ columns ထဲကိုထည့်လိုက်တာပါဘဲ။ဒါဆိုရင် Database ထဲကို Value တွေထည့်လိုက်တာပါဘဲ။ဒါဆိုရင် database ထဲက Values တွေကို ကျွန်တော်တို့ဘယ်လိုပြန်ထုတ်မလဲဆိုတာကိုလုပ်ကြည့်ရအောင် နောက်ထပ် connection object တစ်ခုတည်ဆောက်ပါတယ် ပြီးတော့ select.execute(select_data) ဆိုပြီး data ခေါ်ဖို့ SQL တစ်ခုကိုတည်ဆောက်ပါတယ် အဲဒီထဲက SQL ကတော့ SELECT * FROM student_name ပါ။SELECT ဆိုတာ မှတ်သားတာပါ * ကတော့ ပါသမျှ အားလုံးကိုရွေးချယ်တာပါ FROM နဲ့ student_name ဆိုတဲ့ table ထဲကအားလုံးကိုရွေးယူမယ်လို့ပြောတာပါ။variable တစ်လုံးတည်ဆောက်ပြီး Database ထဲက ထုတ်ထားတဲ့ data အားလုံးကို အဲဒီ variable ထဲကိုထည့်ပြီး for loop နဲ့ ပတ်ပြီး အဖြေထုတ်တဲ့ အခါမှာ ကျွန်တော်တို့ ထည့်ထားတဲ့ Value တွေအဖြေပြန်ထွက်လာမှာဖြစ်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ database program တစ်ခုလောက်ရေးရအောင် သူငယ်ချင်းတွေရဲ့ email တွေကို နာမည်နဲ့တွဲသိမ်းတဲ့ Program မျိုးပေါ့။ဒီတစ်ခေါက် OOP ပုံစံမျိုးရေးကြရအောင်။

```
import sqlite3
class database:

    def __init__(self):
        self.con = sqlite3.connect('email.db')
        print("Database is Connected Successfully")
```

```
        self.cursor = self.con.cursor()
        self.cursor.execute('CREATE TABLE IF NOT EXISTS emaildb(id INTEGER PRIMARY
KEY,name TEXT,email TEXT)')
        print("""
                        -------------------------------
                        -Table is Created Successfully-
                        -------------------------------\n""")

   def insert_data(self,name,email):
        self.name =  name
        self.email = email
        self.cursor = self.con.cursor()
        self.cursor.execute('INSERT INTO emaildb(name,email) VALUES
(?,?)',(self.name,self.email))
        self.con.commit()
        print("""
                        -------------------------------
                        -*Data Insert is Successfully*-
                        -------------------------------\n""")

class sen(database):
    def show(self):
        result = self.cursor.execute('SELECT * FROM emaildb')
        print("Some Thing Show YOU")
        print("--------------------")
        print("No.\t Name.\t \t \t Email")
        for x in result:
            print(x[0],"\t",x[1],"\t",x[2])









exit = ""
obj_2 = sen()



while exit != "100":
        print("\n")
        print("""Please Enter The Choice Number-
                        1.insert Data
                        2.Show Data
                        3.Exit
                       """)
        cn_num = input("Enter The Number:")
        if cn_num == "1":
            input_name = input("Enter Name:")
            input_email = input("Enter Email:")
            obj_2.insert_data(input_name, input_email)
            input("Press Any Key to Continues.....")

        elif cn_num == "2":
            obj_2.show()
            input("Press Any Key to Continues.....")

        elif cn_num == "3":
```

```
        exit = "100"
        print("GoodBye__")
```

program က အနည်းငယ်ရှည်သလိုဖြစ်နေပေမယ့် လည်း တစ်ဆင့်ခြင်း တစ်ခုခြင်းရှင်းလင်းသွားမယ်ဆိုရင် ဘာ ပြဿနာမှမရှိဘဲ နားလည်သဘောပေါက်စေမှာပါ။အရင်ဆုံး sqlite3 ကိုသုံးမှာဖြစ်လို့ import sqlite3 ဆို ပြီး database module ကို import လုပ်ပါတယ်။ပြီးတော့ OOP ပုံစံနဲ့ရေးမှာဖြစ်လို့ class ကိုတည်ဆောက်ပါ တယ် database ဆိုတဲ့နာမည်နဲ့ class ဖြစ်ပါတယ်။နောက်ပြီးအဲဒီ class ထဲမှာ constructor ဘဲ ပြောပြော dunder method ဘဲပြောပြော __init__ ကိုတည်ဆောက်ပါတယ် အဲဒီ method က ဒီ class ကို အခေါ်ခံ ရတာနဲ့ ချက်ခြင်း အလုပ်လုပ်မယ့် method ဖြစ်ပါတယ်။အဲဒီထဲမှာ database ကိုချိတ်ဆက်ထားပါတယ်။ database ကိုချိတ်ဆက်လိုက်တာနဲ့ Database is connected successfully ဆိုတဲ့ message ပေါ်မယ်ပြီးရင် self.cursor ဆိုတဲ့ connection object တစ်ခုတည်ဆောက်ပြီးတော့ connection တည်ဆောက်ပါတယ်။ပြီး တဲ့ အခါမှာ table တည်ဆောက်ပါတယ် Table မပါဘဲ Data တွေသိမ်းဆည်းလို့မရပါဘူး။Table တည် ဆောက် ပုံကတော့ connection object ကိုခေါ်ပြီး .execute() နဲ့ SQL ဘာသာစကားကို အသုံးပြု ပြီး တော့ Table တည်ဆောက်ပါတယ်။CREATE TABLE IF NOT EXISTS emaildb(id INTEGER PRIMARY KEY,name TEXT , email TEXT)') ဆိုပြီးဖြစ်ပါတယ် အပေါ်မှာလဲ ပြောခဲ့ပြီးပြီ emaildb ဆိုတဲ့ Table မရှိရင် တည်ဆောက်မယ်လို့ပြောတာဖြစ်ပါတယ် အဲဒီ Table ထဲမှာ Columns တွေတည်ဆောက်ထားတယ် id ဆိုတာ ရယ် name ရယ် email ရယ်ဖြစ်ပါတယ် name နဲ့ email က TEXT ကိုသိုလှောင်ထားတဲ့ အတွက် String တွေ ဖြစ်ပြီး id ကတော့ INTEGER ဖြစ်ပါတယ် PRIMARY KEY ဆိုတဲ့ အတွက် အလိုလျောက် Data ထည့် တာနဲ့ ဘဲ့ဘိုးသွားမှာဖြစ်ပါတယ်။ဒါတွေပြီးရင်တော့ Table တည်ဆောက်ပြီးကြောင်း Message ပေါ်လာပါမယ်။ ဒါ ဆိုရင် constructor တည်ဆောက်ပြီး ပထမဆုံးလုပ်ရမယ့် database ချိတ်ဆက်တာ table တည်ဆောက်တာ တို့ လုပ်လို့ပြီးပါပြီ နောက်ထပ် Function တစ်ခုတည်ဆောက်ပါမယ်။အဲဒီ Function ကတော့ ကျွန်တော်တို့ data တွေရိုက်ထည့်တဲ့အပိုင်းမှာတာဝန်ယူမယ့် insert_data ဘဲဖြစ်ပါတယ် သူကတော့ ကျွန်တော်တို့ ထည့် ချင်တဲ့ data 2 မျိုးဖြစ်တဲ့ name နဲ့ email ကို parameter နှစ်ခုအဖြစ်ထည့်ထားပါတယ်။ပြီးတော့ variable နှစ်ခုတည်ဆောက်ပြီး parameter က နေဂင်လာမယ့် တန်ဖိုးတွေကိုသိုလှောင်ထားပါတယ် self.name နဲ့ self.email က variable တွေပါ class တွေထဲမှာတည်ဆောက်တဲ့ variable တွေ function တွေကိုခေါ်သုံး မှု့ တွေမှာ self က မဖြစ်မနေထည့်ရပါတယ်။ဒါကိုသတိပြုပေးပါ။connection object ကိုခေါ်ပါတယ် ပြီးတော့ ကျွန်တော်တို့ ရိုက်ထည့်လိုက်တဲ့ name နဲ့ email ကို database columns တွေထဲကို execute() နဲ့ ထည့် ပါတယ်။

```
self.cursor.execute('INSERT INTO emaildb(name,email) VALUES
(?,?)',(self.name,self.email))
```

ဒီမှာပိုပါလာတာကတော့ con.commit() ဖြစ်ပါတယ် အဲဒါက ဘာလုပ်တာလဲဆိုတော့ database ထဲကို ရိုက် ထည့်ပြီးတဲ့ data တွေကို save လုပ်ဖို့အတွက်သုံးတာပါ Notepad ထဲမှာရေးထားတဲ့စာတွေကိုတောင် save လုပ်မှမြင်ရတာဖြစ်ပါတယ် ဒီတော့ အခုလဲ save လုပ်ရပါတယ် con.commit() နဲ့လုပ်ပါတယ်။ပြီးရင် database ထဲ data ထည့်တာအောင်မြင်ကြောင်း print ထုတ်မယ်။နောက်တစ်ခုလုပ်ထားတာကတော့ class နောက်တစ် ခု တည်ဆောက်ပြီး အပေါ်က database class ကို inheritance လုပ်ထားပါတယ်။ပြီးတော့ အဲဒီထဲမှာ function

တစ်ခုတည်ဆောက်ပြီး အဲဒီ function ရဲ့အလုပ်ကတော့ database ထဲက data တွေကို အဖြေအဖြစ်ပြန်ပြဖို့ အတွက် variable တစ်လုံးတည်ဆောက်ပြီး database ထဲက ရှိသမျှ data အားလုံးကို SELECT နဲ့ * ကိုအသုံး ပြုပြီးအားလုံးကို variable ထဲကို ထည့်လိုက်ပါတယ်။နောက်ထပ်လုပ်ထားတာကတော့ ကျွန်တော်တို့ ဖော်ပြချင်တဲ့စာတွေကိုဖော်ပြထားတဲ့လုပ်ဆောင်ချကဘဲဖြစ်ပါတယ်။နောက်ပြီးတော့ အဖြေ ထုတ်ဖို့အတွက် for loop တစ်ခုကိုတည်ဆောက်ပြီး x ထဲကိုထည့်ပြီး Loop လုပ်ပါတယ် ဒီလို Loop လုပ်လိုက်တဲ့ x ဟာ Tuple ပုံစံမျိုးနဲ့ database ထဲက data တွေကိုရလာပါတယ် ဒါကြောင့် index number သုံးပြီး အဖြေထုတ်နိုင်ပါတယ်။

```
print(x[0],"\t",x[1],"\t",x[2])
```

columns သုံးခုအတွက် x[0] ဆိုတာ id အတွက် ဖြစ်ပြီးတော့ x[1] ကတော့ name အတွက်ဖြစ်ပါတယ် နောက် တစ်ခုဖြစ်တဲ့ x[2] ကတော့ email အတွက်ဖြစ်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ပြင်ဆင်စရာရှိတာ ပြင်ဆင်ပြီး ပါပြီး၊ဒါဆိုရင် Data ထည့်ဖို့ Database ထဲကိုထည့်ထားတဲ့ info တွေကြည့်ဖို့ ကို Program ရေးလို့ ရပါပြီ။ အရင်လိုဘဲ while loop သုံးပြီးရေးကြတာပေါ့။

variable တစ်လုံးတည်ဆောက်လိုက်ပါတယ် ပြီးတဲ့ အခါမှာ class ဖြစ်တဲ့ sen() ကို object အဖြစ်ပြောင်း ပြီးခေါ်လိုက်ပါတယ်။တည်ဆောက်ခဲ့တဲ့ variable exit =”” ထဲမှာ ဘာတန်ဖိုးမှမထည့်ပါဘူး ဒါက while loop ကို control လုပ်ဖို့အတွက်ဘဲဖြစ်ပါတယ်။while loop တစ်ခုကိုတည်ဆောက်လိုက်ပါတယ် အဲဒီ while မှာ exit က 100 နဲ့မညီမခြင်း != နဲ့ဆက်အလုပ်လုပ်စေပါတယ် မညီသေးဘူးဆိုတဲ့ အခြေနေမှန်နေရင် ဆက်အလုပ် လုပ်သွားမှာဖြစ်ပါတယ်။လုပ်မယ့်အလုပ်တွေကိုကြည့်ရအောင် print နဲ့ ဘာညာသရကာ ဆိုတာတွေကို အဖြေ ထုတ်ထားတယ် ပြီးတော့ User Input တစ်ခုလက်ခံထားပါတယ်။ပြီးတော့ if နဲ့ condition စစ်ဆေးပါ တယ် ကျွန်တော်တို့က User Input မှာ 1 ရိုက်ထည့်ရင် Data Insert လုပ်မယ် user input နှစ်ခုတည်ဆောက် တယ် name နဲ့ email အတွက်ပေါ့ဗျာ။

```
obj_2.insert_data(input_name, input_email)
```

class အတွက်တည်ဆောက်ထားတဲ့ object ဖြစ်တဲ့ obj_2 ကိုခေါ်ပြီးရင် insert_data ဆိုတဲ့ function ကို ခေါ်သုံးပါတယ် class 2 object ထဲမှာ insert_data ဆိုတဲ့ function ပါနေတာ inheritance လုပ်ထားလို့ ပါ။ထားပါ insert_data function မှာ parameter ၂ လုံးထည့်ရပါမယ် အဲဒီ နှစ်လုံးကို ကျွန်တော်တို့ user input ကနေ ရယူလိုက်ပါတယ်။ပြီးရင် ဘာအတွက်မှမဟုတ်ဘဲနဲ့ input(“Press any key to continue….”) ဆို တဲ့အလုပ်ကိုလုပ်ထားပါတယ် ဒါက ဘာအတွက်လဲဆိုတော့ ချက်ခြင်းကြီးတခြားအလုပ်ကိုပြောင်းသွားမှာမလို လားလို့ဖြစ်ပါတယ် တစ်ခုခုကိုနှိပ်လိုက်မှ နောက်ထပ် အလုပ်ကိုလုပ်ပါတယ်။ကျွန်တော်တို့ 2 ကိုရိုက်ထည့် လိုက်မယ်ဆိုရင်တော့ ကျွန်တော်တို့ရေးပြီးမှတ်ထားတဲ့ နာမည်နဲ့ email တွေကိုတွေ့ရမှာဖြစ်ပါတယ်။3 ကိုထည့်ရေးတယ်ဆိုရင်တော့ exit ကို 100 နဲ့ညီပေးလိုက်တာဖြစ်လို့ exit != 100 အခြေနေက မှားသွားတာ ဖြစ်လို့ While က ထွက်သွားပြီး Program ရပ်သွားပါတယ်။

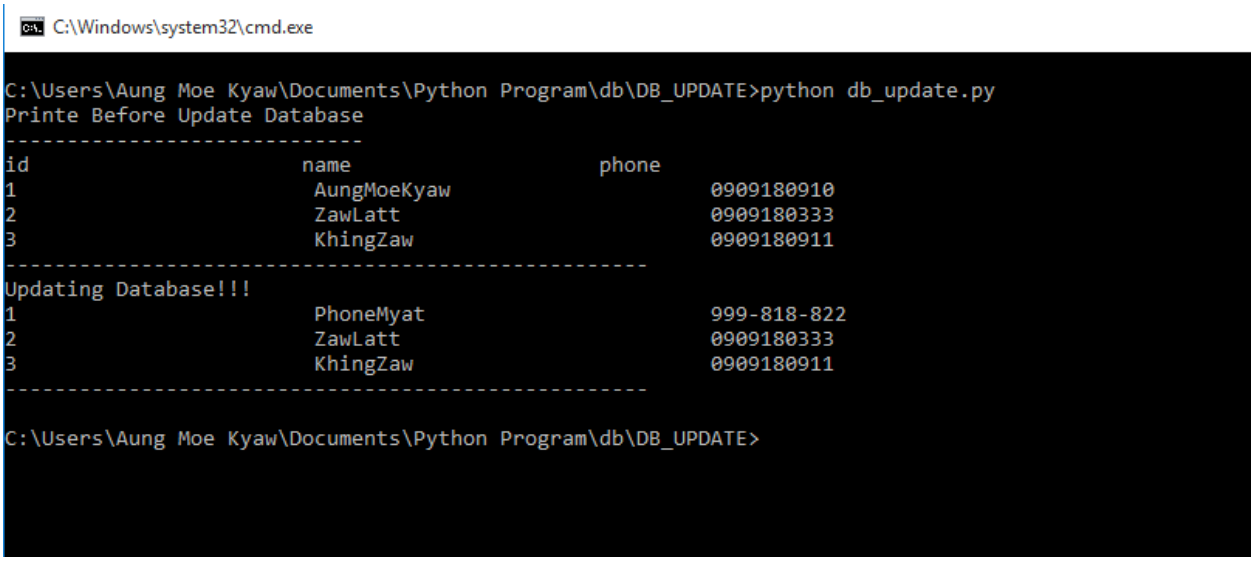
```
python Database_example2.py

C:\Users\Aung Moe Kyaw\Documents\Python Program\db\TEST_TWO>python Database_example2.py
Database is Connected Successfully

                    ------------------------------
                    -Table was Created Successfully-
                    ------------------------------



Please Enter The Choice Number-
                      1.insert Data
                      2.Show Data
                      3.Exit

Enter The Number:2
Some Thing Show YOU
------------------
No.      Name.                  Email
1        AungMoeKyaw     aungmoekyaw1995@gmail.com
2        phonemyatmin    phonemyatmin29@gmail.com
3        zinlatt         zinlatt@gmail.com
4        HtikeSanWin     htikesanwin2001@gmail.com
Press Any Key to Continues.....
```

ပုံပါအတိုင်းပါဘဲဗျာ ကျွန်တော်က insert Data မှာ ကျွန်တော်က email နဲ့နာမည် 4 ခုကိုထည့်ထားပြီး ဘယ်အချိန်ဖွင့်ကြည့်လိုက်ကြည့်လိုက် ကျွန်တော်တို့မှတ်သားထားတာကိုမြင်တွေ့ရမှာဖြစ်ပါတယ်။နောက်ထပ်တစ်ခုကို UPDATE အကြောင်းကိုရေးမှာဖြစ်ပါတယ် SQL ရဲ့ UPDATE ဖြစ်ပါတယ်။UPDATE ကတော့ သိတဲ့အတိုင်းပါဘဲ Database ထဲမှာရှိနေတဲ့ Data တွေကို update လုပ်ဖို့အသုံးပြုတာဖြစ်ပါတယ်။

```
import sqlite3

conn = sqlite3.connect('simple.db')
cursor = conn.cursor()
cursor.execute('CREATE TABLE IF NOT EXISTS storage(id INTEGER PRIMARY KEY,name
TEXT,phone TEXT)')

cursor.execute('INSERT INTO storage(name,phone)
VALUES(?,?)',("AungMoeKyaw","0909180910"))
cursor.execute('INSERT INTO storage(name,phone) VALUES(?,?)',("ZawLatt","0909180333"))
cursor.execute('INSERT INTO storage(name,phone)
VALUES(?,?)',("KhingZaw","0909180911"))


print("Printe Before Update Database")
print("-----------------------------")
print("id\t\t\tname\t\t\tphone")
database_list = cursor.execute('SELECT * FROM storage')
for varx in database_list:
    print(varx[0],"\t\t\t",varx[1],"\t\t\t",varx[2])
print("-----------------------------------------------------")
print("Updating Database!!!")

cursor.execute('UPDATE storage SET name ="PhoneMyat",phone="999-818-822" WHERE id =
1')
database_list1 = cursor.execute('SELECT * FROM storage')
for varz in database_list1:
```

```
    print(varz[0],"\t\t\t",varz[1],"\t\t\t",varz[2])
print("-----------------------------------------------------")
```

ဒီမှာ ပုံမှန်လုပ်ရိုးလုပ်စဉ်အတိုင်း sqlite3 ကို import လုပ်တယ် Table တည်ဆောက်တယ် ပြီးတော့ data တွေ ထည့်ပါတယ် Data သုံးခုထည့်ပါတယ်။ပြီးတော့ print before updating လို့ စာထုတ်ထားပြီးတော့ ကျွန်တော်တို့ update မလုပ်ခင် data ထည့်ထားတဲ့အတိုင်း အဖြေထုတ်ပါတယ်။ကျွန်တော်တို့ထည့်ထားတဲ့ အတိုင်းဘဲ အဖြေထွက်ပါတယ်။ဒါဆိုရင် Update လုပ်တဲ့အပိုင်းကိုသွားရအောင် Updating Database ဆိုတဲ့စာ ကို အရင်ဆုံးထုတ်တယ် ပြီးတော့ update လုပ်ငန်းစဉ်လုပ်ပါတော့မယ် cursor object ကို အရင်ဆုံး ချိတ်တယ် ပြီးတော့ execute() နဲ့လုပ်တယ် ပြီးတော့ SQL Statement ကိုအသုံးပြုပြီးလုပ်မယ် UPDATE storage SET ဆိုတာက UPDATE လုပ်ငန်းစဉ်ပါ ဘာကို UPDATE လုပ်မယ်ဆိုတာကိုတော့ SET နောက်မှာရေးရပါတယ် name နဲ့phone ကို ကျွန်တော်တို့ update လုပ်ဖို့ data ထည့်မယ် ပြီး ရင် WHERE ဆိုတာက Conditionဖြစ်ပါတယ် ဘယ်နေရာကိုလုပ်မယ် ဘယ်နေရာက Data ကို update လုပ်မယ်ဆိုတာမျိုးပေါ့။ဒီမှာ တော့ WHERE id =1 ဆိုပြီးလုပ်ထားတဲ့အတွက် id = 1 နေရာက name နဲ့ phone ကို update လုပ်မယ်လို့ပြောလိုက်တာပါ။နောက်ပြီး update လုပ်ပြီး အဖြေပြန်ထုတ်လိုက်တဲ့ အခါမှာ id = 1 နေရာမှာရှိတဲ့ name နဲ့ phone က ကျွန်တော်တို့ UPDATE လုပ်ထားတဲ့ Data တွေကိုတွေ့ရမှာဖြစ်ပါတယ်။



ဒါဆိုရင် အထက်ပါ ပုံအတိုင်းတွေ့မြင်ရမှာဖြစ်ပါတယ်။

Vault ဆော့ဝဲလ် လိုပထမဆုံး ကျွန်တော်တို့ constant ထားတဲ့ original password ကိုရိုက်ထည့်ပြီးမှ ကျွန်တော်တို့ စိတ် ကြိုက် password ပြောင်းနိုင်တဲ့ လုပ်ဆောင်ချက်ကိုလုပ်လို့ရနိုင်ပါတယ်။အခုလည်း ဒီလိုလုပ်မယ် ပထမဆုံး ကျွန်တော်တို့ database ဆောက်ပြီး original username နဲ့ password တည်ဆောက်မှာပါ ပြီးမှ ကျွန်တော် တို့ပြန်ပြောင်းချင်ရင် Change ကိုရွေးပြီးပြောင်းလို့ရတဲ့ ပုံစံမျိုးပေါ့။ဒါဆိုရင် ရေးကြည့်ရအောင်။

```
import sqlite3
```

```
class simple_login:

    def __init__(self):
        self.con = sqlite3.connect('simpledb.db')
        self.db_con = self.con.cursor()
        self.db_con.execute('CREATE TABLE IF NOT EXISTS simpledb(name TEXT , password
TEXT)')
        var = self.db_con.execute('SELECT * FROM simpledb')
        for iz  in var:
            name = iz[0]
            passw = iz[1]
        try:
            if name != None and passw != None:
                print("Hello Default Username and Password are admin!!")
            else:
                pass
        except UnboundLocalError:
            self.db_con.execute('INSERT INTO simpledb(name,password) VALUES(?,?)',
("admin", "admin"))
            self.con.commit()
            print("ehhhh")

        else:
            pass


    def change(self,name,password):
        self.name = name
        self.password = password
        self.db_con.execute('UPDATE simpledb set
name=?,password=?',(self.name,self.password))
        self.con.commit()

    def login(self,in_name,in_password):
        self.in_name = in_name
        self.in_password = in_password
        database_data = self.db_con.execute('SELECT * FROM simpledb')
        for db_info in database_data:
            name = db_info[0]
            password = db_info[1]

        if self.in_name == name and self.in_password == password:
            print("Login Successfully")
        else:
            print("Login Fail")


login_object = simple_login()

exit =""

while exit != "exit":
    print("""Please Choice Number:
             1.Login
             2.Change Name and Password
             3.Exit Program""")
    ch_num = input("Enter The Number:")

    if ch_num == "1":
        n = input("Enter Username:")
        p = input("Enter Password:")
        login_object.login(n,p)
```

```
    elif ch_num == "2":
        change_n = input("Enter Change Username:")
        change_p = input("Enter Change Password:")
        login_object.change(change_n,change_p)

    elif ch_num == "3":
        exit = "exit"
        print("GoodBye")

    else:
        print("Please Choice 1,2 or 3,Thank!")
```

ပုံမှန်နည်းအတိုင်းပါဘဲ class တစ်ခုတည်ဆောက်ပြီး constructor ထဲမှာ Database ချိတ်တယ် Table တည် ဆောက်တယ် ဒီမှာပိုပါတာကတော့ အရင်ဆုံး ဘာ data မှမထည့်သေးဘဲ SELECT ကိုသုံးပြီး data ယူထားပါတယ်။

```
var = self.db_con.execute('SELECT * FROM simpledb')
```

var ထဲကို Database ထဲက data တွေယူထားတာတွေ့ရမှာပါ။ဒါက ဘာကြောင့်လဲဆိုတော့ ကျွန်တော်တို့က အရင်ဆုံး default username နဲ့ password ကိုထည့်ချင်တာဖြစ်ပါတယ် အဲလိုထည့်တဲ့အခါမှာ ဒီအတိုင်း insert data နဲ့ရိုးရိုးထည့်ရင် program အစကပြန် run တိုင်း default program ဘဲဖြစ်နေမှာပါ ဒါကြောင့် အရင်ဆုံး database ထဲမှာ data ရှိမရှိစစ်ဆေးတဲ့အလုပ်ကိုလုပ်မှာဖြစ်ပါတယ်။

```
for iz  in var:
    name = iz[0]
    passw = iz[1]
```

for loop ပတ်ပြီး var ထဲက data ဖြစ်တဲ့ index 0 နဲ့ 1 ကို name နဲ့ passw ထဲကိုထည့်ပါတယ်။ ပြီးတော့ if နဲ့ဒီအတိုင်းစစ်လိုက်ရင် သေချာပေါက် ဘာ data မှမထည့်ရသေးတဲ့အတွက် data မရှိတဲ့ error တတ်မှာ ဖြစ်ပါတယ်။အဲဒါက

```
UnboundLocalError
```

ဖြစ်ပါတယ်။ဒါကြောင့် Error တတ်မှ လုပ်မယ့် အလုပ်ကိုထည့်ဖို့အတွက် try: except: else: ကို အသုံး ပြုပါမယ်။

```
try:
    if name != None and passw != None:
        print("Hello Default Username and Password are admin!!")
    else:
        pass
except UnboundLocalError:
    self.db_con.execute('INSERT INTO simpledb(name,password) VALUES(?,?)', ("admin",
"admin"))
    self.con.commit()
    print("ehhhh")

else:
    pass
```

try ထဲမှာ name နဲ့ passw က None နဲ့မညီဘူးဆိုရင်(သဘောက data တစ်ခုခုရှိနေရင် ဗလာဖြစ်မနေဘူးဆိုရင်) print နဲ့ Hello Default Username and Password are admin!! ဆိုတာပြမယ် မဟုတ်ရင် pass ဆိုတာ ဘာအလုပ်မှမလုပ်ဘဲကျော်သွားမယ်လို့ပြောတာပါ။သေချာပေါက် ပထမကြိမ် Run တာဖြစ်တဲ့ အခါမှာ

name နဲ့ passw က ဗလာဖြစ်နေမှာပါ ဘာ data မှမပါပါဘူး ဒါကြောင် Error ဖြစ်မယ် ဒီ Error ဖြစ်ရင် လုပ်မယ့် အလုပ်က except ထဲမှာ

```
self.db_con.execute('INSERT INTO simpledb(name,password) VALUES(?,?)', ("admin",
"admin"))
self.con.commit()
```

ဖြစ်ပါတယ် admin,admin ဆိုပြီး name ရော password ရောထည့်မယ်ပြီးရင် self.con.commit() နဲ့ save မယ်လို့ပြောလိုက်တာပါ။ပြီးတော့ ehhhh ဆိုတဲ့ Message ထွက်လာမယ်ပေါ့ else ထဲမှာတော့ pass နဲ့ ကျော်လိုက်ပါတယ်။ဒီတော့ပထမကြိမ် Run ရင် name နဲ့ passw ထဲမှာ ဘာမှမရှိလို့ except ထဲက အလုပ် လုပ်ပြီး admin admin ဖြစ်သွားပါတယ် နောက်တစ်ကြိမ် Run ရင်တော့ name ရော passw ရောက None ဖြစ်မနေတော့တဲ့ အတွက် print ထုတ်ထားတဲ့ Hello စာသားထွက်လာမှာဖြစ်ပါတယ်။

ဒါဆိုရင်နောက် Function တစ်ခုတည်ဆောက်ပါတယ် ဒါက ဘာအတွက်လဲဆိုတော့ Password နဲ့ user name ကို ကိုယ့်ကြိုက်တဲ့ဟာပြောင်းချိန်းဖို့အတွက်ဖြစ်ပါတယ်။

```
def change(self,name,password):
    self.name = name
    self.password = password
    self.db_con.execute('UPDATE simpledb set
name=?,password=?',(self.name,self.password))
    self.con.commit()
```

ဒါက မထူးဆန်းဘူး variable ၂ ခုတည်ဆောက်ပြီးတော့ parameter ၂ လုံးကလာတဲ့ data ကို လက်ခံ တယ်ပြီးတော့ UPDATE ကိုသုံးပြီးတော့ Username နဲ့ Password ကို UPDATE လုပ်လိုက်တယ် ပေါ့ ဒါဆိုရင် ကျွန်တော်တို့ default admin,admin ကနေ ကျွန်တော်တို့ပြောင်းလိုက်တဲ့ ဟာတွေဖြစ်သွားပါတယ်။

ဒါဆိုရင်နောက် Function တစ်ခုတည်ဆောက်ရမှာပေါ့ Login လုပ်မယ့် Login အောင်မြင်လား မအောင်မြင်ဘူး လာပြမယ့် အမျိုးအစားပေါ့။

```
def login(self,in_name,in_password):
    self.in_name = in_name
    self.in_password = in_password
    database_data = self.db_con.execute('SELECT * FROM simpledb')
    for db_info in database_data:
        name = db_info[0]
        password = db_info[1]

    if self.in_name == name and self.in_password == password:
        print("Login Successfully")
    else:
        print("Login Fail")
```

ဒီ Function မှာ parameter ၂ ခုပါတယ် variable ၂ ခုထဲကိုထည့်တယ် ပြီးတော့ database_data ဆိုတဲ့ database ထဲက name နဲ့ password ကို တိုက်စစ်ဆေးပေးမယ့် နှစ်ခုကို if နဲ့ ကျွန်တော်တို့ရဲ့ parameter ၂ခုထဲကလာတဲ့ Data တွေကိုတိုက်စစ်ပါတယ်နှစ်ခုလုံးမှန်မယ်ဆိုရင် Login Successfully ဖြစ်မယ် မဟုတ်ရင် Fail ပြမယ်ပေါ့။

ပြီးရင်တော့သိတဲ့အတိုင်းဘဲ class ကို object အဖြစ်ပြောင်းခေါ်တယ်။ပြီးရင် ကျွန်တော်တို့ while loop နဲ့ ထပ်ခါထပ်ခါပြနေမယ် program တည်ဆောက်တယ်။1 ကိုရေးရင် Login အတွက် login function ကိုခေါ်မယ် user name နဲ့ password အတွက် user input နှစ်ခုလက်ခံမယ် database ထဲမှာရှိတဲ့ username နဲ့ password နဲ့ ရိုက်ထည့်လိုက်တဲ့ user input က username နဲ့ password နဲ့တူလားစစ်ဆေးတယ် တူရင် Login Successfully ဖြစ်မယ် မတူရင် Login Fail။2 ကိုရိုက်ရင် change function ကိုခေါ်မယ် user input ၂ ခု လက်ခံမယ်ပြီးရင် ရိုက်ထည့်လိုက်တဲ့ user input နှစ်ခုကို admin admin ဆိုတဲ့ default user name နဲ့ password နဲ့နေရာမှာထည့်ပြီး save ပါမယ်။3 ကိုရိုက်ထည့်လိုက်ရင်တော့ program က != ဖြစ်စဉ်များယွင်းသွားပြီး ပိတ်သွားမှာဖြစ်ပါတယ်။ဒါဆိုရင် Delete ကိုသွားရအောင် DELETE ကတော့ ကျွန်တော်တို့ Data တွေကို ဖျက်ဖို့အသုံးပြုတာဖြစ်ပါတယ်။DELETE ကို SQL Statement အနေနဲ့ဘယ်လိုရေးလဲဆိုရင်။

DELETE FROM database WHERE Condition

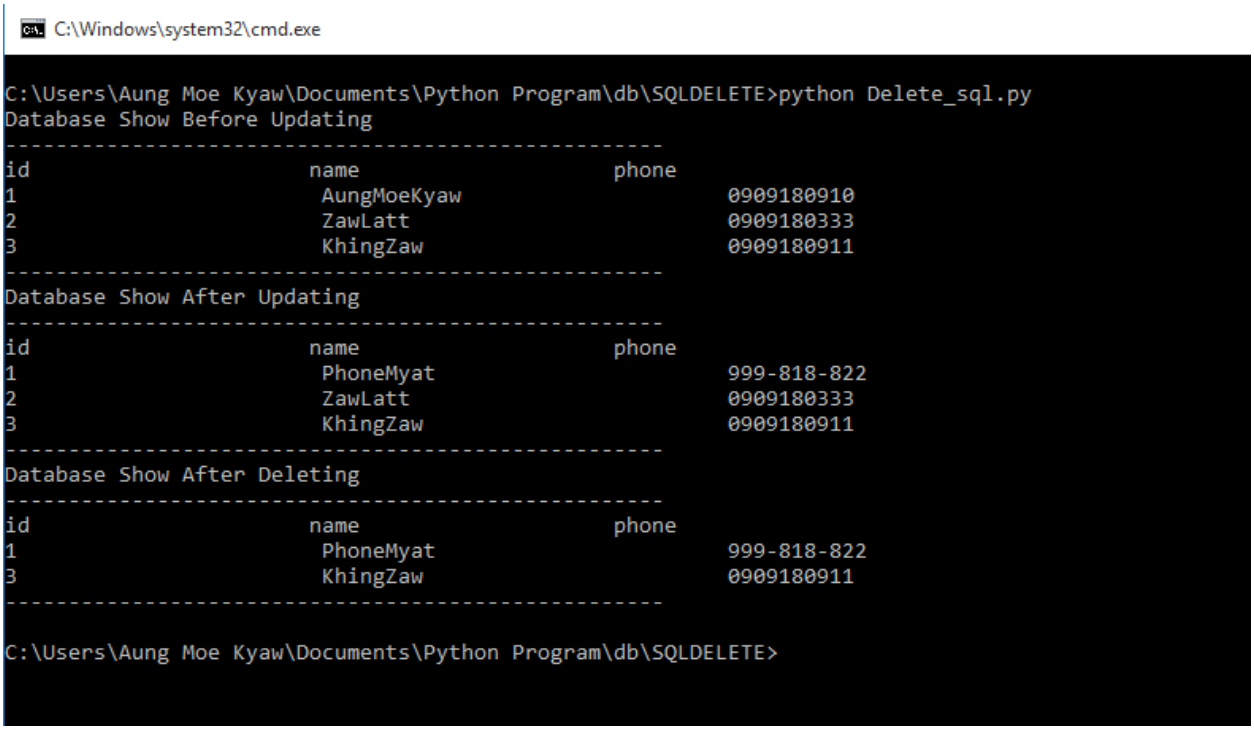
Database နေရာမှာကျွန်တော်တို့ တည်ဆောက်ထားတဲ့ database နာမည်ကိုထည့်မယ် WHERE ကတော့ စစ်ဆေးတဲ့ အခြေနေတွေတွက်ထုတ်ပေးတာဖြစ်ပါတယ် WHERE id = 1 ဆိုရင် id 1 ကိုစစ်ရွေးချယ်တယ်လို့ ပြောတယ် WHERE နောက်မှာ name နဲ့လည်းစစ်နိုင်သလို တခြားသော columns တွေ ကိုအသုံးပြုပြီးလဲစစ်နိုင်ပါတယ်။ဒါဆိုရင်ကျွန်တော်တို့ စမ်းရေးကြည့်ရအောင်။

```
import sqlite3

def print_def(print_text):
    print(print_text)
    print("-------------------------------------------------------")
    print("id\t\t\tname\t\t\tphone")
    database_list = cursor.execute('SELECT * FROM storage')
    for varx in database_list:
        print(varx[0], "\t\t\t", varx[1], "\t\t\t", varx[2])
    print("-------------------------------------------------------")

conn = sqlite3.connect('simple.db')
cursor = conn.cursor()
cursor.execute('CREATE TABLE IF NOT EXISTS storage(id INTEGER PRIMARY KEY,name
TEXT,phone TEXT)')

cursor.execute('INSERT INTO storage(name,phone)
VALUES(?,?)',("AungMoeKyaw","0909180910"))
cursor.execute('INSERT INTO storage(name,phone) VALUES(?,?)',("ZawLatt","0909180333"))
cursor.execute('INSERT INTO storage(name,phone)
VALUES(?,?)',("KhingZaw","0909180911"))


print_def("Database Show Before Updating")

cursor.execute('UPDATE storage set name ="PhoneMyat",phone="999-818-822" WHERE id =
1')

print_def("Database Show After Updating")

cursor.execute('DELETE FROM storage WHERE id = 2')

print_def("Database Show After Deleting")
```

လုပ်နေကြအတိုင်း sqlite3 ကို import လုပ်ပြီးတော့ function တစ်ခုကိုတည်ဆောက်လိုက်ပါတယ် ဒီ function ကတော့ အဖြေထုတ်ဖို့အတွက်သုံးတဲ့ function ဖြစ်ပါတယ်။အဲဒီထဲမှာ SELECT * FROM database ထည့်ပြီး for loop နဲ့လုပ်ထားပါတယ်။ကျွန်တော်တို့က data insert လုပ်ပြီးတော့ တစ်ကြိမ် data update လုပ်ပြီးတော့ တစ်ကြိမ် data delete လုပ်ပြီးတော့ တစ်ကြိမ် သုံးကြိမ်ရေးရမှာပါ ဒီလိုထပ်ခါထပ်ခါရေးရမှာ စိုးတဲ့အတွက် function တစ်ခုတည်ဆောက်ပြီး parameter တလုံးထည့်ထားတယ် အဲဒီ Function ကိုခေါ်ပြီးအဖြေထုတ်တဲ့လုပ်ငန်းစဉ်လုပ်နိုင်ပါတယ်။ဒီမှာတော့

```
cursor.execute('DELETE FROM storage WHERE id = 2')
```

ဒီ command နဲ့ id =2 မှာရှိတဲ့ data တွေကို DELETE လုပ်လိုက်တာဖြစ်ပါတယ်။



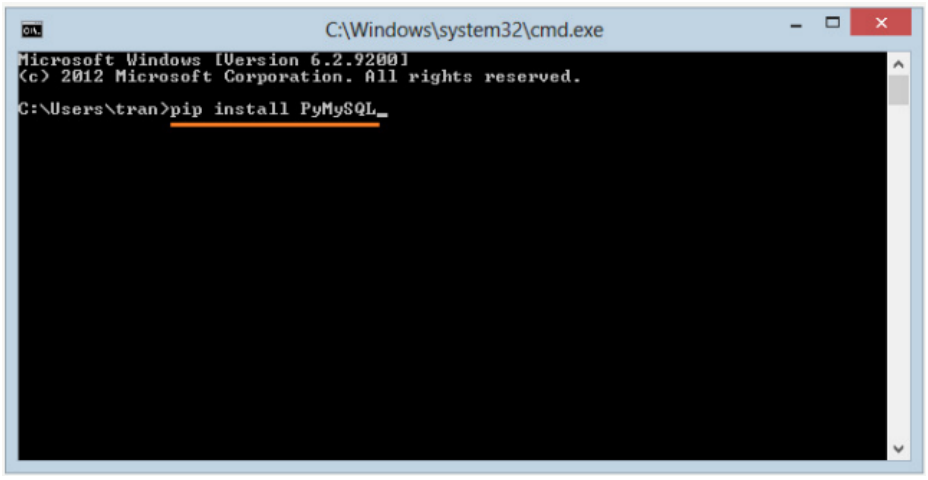
id = 2 ကိုဖျက်လိုက်တဲ့အတွက် id = 2 ပါမလာတော့ပါဘူး။ဒါဆိုရင် ကျွန်တော်တို့ Database ရဲ့ အခြေခံ ဖြစ်တဲ့ INSERT UPDATE DELETE SELECT ကိုလုပ်ဆောင်ပြီးပါပြီ။ဒါဆိုရင် MySQL ဆိုတဲ့ Database အမျိုးအစားနဲ့ချိတ်ဆက်အသုံးပြုနည်းဆက်ရေးသွားမှာဖြစ်ပါတယ်။

MySQL with Python

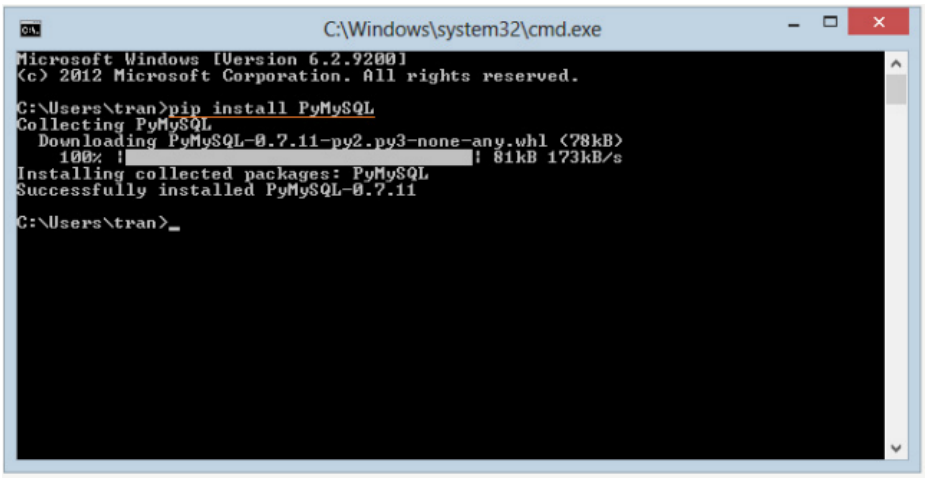
MySQL database ထဲကို data တွေသိမ်းဆည်းဖို့အတွက် ကိုပြင်ဆင်မှုတွေအရင်ဆုံးလုပ်ရပါမယ်။ ဘာကြောင့်လဲဆိုတော့ MySQL အတွက်က sqlite3 လိုမျိုး serverless တွေ မှီခိုမှုကင်းတာတဲ့ စွမ်းဆောင် မှုမျိုးမဟုတ်ပါဘူး။သူက ပိုပြီးများပြားတဲ့ data တွေနဲ့ ပိုပြီးလုံခြုံရေးကောင်းတဲ့စနစ်ကိုပိုင်ဆိုင်ထားသလို

အွန်လိုင်း စနစ်လို့ခေါ်တဲ့ network ကိုအသုံးပြုပြီးလဲ ချိတ်ဆက်နိုင်ပါတယ်။ဒါဆိုရင်ဘာတွေလိုအပ် တာလဲ ကြည့်ရအောင်။Xampp သို့မဟုတ် Wamp Server တစ်ခုသုံးရပါမယ်။ဒါကတော့ web developer တွေ web site နဲ့ပတ်သတ်တဲ့ ရင်းနှီးသူတွေကတော့ သိပြီးသားဖြစ်မှာပါ။ကျွန်တော်ကတော့ xampp ကို အသုံးပြု ပြီး တည်ဆောက်သွားမှာဖြစ်ပါတယ်။သူက local မှာတင် Mysql database နဲ့ PHP myadmin ကိုအသုံးပြုနိုင်ပါ တယ်။website တစ်ခုကို စက်ထဲမှာတင်တည်ဆောက်ကြည့်လို့ရတယ်ပေါ့ဗျာ။xampp ကို DVD ထဲက softwares ထဲမှာပါပါတယ်။install လုပ်လိုက်ပါ။ပြီးရင် MySQL Connector သို့မဟုတ် MySQL နဲ့ချိတ်ဆက် နိုင်တဲ့ python connector module တစ်ခုခုကို install လုပ်ရပါမယ်။ကျွန်တော်က ဒီစာအုပ်ရေးနေတုန်းမှာ Python 3.6 ထွက်နေပြီဖြစ်လို့ version အသစ်ကို အသုံးပြုထားတာဖြစ်တဲ့အတွက် MySQL Connector ကို အသုံးပြုလို့မရပါဘူး Mysql Connector module ဟာ Python 2.7 နဲ့ Python 3.4 အတွက်ဘဲ Support လုပ် ပါတယ်။ဒါပေမယ့် ကံကောင်းထောက်မစွာနဲ့ pyMySQL module ကို install လုပ်ရင် အဆင်ပြေပါတယ်။မ ဟုတ်ရင်တော့ ကျွန်တော်တို့ ချိတ်ဆက်လို့ရမှာမဟုတ်ပါဘူး။ဒါဆိုရင် pyMySQL ကို install လုပ်မှာဖြစ်ပါတယ်

ပထမဆုံး cmd ကိုဖွင့်လိုက်ပါ Windows + R ကိုနှိပ်လိုက်ပါ။



pip ကိုအသုံးပြုပြီး module တွေကို install လုပ်နိုင်ပါတယ်။pip install pyMySQL လို့ရိုက်ပြီး Enter နှိပ်လိုက်ပါ။



အထက်ပါပုံအတိုင်း Download လုပ်ပြီး install လုပ်သွားပါလိမ့်မယ်။ဒါဆိုရင် PyMySQL ကို install လုပ်ပြီးပါပြီ။

What is Pip?

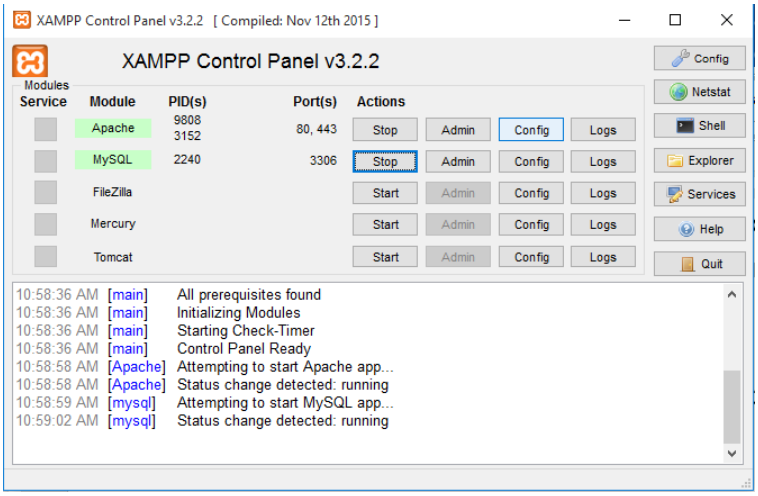
Pip ဆိုတာ package management system တစ်ခုဘဲဖြစ်ပါတယ်။module package တွေကို install လုပ်ဖို့ uninstall လုပ်ဖို့နဲ့ တခြားသော manage လုပ်ဖို့အသုံးပြုတဲ့ စနစ်တစ်ခုဘဲဖြစ်ပါတယ်။ဒီထဲက package တော်တော်များများကို Python Package Index မှာ တွေ့နိုင်ပါတယ်။pip ကိုအသုံးပြုပြီး install လုပ်ခြင်းအားဖြင့် version မကိုက်တာတွေ လိုချက်အပ်မပြည့်စုံလို့သုံးမရတာ စတဲ့ Error တွေမရှိဘဲ စိတ် ချလက်ချအသုံးပြုနိုင်ပါတယ်။pip ကို command-line interface နဲ့အသုံးပြုနိုင်ပါတယ်။

pip install package-name

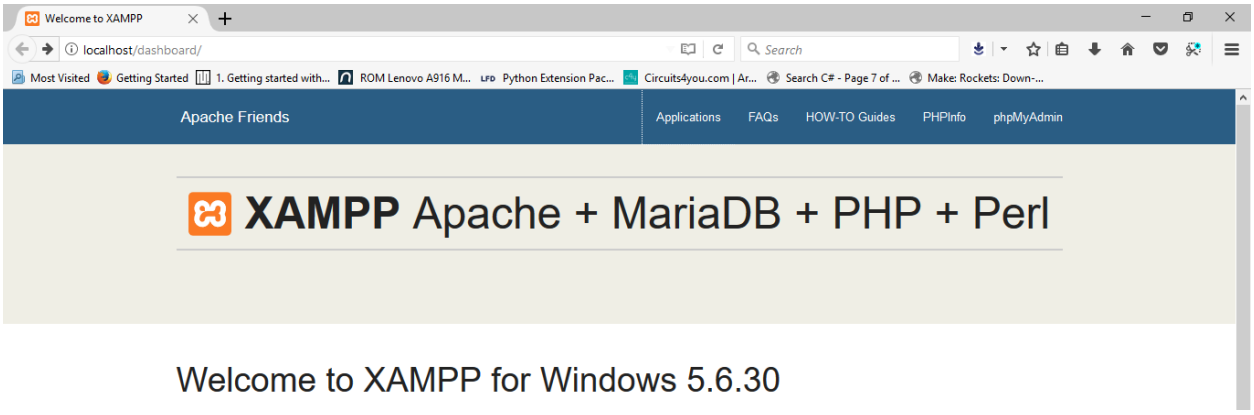
ဒါက package install လုပ်တဲ့ command ဖြစ်ပါတယ်။

pip uninstall package-name

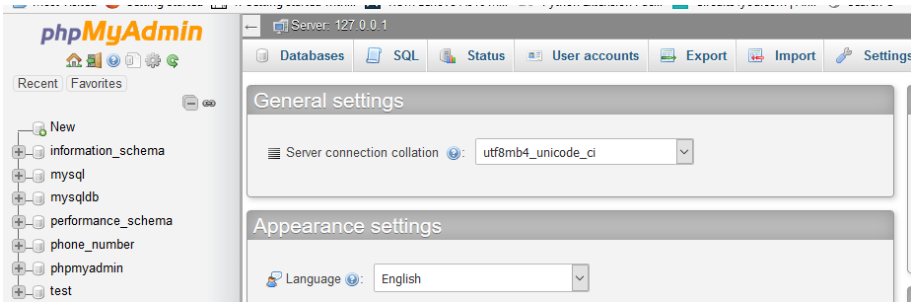
ဒါကတော့ package uninstall လုပ်တဲ့ command ဖြစ်ပါတယ်။အပေါ်မှာ PyMySQL ကို install လုပ်ပြသလိုဘဲ တခြား package တွေကိုလည်း နာမည်ရိုက်ထည့်ပြီး install လုပ်နိုင်မှာဖြစ်ပါတယ်။ဒါဆိုရင် PyMySQL ရော Xampp ရော install လုပ်ပြီးပြီ ဖြစ်လို့စပြီး program ရေးနိုင်မှာဖြစ်ပါတယ်။အရင်ဆုံး Xampp Control Panel ကိုဖွင့်မှာဖြစ်ပါတယ်။



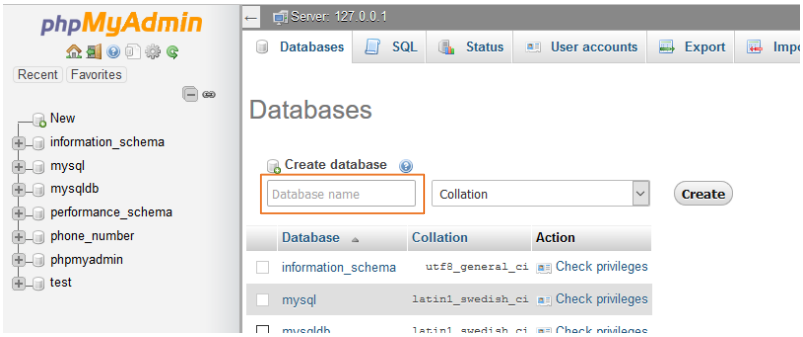
Apache နဲ့ MySQL ကို Start လုပ်ပေးလိုက်ပါ။ပြီးရင် MySQL ရဲ့ဘေးက Admin ဆိုတာကို တစ်ချက်နှိပ်ပေး ပါ Website တစ်ခု Broswer မှာတတ်လာပါလိမ့်မယ်။လိပ်စာကတော့ localhost ဖြစ်ပါတယ်။မတတ်လာဘူး ဆိုရင် ကိုယ်တိုင်ဘဲ Adress bar မှာ localhost လို့ရိုက်လိုက်ပါ။ဒီ xampp က ကိုယ့်စက်ထဲမှာတင် webserver တစ်ခုတည်ဆောက်ပေးလိုက်တာဖြစ်ပါတယ်။



ဒီလို Website တစ်ခုပေါ်လာပါတယ်။ဒီမှာကျွန်တော်တို့က phpMyAdmin ဆိုတာကိုနှိပ်ပေးပါ Database တည်ဆောက်ဖို့အတွက်လိုအပ်တဲ့နေရာကိုရောက်ပါတယ်။

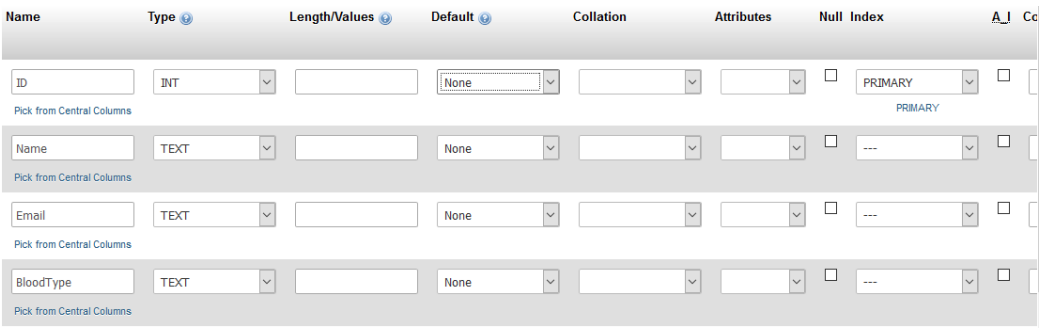


ဒီမှာ Database တည်ဆောက်တဲ့နေရာဖြစ်ပါတယ် New ဆိုတဲ့ ဘယ်ဘက်ဘေးကနေရာလေး မှာ ပါတဲ့ Tab လေးက Database အသစ်ကိုတည်ဆောက်နိုင်စေပါတယ်။

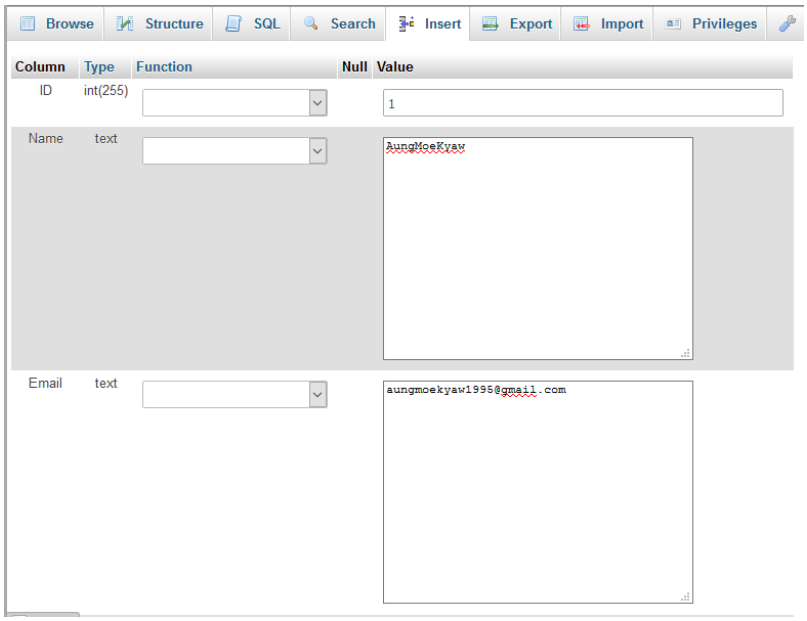


ကျွန်တော် လေးထောက်ကွက်ပြထားတဲ့နေရာမှာ Database နာမည်ပေးပြီး create လုပ်လို့ရပါတယ်။ ကျွန်တော်ကတော့ simpledb လို့ဘဲပေးလိုက်ပါတယ်။

Table Name တစ်ခုပေးပြီး Table ဘယ်နှခုတည်ဆောက်မယ်ဆိုတာပေးလိုက်ပါမယ် Go နှိပ်လိုက်ပါပြီးရင် တော့။ကျွန်တော်ကတော့ Table Name ကို info လို့ပေးထားပြီးတော့ Columns ကိုတော့ မပြောင်းလဲ ဘဲ 4 ဘဲထားတာပါတယ်။



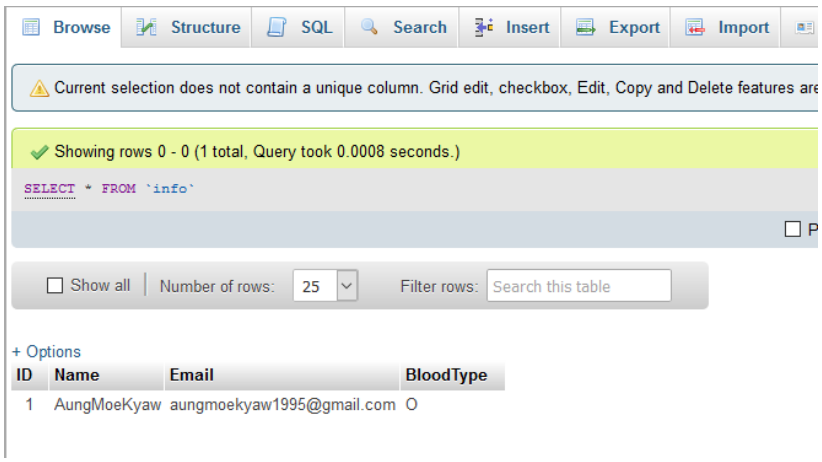
ကျွန်တော်က Columns ၄ ခုမှာ ID, Name , Email , BloodType ဆိုပြီး ၄ ခုကိုတည်ဆောက်ပြီးတော့ ID ကတော့ Type အနေနဲ့ INT ဖြစ်ပြီး IDEX ကို PRIMARY ထားပါတယ်။တခြား ၃ ခုကတော့ String တွေသိုလှောင်ချင်လို့ TEXT ဘဲထားပါတယ်။Length/Values မှာတော့ 255 ဘဲထားပေးပါ။



Tabs ထဲက Insert ကိုရွေးချယ်ပြီးတော့ ကျွန်တော်တို့ Columns တွေနဲ့သက်ဆိုင်တဲ့ Data တွေကို ထည့် ပေးရပါမယ်။ကျွန်တော်ကတော့ ID မှာ 1 Name မှာ AungMoeKyaw Email မှာ aungmoekyaw1995@gmail.com ထည့်ပြီး BloodType မှာ O ထည့်ပါတယ်။ပြီးရင် Go နှိပ်လိုက် ပြီးတော့ ဒီ Data တွေကို Database ထဲကို ထည့်လိုက်ပါတယ်။ဒါက ကျွန်တော်တို့ GUI အနေနဲ့ထည့်တာပေါ့ SQL Statement အနေနဲ့လည်းသူကဘဲဖော်ပြထားပါတယ်။

```
INSERT INTO `info` (`ID`, `Name`, `Email`, `BloodType`) VALUES ('1', 'AungMoeKyaw',
'aungmoekyaw1995@gmail.com', 'O');
```

ဒီလိုဖော်ပြထားတာတွေ့ရမှာပါ sqlite3 ကို statement မှာလည်း ဒီလိုဘဲရေးပါတယ်။



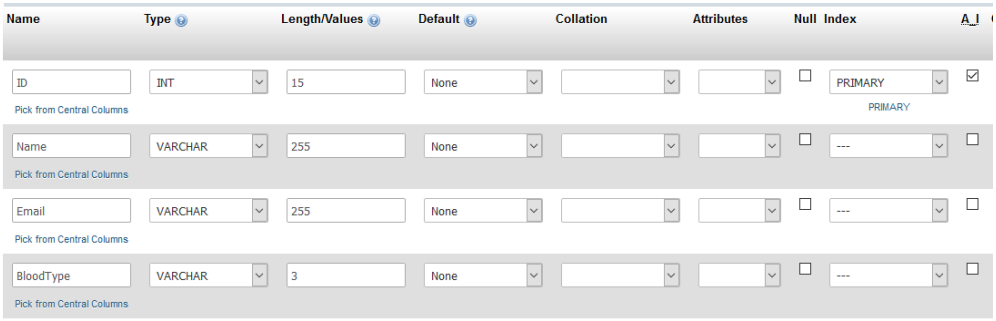
ဒါဆိုရင် Browse Tabs မှာ ကြည့်လိုက်ရင် အထက်ပါပုံအတိုင်း Database ထဲမှာ 1 AungMoeKyaw aungmoekyaw1995@gmail.com O ဆိုတာကိုတွေ့ရမှာပါ။ဒါဆိုရင် ဒီထဲက Data ကိုချိတ်ဆက်ပြီး Python ကနေယူကြည့်ရအောင်။

```
import pymysql.cursors

conn = pymysql.connect(host='localhost', user='root', password='', db='simpledb')
cursor = conn.cursor()

cursor.execute('SELECT * FROM info')

for x in cursor:
    print(x)
```

အရင်ဆုံး pymysql.cursors ဆိုတာကို import လုပ်တယ်ပြီးတဲ့ အခါမှာ database ချိတ်ဆက်တဲ့ connection လုပ်လိုက်ပါတယ်။pymysql.connect() ကိုသုံးပြီး host ဆိုတာ လိပ်စာဖြစ်ပါတယ် ကျွန်တော်တို့ကကိုယ့်စက်ထဲက database ကိုချိတ်မှာမို့ localhost ကိုအသုံးပြုပါတယ် user နေရာမှာ localhost ရဲ့ user က root ဖြစ်ပါတယ် password =''ကတော့ ဘာမှမပါပါဘူး။db ဆိုတာကတော့ ကျွန်တော်တို့တည်ဆောက်ထား တဲ့ db နာမည်ထည့်ရမှာပါ simpledb ဖြစ်ပါတယ်။ဒါဆိုရင် localhost ထဲမှာ ရှိတဲ့ database ကိုချိတ်ဆက် လိုက်ပြီဖြစ်ပါတယ် Online database အသုံးပြုချင်ရင်တော့ Database Service ပေးတဲ့ နေရာတစ်ခုမှာ Database တည်ဆောက်ပြီး လိပ်စာနေရာမှာ အဲဒီ online website လိပ်စာ user နေရာမှာ account user ပြီးတော့ password နေရာမှာတော့ account password ပေးရမှာပေါ့။ဒါပြီးရင် cursor.excute('SELECT * FROM info') ဆိုပြီးတော့ info table ထဲက data တွေကိုရွေးထုတ်လိုက်ပါတယ် ပြီးတော့ for loop ပတ်ပြီး အရင် ကျွန်တော်တို့ sqlite3 လိုဘဲ အဖြေထုတ်လိုက်တဲ့အခါမှာ အဖြေပေါ်လာမှာတွေ့ရမှာပါ။

ဒါဆို Database ရဲ့ အခြေခံလုပ်ဆောင်ချက်တွေဖြစ်တဲ့ UPDATE, DELETE , SELECT, INSERT ဆိုတဲ့ လုပ်ငန်းစဉ်တွေလုပ်ကြည့်ရအောင်။

အရင်ဆုံး ကျွန်တော် တို့ Database တည်ဆောက်မယ် အသစ်တစ်ခုပေါ့။xampp control panel ကိုဖွင့်ပါ ပြီးရင် service ၂ ခုဖြစ်တဲ့ Apache နဲ့ MySQL ကိုဖွင့်ပြီး localhost ထဲကိုဝင်လိုက်ပါ။ပြီးရင် အပေါ်က database တည်ဆောက်တဲ့ စနစ်တွေအတိုင်း အသစ်တည်ဆောက်လိုက်ပါ။ကျွန်တော်ကတော့ simple ဆိုတဲ့ နာမည်နဲ့ဘဲတည်ဆောက်ပါတယ်။Table ကိုတော့ info ဆိုတဲ့ နာမည်နဲ့ table တည်ဆောက်တယ် columns က ၄ ခုပါဝင်ပါတယ်။ID, Name , Email, BloodType တို့ဖြစ်ပါတယ်။



ဒီမှာ ID မှာ ကျွန်တော်တို့က သူ့အလိုလို auto increament အနေနဲ့ row တစ်ခုထည့်ရင် 1 တိုးသွား စေချင် လို့ A_I ဆိုတဲ့နေရာမှာ အမှတ်ခြစ်ပေးပြီးတော့ PRIMARY ပြောင်းထားရပါတယ်။Type အနေနဲ့ကတော့ INT ဘဲဖြစ်ပါတယ်။Name ကတော့ Type ကို VARCHAR ထားပြီး 255 အထိထားတာပါတယ်။တခြားဟာကတော့ ပြဿနာသိပ်မရှိပါဘူး။ရိုးရိုးဘဲတည်ဆောက်ထားပါတယ်။ဒါဆိုရင် Database ကတော့ပြင်ဆင်ပြီးပြီ။Program နဲ့ ချိတ်ဆက်ဖို့ဘဲလိုပါတော့တယ်။

```
import pymysql
import sys



try:
    db = pymysql.connect(
        host = 'localhost',
        user = 'root',
        password = '',
        db = 'simple'

        )
except Exception as e:
    sys.exit('We cant get into the database')

cursor = db.cursor()

def show_info():
    cursor.execute('SELECT * FROM info')
    result = cursor.fetchall()

    for x in result:
        print(x)
```

```
def insert_data():

    user_in = input("Enter Name:")
    email_in = input("Enter Email:")
    blood_t = input("Enter BloodType")
    cursor.execute('INSERT INTO info(Name,Email,BloodType)
VALUES(%s,%s,%s)',(user_in,email_in,blood_t))
    db.commit()

def delete_data():
    user_de = input("Enter Delete Name:")
    cursor.execute('DELETE FROM info WHERE Name = %s',user_de)
    db.commit()

def update():
    id_num = input("Enter Edit ID:")
    user_in = input("Enter Name:")
    email_in = input("Enter Email:")
    blood_t = input("Enter BloodType:")
    cursor.execute('UPDATE info set Name = %s,Email =%s,BloodType=%s WHERE ID =
%s',(user_in,email_in,blood_t,id_num))
    db.commit()

a = ""
while a !="100":
    print("""
         1.Insert Data
         2.SHOW Data
         3.Delete Data
         4.Update Data

         """)
    num = input("Enter Number:")
    if num == "1":
        insert_data()

    elif num == "2":
        show_info()

    elif num == "3":
        delete_data()

    elif num == "4":
        update()
```

အရင်ဆုံး pymysql နဲ့ sys module နှစ်ခုကို import လုပ်ပါတယ်။ပြီးတော့ database ချိတ်ဆက်ဖို့ connection လုပ်တဲ့နေရာကို try: နဲ့ Except လုပ်ထားပါတယ်။ဘာကြောင့်လဲဆိုတော့ အကြောင်း တစ်ခုခု ကြောင့် database ချိတ်ဆက်မှု့မဖြစ်ခဲ့ရင် Error တတ်မလာအောင်လို့ပါ။အင်တာနက် အသုံးပြုပြီး ချိတ်ဆက်တဲ့အခါမှာ internet connection ကျနေတာမျိုး၊Localhost မှာဆိုလဲ service တွေမဖွင့်မိတာမျိုးဖြစ်နေရင် error မတတ်ဖို့ပါ။except Exception as e: ဆိုပြီးလုပ်ထားတယ် Error တစ်ခုတတ်တယ်ဆိုရင် sys.exit ဆိုတာဖြစ်မယ် Message လဲထွက်လာမယ်ပေါ့ဗျာ။sys.exit ဆိုတာ program က error ကြောင့်မဟုတ်ဘဲ ကျွန်တော်တို့ပေးချင်တဲ့ Message လာပြီးပိတ်သွားတာမျိုးပေါ့။

cursor = db.cursor() ဆိုပြီး connection object ကို database နဲ့ချိတ်ဆက်ပါတယ်။ကျွန်တော်တို့ basic query လုပ်ဖို့ရပြီပေါ့ basic query ဆိုတာ UPDATE, INSERT , SELECT, DELETE စတာတွေပေါ့။အရင်ဆုံး def နဲ့ show_info() ကိုတည်ဆောက်တယ်။ဒီထဲမှာလုပ်ထားတဲ့ အလုပ်က SELECT * FROM info ဆိုတာပေါ့နောက်ပြီး result ထဲမှာ cursor.fetchall() နဲ့ အဖြေထုတ်လိုက်တယ် အဲဒါကို loop ပတ်ပြီး အဖြေထုတ်ပါတယ်။ဒီ Function ကိုအခေါ်ခံရရင်ပေါ့။

နောက်တစ်ခုက insert_data() ဖြစ်ပါတယ် user input သုံးခုလက်ခံထားပြီး info table ထဲက columns တွေဖြစ်တဲ့ Name , Email , BloodType ထဲကိုထည့်ပါတယ်။ID ကတော့ auto increment ဖြစ်လို့ သူ့အလို လျောက် row တစ်ခုဖြင်လာတိုင်း 1 တိုးမှာဖြစ်လို့ထည့်ပေးစရာမလိုပါဘူး။ဒီထဲမှာ Data ထည့်တာ ကလဲ sqlite3 တုန်းကလိုပါဘဲ ကွာသွားတာ တစ်ခုပါဘဲ sqlite မှာ VALUES(?,?,?) မှာ ? လေးတွေ သုံးပေမယ့် MySQL မှာတော့ %s %s နဲ့ဘဲသုံးပါတယ်။ဒါဘဲကွာတာပါ။ထည့်လိုက်တဲ့ Data တွေကို Save ရမယ် ဒီတော့ db ဆိုတဲ့ database connection ကိုယူပြီး db.commit() နဲ့ save ပါတယ်။

နောက် Function ကတော့ update() ဖြစ်ပါတယ်။Name , Email , BloodType သုံးခုကို UPDATE လုပ်ဖို့ပါဒါပေမယ့် ဘယ်အရာကို UPDATE လုပ်မယ်ဆိုတာကို WHERE နဲ့ ID ကိုအရင်ဆုံးရွေးချယ် UPDATE လုပ်နိုင်မှာ ဖြစ်ပါတယ်။ထားပါတော့ 1 ကိုရွေးရင် ID 1 ရဲ့နေရာကို UPDATE လုပ်မှာဖြစ်ပါတယ်။UPDATE လုပ်ပြီးရင်လဲ commit() နဲ့ save ရပါတယ်။

နောက် Function ကတော့ delete_data() ဖြစ်ပါတယ်။ဘယ် row ကို Delete မလဲဆိုတာကို ကျွန်တော်က WHERE နဲ့ Name ကိုရွေးပြီး delete လုပ်ခိုင်းစေထားပါတယ်။ဥပမာ AungMoeKyaw လို့ရွေးလိုက်ရင် AungMoeKyaw နဲ့မည်နဲ့ Row တွေအကုန် Delete လုပ်သွားမှာပါ။Delete လုပ်ပြီးရင်လဲ commit() နဲ့ Save ရပါတယ်။

အောက်မှာ while နဲ့ နာပတ်ရွေးချယ်အလုပ်လုပ်တာတော့ထပ်ခါထပ်ခါပြောထားတာ ဖြစ်လို့ မပြောတော့ပါဘူး။ဒါဆိုရင် MySQL နဲ့ချိတ်ဆက်ခြင်း အခြေခံကိုနားလည်သဘောပေါက်ပြီလို့ မျှော်လင့်ပါတယ်။နောက် Database တစ်ခုကတော့ အခုခေတ်စားလာတဲ့ Firebase Online Database ဖြစ်ပါတယ်။

Firebase Online Database with Python

Firebase ဆိုတာ Database အမျိုးအစားတစ်ခုပါဘဲ ဒါပေမယ့် SQL မဟုတ်တဲ့ အမျိုးစား ဖြစ်ပါတယ် ဒါကြောင့် SQL Statement တွေသုံးတာနဲ့မတူပါဘူး။အခုနောက်ပိုင်း Database တွေအနေနဲ့ ဆိုရင်တော့ Firebase က လွယ်ကူရိုးစင်းပြီးတော့ Powerful ဖြစ်တဲ့ DB ဖြစ်တယ်လို့ဆိုပါတယ်။Android ,iOS နဲ့ တခြားသော Programming တော်တော်များများဟာ server-side libraries ဒါမှမဟုတ် REST API တွေနဲ့ Firebase ကို Backend အနေနဲ့ချိတ်ဆက်နိုင်ပါတယ်။ဒီမှာတော့ ကျွန်တော်က Firebase နဲ့ Python ချိတ်ဆက်မှု့အနည်းငယ်ပြုလုပ်မှာဖြစ်ပါတယ်။နောက်ထွက်မယ့် Electronics Python စာအုပ်မှာ IOT projects တွေ ပြု

လုပ်တဲ့အခါမှာဒီ Firebase နဲ့ Python , C# ,VB , Android တို့နဲ့သုံးမှာဖြစ်ပါတယ်။ဒီမှာတော့ အနည်းငယ်သာဖော်ပြမှာဖြစ်ပါတယ်။ဒါဆိုရင် Firebase ရဲ့ Features အနည်းငယ်လေ့လာကြည့်ရအောင်။

Firebase Features

- Real-time Database – Firebase ဟာ JSON data နဲ့ Users တွေကို ချိတ်ဆက်ပြီး အချိန်နဲ့တပြေးညီ data update တွေကိုလက်ခံနိုင်ပါတယ်။ဥပမာ - Messager Chat Application တွေလိုပေါ့။
- Authentication – Firebase ဟာ anonymous အနေနဲ့ သော်လည်းကောင်း social authentications တွေဖြစ်တဲ့ google , Facebook တို့နဲ့လည်း authentication လုပ်နိုင်ပါတယ်။
- Hosting – Firebase ရဲ့ Hosting ဟာ လုံခြုံမှု့ကောင်းကောင်းနဲ့ အသုံးပြုနိုင်ပါတယ်။ပြောရရင် လုံခြုံမှု့ရှိတယ်ပေါ့ဗျာ။

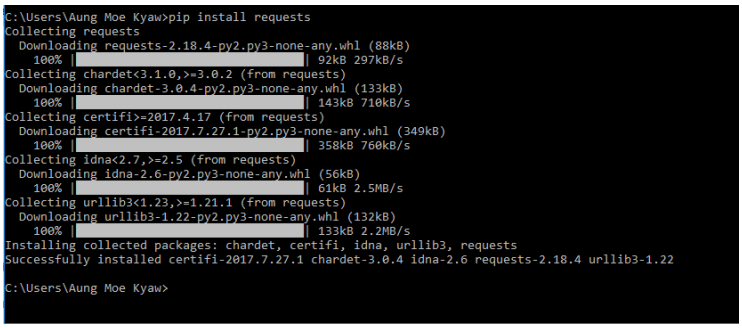
Firebase Advantages

- Firebase က user Friendly ဖြစ်တယ် ပြောရရင် သုံးရလွယ်တယ်ပေါ့ဗျာ ရှုပ်ထွေး တဲ့ ပြင်ဆင်မှု့တွေကိုလုပ်စရာမလိုဘဲ အသုံးပြုနိုင်ပါတယ်။
- နောက်တစ်ခုကတော့ real-time data အသုံးပြုနိုင်ခြင်းပါ ပြောတဲ့သဘောက အမြဲတမ်း အလိုလျောက် Update တွေကို clients တွေစီကနေပြုလုပ်ပေးနိုင်ပါတယ်။
- နောက်တစ်ခုက dashboard လို့ခေါ်တဲ့ သူ့ရဲ့ Control Panel ကြီးက ရိုးစင်းပြီးသုံးရလွယ်တာ ဘဲ့။ မိုက်တယ်။

Limitation အနေနဲ့ Free ပေးထားတဲ့ Firebase service က Real-time DB အတွက် 100 Connection ချိတ်ဆက်နိုင်တယ် Stoarge အနေနဲ့ 1GB ဖြစ်တယ် GB Downloaded အနေနဲ့ 10GB ဖြစ်ပါတယ်။အသေးစိတ်ကို https://firebase.google.com/pricing/ မှာ ဖတ်ရှု့နိုင်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ Python နဲ့ချိတ်ဆက်ဖို့လုပ်ရအောင်။သူ့ရဲ့ Officaially အနေနဲ့မရှိပေမယ့် ကံကောင်းထောက်မစွာနဲ့ ကျွန်တော် supported Module တစ်ခုတွေ့ထားတယ် Module ကို Develop လုပ်တဲ့ Developer ကိုကျေးဇူးတင်ပါတယ်။

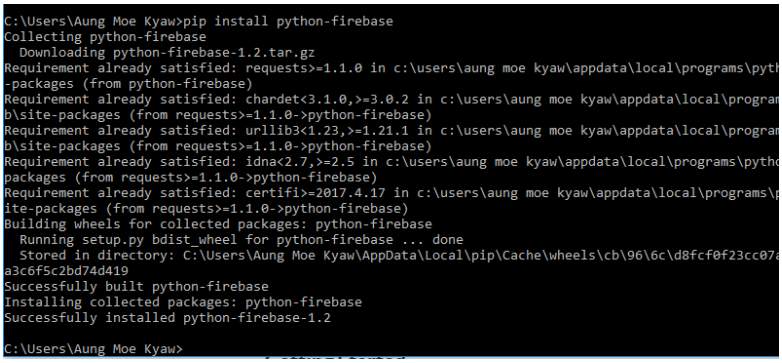
ဒါဆိုရင် အရင်ဆုံးလိုအပ်တဲ့ requests module ကို install လုပ်မယ်။

pip install requests



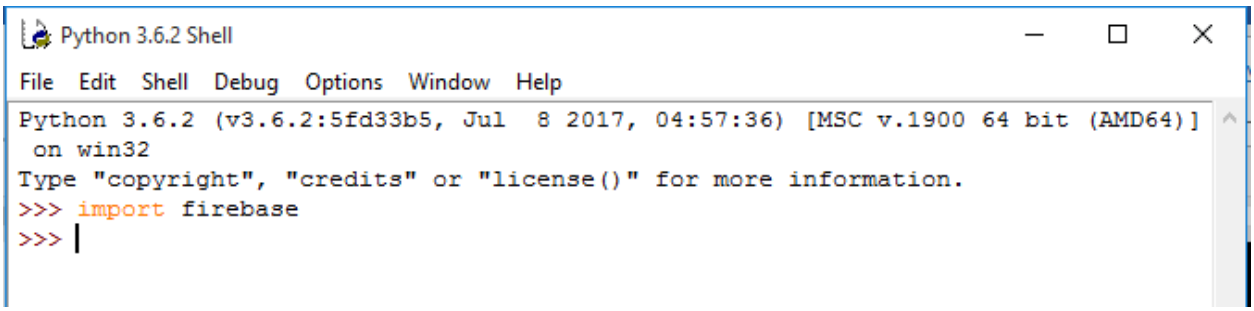
နောက်ထပ် install လုပ်ရမှာကတော့ Firebase Module ကိုပါ။

pip install python-firebase



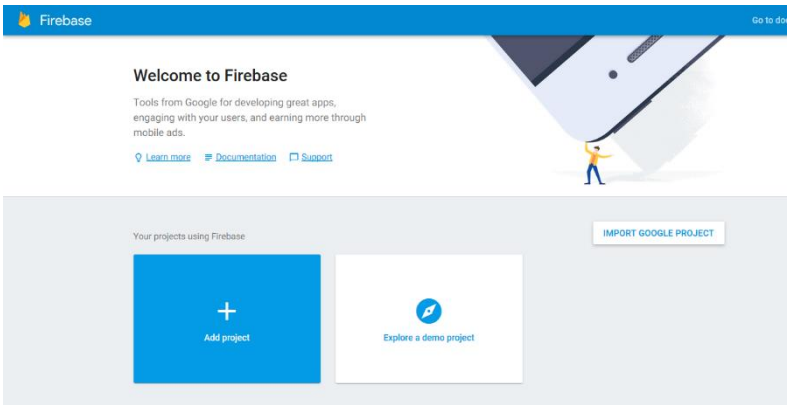
အထက်ပါအတိုင်း CMD မှာ Linux မှာဆိုရင်တော့ Terminal မှာပေါ့ အထက်ပါအတိုင်း Module တွေ ကို install လုပ်ပြီးတဲ့ အခါမှာ module တွေအလုပ်လုပ်လား မလုပ်လားသိရအောင် စမ်းကြည့်မယ်။

Python Shell ဖြစ်တဲ့ IDLE ကိုဖွင့်လိုက်ပါ။ပြီးရင် ကျွန်တော်တို့ import firebase လို့ရိုက်ကြည့်ပါမယ် firebase library ကို install လုပ်တာ မအောင်မြင်ဘူးဆိုရင် သေချာပေါက် Error တတ်ပါမယ်။ဘာ Error မှ မပြရင်တော့ အောင်မြင်တယ်လို့ယူဆရမှာပါ။

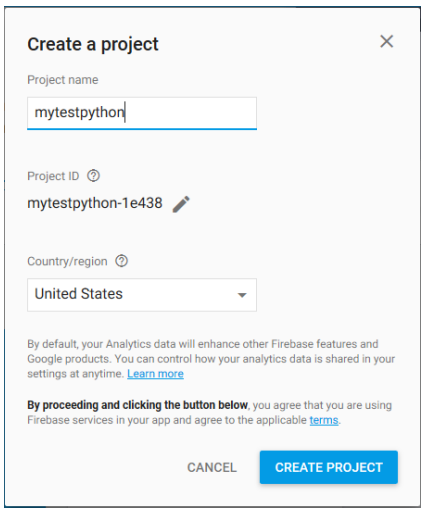


ဘာ Error မှမပြတဲ့ အတွက်အောင်မြင်ပါတယ်။ဒီတော့ ကျွန်တော်တို့ Firebase မှာ အကောင့်တစ်ခု သွား လုပ်ပြီး Free Plan အသုံးပြုကြတာပေါ့။

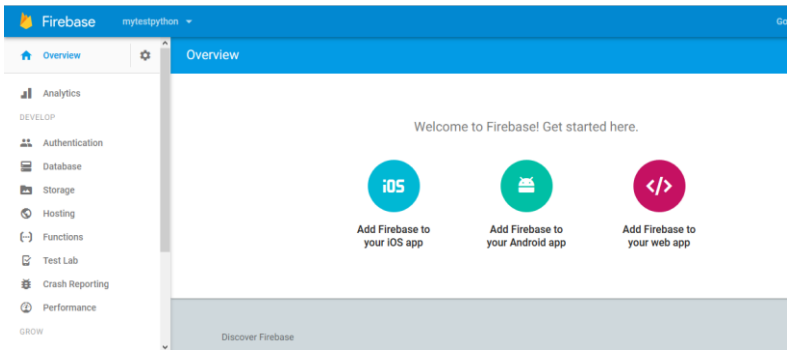
https://firebase.google.com/ မှာ GET STARTED ကိုနှိပ်ပြီး မိမိရဲ့ gmail အကောင့်နဲ့ဘဲ Login ဂင် လိုက်ပါဗျာ။



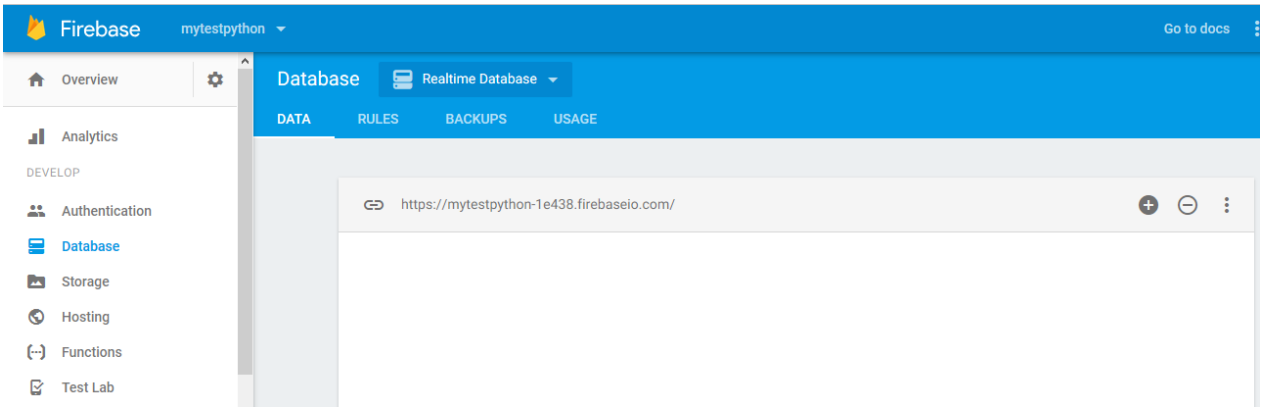
ဒီစာမျက်နှာကိုရောက်တဲ့အခါမှာ Add project ကိုနှိပ်ပေးပါ။project name နေရာမှာ ကြိုက်တာပေးလို့ ရပါတယ်။



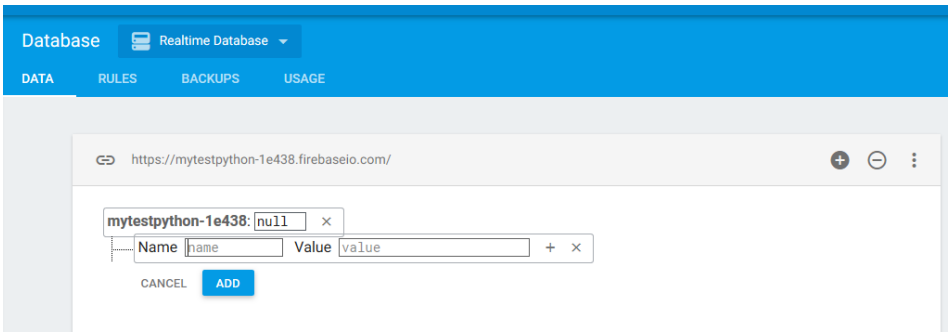
ပြီးရင် CREATE PROJECT ပေါ့ဗျာ။



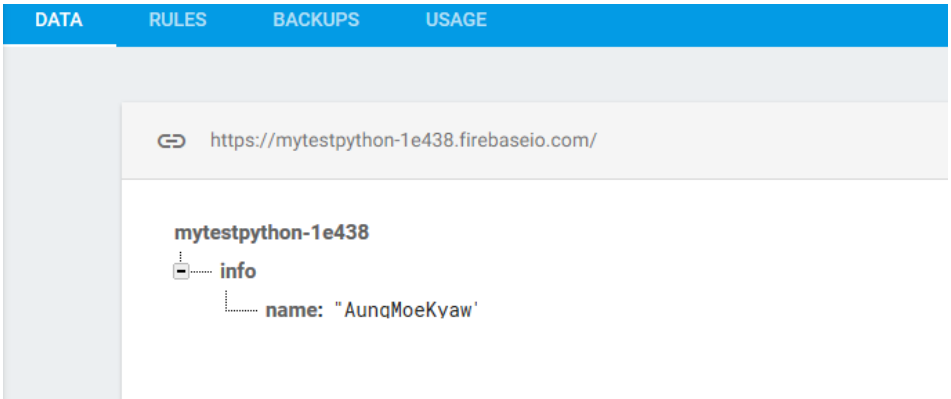
ဒီစာမျက်နှာရောက်ပြီဆိုရင်တော့ ကျွန်တော်တို့ Firebase DB တစ်ခုကို ပိုင်ဆိုင်လိုက်ပြီ ဖြစ်ပါတယ်။ကျွန်တော်တို့က Database သုံးချင်တာဖြစ်လို့ Database ကိုရွေးပေးလိုက်ပါ ပြီးရင် GET START ကို နှိပ်လိုက် တဲ့ အခါမှာ ကျွန်တော်တို့ရဲ့ Database စီရောက်သွားမှာဖြစ်ပါတယ်။



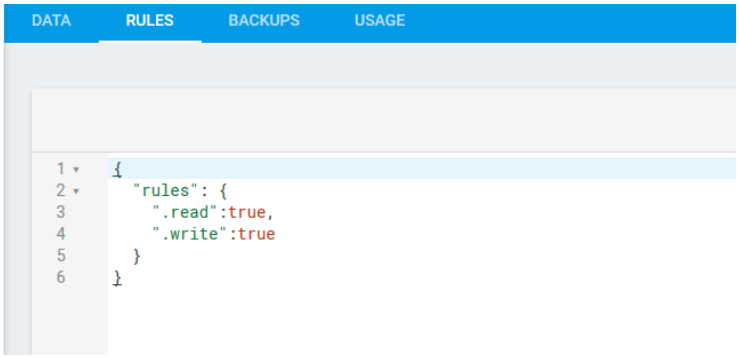
ဒါဆိုရင် ကျွန်တော်တို့ Data နည်းနည်းထည့်ပါမယ်။ဘေးဘက်က + လေးကိုနှိပ်လိုက်တဲ့ အခါမှာ ကျွန်တော်တို့အတွက် စာလေးတစ်ခုထွက်လာပါတယ်။အဲဒီထဲမှာ null ထဲမှာ ဘာမှမထည့်ဘဲ + လေးထပ် နှိပ်ပြီး Data Folder တစ်ခုတည်ဆောက်လိုက်တယ် + ထပ်နှိပ်ပါ ပြီးတော့မှ Name ထဲမှာ name ကို ထည့် ပြီး Value ထဲမှာ AungMoeKyaw ဆိုတာလေးထပ်ဆောက်ထားပါတယ်။ဒါက ဘာလဲဆိုတော့ info ရဲ့ အောက်မှာ name ဆိုတဲ့ Key ထဲမှာ AungMoeKyaw ဆိုတဲ့ Data လေးသိုလှောင်လိုက်တာပါဘဲ။



အထက်ပါပုံအတိုင်း Data ထည့်တည်ဆောက်ပါတယ်။



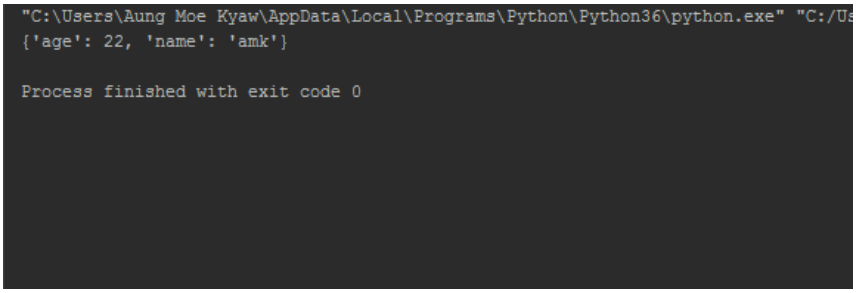
ဒါကတော့ တည်ဆောက်ထားတဲ့ Data Table ဖြစ်ပါတယ်။ဒါပြီးရင် ကျွန်တော်တို့ Rules တွေ ကိုပြောင်း ရမယ် ဒါကြောင့် Rules ဆိုတဲ့ Tab လေးကိုနှိပ်ပါ။နဂိုက ပါတဲ့ စာများကို အောက်ပါအတိုင်းပြင်လိုက်ပါ။



read ရော write ရော true ပေးပြီး publish လုပ်လိုက်ပါပြီးရင်တော့ ကျွန်တော်တို့ Database ထဲက Data တွေကို လက်ခံကြည့်လို့ရပါပြီ။

```
from firebase import firebase

firebase = firebase.FirebaseApplication('https://mytestpython-
1e438.firebaseio.com/',None)
result = firebase.get('/info',None)
print(result)
```

အရင်ဆုံး from firebase ကနေပြီး import firebase ဆိုပြီး firebase module ကို import လုပ်ပါတယ်။ နောက်ပြီး object တစ်ခုတည်ဆောက်ပါတယ် firebase ဆိုတဲ့နာမည်နဲ့ပါဘဲ။ပြီးရင် firebase.FirbaseApplication() နဲ့ connection object တစ်ခုတည်ဆောက်ပါတယ် အဲဒီထဲမှာ Argument ၂ ခုပါပါတယ်။တစ်ခုက connection link ပေါ့ အဲဒါက Database Dashboard အပေါ်မှာရှိပါတယ် ကျွန်တော်တို့ .firebaseio.com ဆိုတာနဲ့ဆုံး တဲ့ Link ဖြစ်ပါတယ်။နောက်တစ်ခုကတော့ None ပါ။ဒီ Firbase မှာ Data တွေကို JSON နဲ့ ခေါ်ယူပါတယ်။JSON ဟာ .json နဲ့အဆုံးသတ်လေ့ရှိတဲ့ URL ဖြစ်ပါတယ်။Data တွေကိုယူတဲ့အခါမှာ HTTPS request တွေက ကျွန်တော်တို့ရဲ့ browser တွေကိုဖြတ်ပြီး .json တွေကို Data တွေကိုပို့ပါတယ်။တခြားသော REST APIs တွေကဲ့သို့ဘဲ ဒီ python-firebase module က အလုပ်လုပ်နိုင်ပါတယ်။ဘာတွေလဲဆိုတော့ client တွေ အနေနဲ့ firebase ကို update(PATCH,PUT), create(POST), or remove(DELETE) တွေကိုပေါ့။နောက်ပြီး သူ့ထဲမှာသိမ်းထားခဲ့တဲ့ Data တွေကိုလည်း ရယူခွင့်ပြုထားပါတယ်။နောက်တစ်ခုအနေနဲ့ object တစ်ခု သို့မဟုတ် variable တစ်ခုကိုတည်ဆောက်ပြီး firebase object ကိုခေါ်ပြီး firebase.get() ဆိုပြီး get() method နဲ့ firebase ထဲက data တွေကိုရယူထားပါတယ်။get() method မှာ Argument ၂ ခုပါတယ်။Argument ၁ ကတော့ callback function မျိုးနဲ့ request ပို့ပါတယ် ပြောရရင် firebase database အောက်က info ဆိုတဲ့နေရာကိုလမ်းညွှန်းပေးတာဖြစ်ပါတယ်။နောက်ကတစ်ခုကတော့ None ပါ ကျွန်တော်တို့က None လို့ရေးထားတဲ့အတွက်ဘာကိုမှသီးသန့်ရွေးမထားဘူးဆိုတဲ့ အဓိပ္ပါယ်ကြောင့် info.json ကိုခေါ်ပါတယ် အဲဒီနောက် result ဆိုတဲ့ object ကိုအဖြေထုတ်တဲ့အခါမှာ info အောက်မှာရှိတဲ့ တစ်နည်းအားဖြင့်ပြောရရင် info.json ထဲမှာရှိတဲ့ data အားလုံးကိုအဖြေအဖြစ်ထုတ်ပေးပါတယ်။



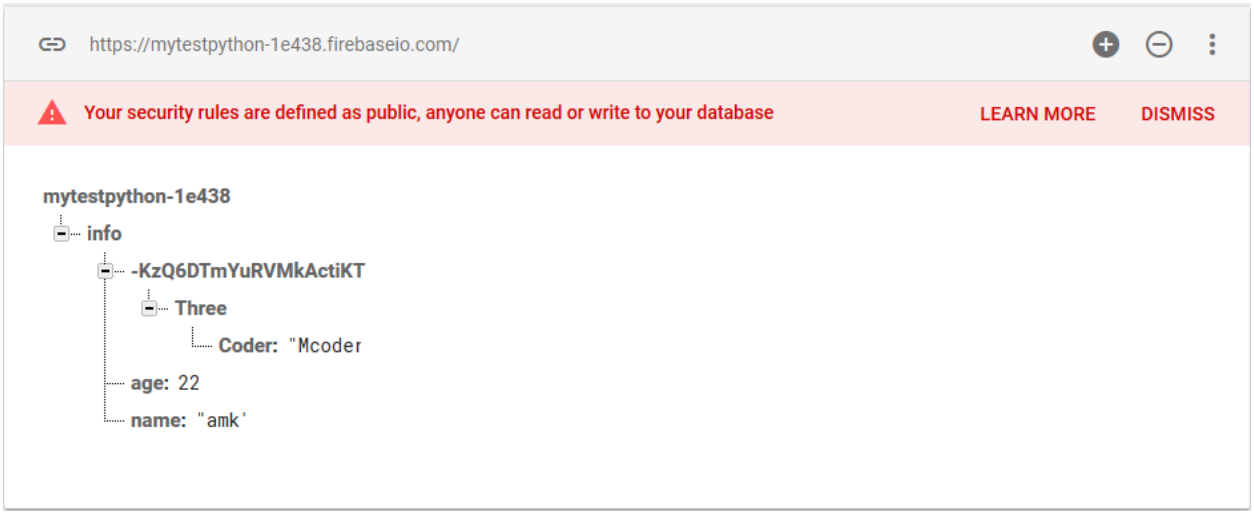
အထက်ပါပုံစံအတိုင်းအဖြေထွက်လာမှာဖြစ်ပါတယ်။ဒါက Argument ၂ မှာ None ရှေ့းထားလို့ ဖြစ်ပါတယ်။ဒီလို မှမဟုတ်ဘဲ။None နေရာမှာ age ကိုရိုက်ထည့်လိုက်မယ်ဆိုရင် /info/age.json ကို သူက အဖြေ အဖြစ်ထုတ် ပေးမှာဖြစ်ပါတယ်။Age နဲ့ ဆိုင်တဲ့ 22 ကိုပေါ့။ဒီသဘောတရားက Dictionary နဲ့ဆင်တူပါတယ်။ Key တစ်ခုကိုရိုက်ထည့်တဲ့အခါမှာသူ့နဲ့သက်ဆိုင်တဲ့ Value တစ်ခုကိုထုတ်ပေးတာဘဲဖြစ်ပါတယ်။

```
result = firebase.get('/info','age')
```

ဒီလိုမှမဟုတ်ဘဲ name ကိုရိုက်ထည့်မယ်ဆိုရင်တော့ amk ဆိုတဲ့ value အဖြေထွက်လာမှာဘဲဖြစ်ပါတယ်။ဒါက Database ထဲက data ထုတ်ယူတာပါ ဒါဆိုရင် Database ထဲကိုဘယ်လိုထည့်မလဲ။

```
post = firebase.post('/info',{'Three':{'Coder':'Mcoder'}})
```

firebase.post() ဆိုတာနဲ့ database ထဲကို save ပါတယ်။အခုအခါမှာ '/info' ဆိုတာ info အောက်ထဲမှာ { } နဲ့ Three ဆိုတာ Key တစ်ခုတည်ဆောက်ပြီး အဲဒီအောက်ထဲမှာဘဲ Coder Mcoder ဆိုတဲ့ Key နဲ့ value တစ်စုံကိုထည့်သွင်းပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ firebase ထဲကိုပြန်ကြည့်မယ်။



ဒါပေမယ့် post နဲ့ထည့်တဲ့အခါမှာ သူ့ဘာသာ ထုတ်သွားတဲ့ data column တစ်ခုက ဘာတွေမှန်းမသိဘဲ များပြားတယ်ဒီလိုမျိုးမဖြစ်စေချင်ဘူး နောက်ပြီး dictionary တွေတည်ဆောက်တဲ့အခါမှာလည်း post က { 'Three' : {'1':'2'} } ဆိုတာမျိုးရေးနေရတယ်ဒါကိုပြေလည်ဖို့ Python-firebase developer က ရေးပေးထားပါတယ်။firebase.put() ဆိုတဲ့ function ဖြစ်ပါတယ်။ဒီဟာက argument နှစ်ခုထည့်ရတယ်။တစ်ခုက info

ဆိုတဲ့ firebase column ဖြစ်ပြီးနောက်တစ်ခုက ကျွန်တော်တို့အသစ်ထည့်ချင်တဲ့ column နဲ့ value တွေ ဖြစ်ပါတယ်။

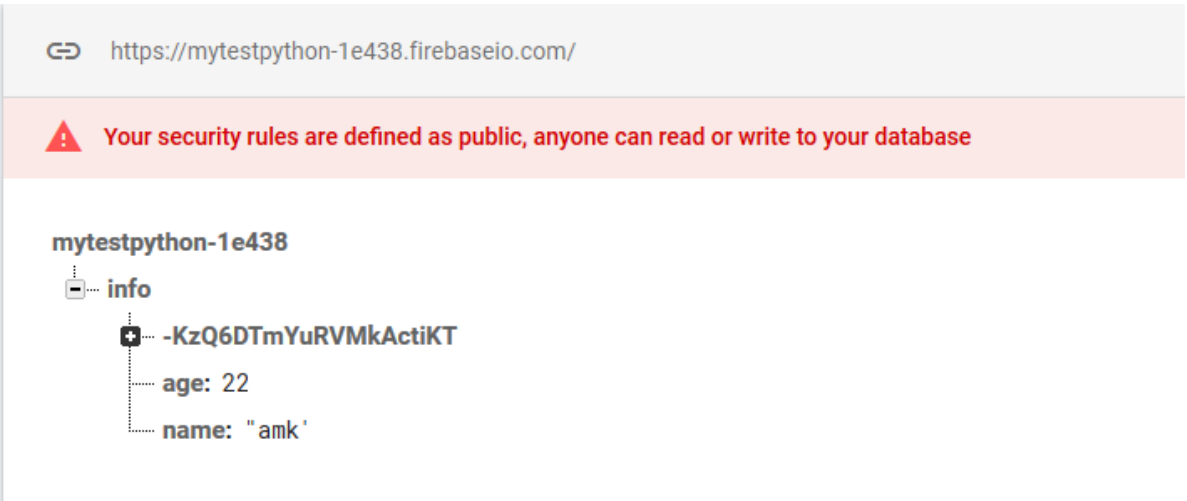
```
put = firebase.put('info','Three',{'mcoder':'mm'})
```

ဒီလိုရေးလိုက်တာက Three ဆိုတဲ့ column တစ်ခုကိုတည်ဆောက်မယ် သူ့ထဲမှာ mcoder ဆိုတဲ့ Key နဲ့ mm ဆိုတဲ့ value ကိုထည့်မယ်လို့ပြောလိုက်တာဘဲဖြစ်ပါတယ်။

ကျွန်တော်တို့ထည့်ထားတဲ့ Three ဆိုတဲ့ column နဲ့ mcoder : mm ဆိုတာကိုတွေ့ရမှာပါ။ဒါဆိုရင် ကျွန်တော်တို့ database ထဲက ထုတ်ယူတာ သိမ်းဆည်းတာကိုလုပ်ပြီးသွားပြီဖြစ်ပါတယ်။remove လုပ်ဖို့ delete လုပ်ဖို့ဘဲလိုပါတော့တယ်။

```
delete = firebase.delete('/info','Three')
```

firebase.delete() ထဲမှာ 'info' ဆိုတဲ့ column ရယ် ကျွန်တော်တို့ delete လုပ်ချင်တဲ့ Column ရယ်ကို ရေး ပေးလိုက်ရင်ရပါတယ်။အထက်က Command မှာဆိုရင် Three ကို delete လုပ်မယ်လို့ပြော ထားတာဖြစ်ပါတယ်။



ဒါဆိုရင် firebase သင်ခန်းစာတချို့ပြီးပါပြီ ကျွန်တော့်ရဲ့ Blog မှာ firebase နဲ့ဆိုင်တဲ့ Tutorials တွေ တင်ပေးပါအုန်းမယ်နောက်ထွက်မယ့်စာအုပ်မှာလဲ Esp8266 နဲ့ Firebase သုံးပြီး အသုံးပြုမယ့် Applications တွေ အကြောင်းလဲပါဝင်မှာဖြစ်ပါတယ်။

Creating the GUI Form with Python Tkinter

Python မှာ GUI လို့ခေါ်တဲ့ Graphical User Interface တည်ဆောက်ဖို့ အတွက် Libraries တွေ များစွာရှိတဲ့ကြားက လွယ်ကူပြီးသုံးရတာ ရိုးရှင်းတဲ့ Tkinter Library ကိုအသုံးပြုပြီး ကျွန်တော်တို့ လေ့လာသွားမှာဖြစ်ပါတယ်။Tkinter Module က python က တရားဝင်ထောက်ပံ့ပေးထားတဲ့ module လည်း ဖြစ်ပါတယ်။နောက်တစ်ချက်ပြောရအုန်းမယ် code လေး ၄ ကြောင်း လောက်နဲ့ Windows Form တစ်ခုတည်ဆောက်လို့ရပါတယ်။

Tkinter ဆိုတာ Tk ရဲ့ Python အတွက် GUI အထောက်ပံ့တစ်ခုဖြစ်ပါတယ်။GUI toolkit တွေက Tcl/Tk စတာတွေပေါ့။Tcl ဆိုတာ Tool Command Language တစ်ခုဖြစ်ပါတယ် popular ဖြစ်တဲ့ scriptin language တစ်ခုလည်းဖြစ်ပါတယ်။တခြားတစ်ဖက်မှာတော့ Tk ဟာ open source ဖြစ်တယ် widget လို့ခေါ်တဲ့ GUI ပေါ်က Button လို Textview လို စပ်ဆက်ပစ္စည်းလေးတွေဟာ multi-platform တွေ အတွက် အထောက်ပံ့ပေးနိုင်ပြီး မတူညီတဲ့ different languages တွေနဲ့လည်း GUI တည်ဆောက်နိုင်အောင် အထောက်ပံ့အများကြီးပေးပါတယ်။Tkinter ကိုတော့ Python module အနေနဲ့ Tkinter.py ဆိုပြီး Python2.x versions နဲ့ tkinter/__init__.py ဆိုပြီး Python 3.x version တွေမှာပါဝင်ပါတယ်။Tkinter ကို C extension တွေနဲ့ထုတ်ပိုးထားပြီး ဒီ Tkinter ဟာ Tcl/Tk libraries တွေကိုအသုံးပြုပြီးဘဲတည်ဆောက်ထားပါတယ်။

Tkinter ဟာ ဘာမှမဟုတ်တဲ့ module တော့မဟုတ်ပါဘူး။သာမန်သုံးတဲ့ desktop applications တွေကနေ scientific modeling တွေအထိ တည်ဆောက်နိုင်ပါတယ်။Python ရဲ့ အခြေခံရေးထုံးတွေနဲ့ programming အခြေခံ အယူအဆတွေကို နားလည်သွားတဲ့အခါမှာ Tkinter ကိုဆက်ပြီးလေ့လာတဲ့ အခါမှာ ပိုပြီးလွယ်ကူမှာဖြစ်သလို မြန်မြန်သင်ယူတတ်မြောက်စေပါတယ်။ဒါ့အပြင် အပေါ်မှာပြောခဲ့သလိုဘဲ GUI Windows Form တစ်ခုကို code ၄ ကြောင်းလောက်နဲ့တည်ဆောက်နိုင်ပါတယ်။Tkinter module ဟာ Python အတွက် လုံးဝလန်းတဲ့ Module တစ်ခုဖြစ်ပါတယ်။

Tkinter ရဲ့ကောင်းကျိုးတချို့ကိုဖော်ပြလိုက်ပါတယ်

- Tkinter ဟာ ရိုးရှင်းပြီးလေ့လာဖို့လွယ်ကူတယ်။(တခြားသော GUI Module တွေထက်ပေါ့)
- Code နည်းနည်းနဲ့ Powerful ဖြစ်တဲ့ GUI Applications တွေတည်ဆောက်နိုင်တယ်။
- တည်ဆောက်ပုံစနစ်ဟာ အလွှာလိုက်ပုံစံဖြစ်ပြီး တစ်ခုချင်းအလိုက်တည်ဆောက်နိုင်ပါတယ်။
- Tkinter ဟာ portable ပုံစံဖြစ်ပြီး Operating systems တိုင်းနဲ့အလုပ်လုပ်နိုင်ပါတယ်။

- Tkinter ဟာ အလွယ်တကူအသုံးပြုနိုင်ပါတယ် နောက်ပြီး Python မှာနဂို Library အနေနဲ့ပါဝင် ပြီးဖြစ်ပါတယ်။

ဒီသင်ခန်းစာမှာတော့ GUI Windows Form တစ်ခုကို Tkinter သုံးပြီးဘယ်လိုတည်ဆောက်မလဲဆိုတာကိုလေ့လာမှာဖြစ်ပါတယ်။အဲဒီနောက်မှာတော့ Widgets တွေအကြောင်းကိုထပ်မံလေ့လာမှာဖြစ်ပါတယ်။

widgets ဆိုတာကတော့ labels, buttons , text boxes, combo boxes, check buttons နဲ့ radio buttons စတဲ့ GUI နဲ့စပ်ဆက်အထောက်ကူပစ္စည်းတွေကိုခေါ်တာပါ။

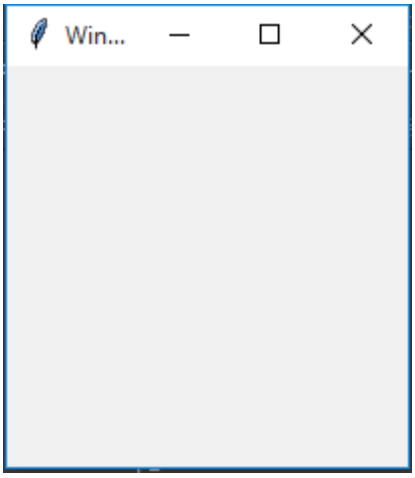
Creating Our first Python GUI

Python ဟာ powerful ဖြစ်တဲ့ programming language တစ်ခုပါ။Tkinter နဲ့ပေါင်းလိုက်တော့ ဦးသိန်းတန် စကားလိုပါဘဲ ကြယ်ငါးပွင့်သဘော်ကြီးဖြစ်သွားပါတယ်။Code နည်းနည်းနဲ့ Windows Form တစ်ခုတည်ဆောက်ကြည့်ရအောင်။

IDLE ကိုတောင် Tkinter နဲ့ တည်ဆောက်ထားတာပါ.....

```
import tkinter as tk
win = tk.Tk()
win.title("Windows Form")
win.mainloop()
```

ဒီ Code လေးကြောင်းထဲကဘဲ အောက်ပါ Windows Form ကိုတည်ဆောက်နိုင်ခဲ့ပါတယ်။



တကယ့်ကို Code နည်းနည်းလေးနဲ့ပါ။ဒါဆို Code တွေကိုရှင်းရအောင်။အရင်ဆုံး tkinter as tk ဆိုပြီး tkinter module ကို import လုပ်ပြီး tk ဆိုပြီးသတ်မှတ်လိုက်ပါတယ်။ပြီးတဲ့အခါမှာ win ဆိုတဲ့ variable ကိုတည်ဆောက်ပြီး tk.Tk() function ကိုခေါ်လိုက်ပြီး object တည်ဆောက်လိုက်ပါတယ်။win ဆိုတဲ့နာမည်နဲ့ပေါ့ဗျာ။ဒီတော့ Tk() ထဲမှာရှိသမျှ functions တွေရော variables တွေရောအကုန်လုံးကို win ဆိုတဲ့ object

ကိုခေါ်ပြီးသုံးလို့ရပြီပေါ့ဗျာ ဒီတော့ win.title() ကိုခေါ်ပြီး Windows Form မှာ နာမည်ပေးတဲ့အလုပ်ကိုလုပ်ပါတယ်။Form မှာ နာမည်ပေးချင်ရင် win.title("") နဲ့သုံးပါတယ်။နောက်ဆုံးတစ်ခုကတော့ win.mainloop() ပါ။ဒီ Function ကတော့အရေးကြီးပါတယ် mainloop() ဆိုတဲ့ Function က ကျွန်တော်တို့ရေးမှာရေးခဲ့တဲ့ while လိုပါဘဲ program ကို ပိတ်မသွားအောင်ထပ်ခါထပ်ခါ Run နေအောင်လုပ်ပေးပါတယ်။

The GUI Form Resized

Form Resized လုပ်တယ်ဆိုတာ GUI Form ကို သတ်မှတ်တာဖြစ်ပါတယ်။Minimize , Maximize လုပ်တာတွေပေါ့။ဒီအလုပ်လုပ်ဖို့အတွက် Function တစ်ခုခေါ်ဖို့လိုပါတယ်။ resizable(0,0) အသုံးပြုရပါတယ် 0,0 ထားလိုက်တဲ့အခါမှာ Resize လုပ်လို့မရတော့ပါဘူး Maximize လုပ်လို့မရတော့ပါဘူး။

```
import tkinter as tk
win = tk.Tk()
win.title("Windows Form")
win.resizable(0,0)
win.mainloop()
```

Adding a Label to the GUI Form

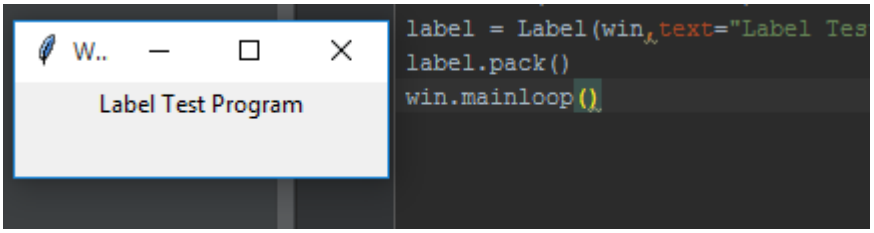
Label တစ်ခု ကျွန်တော်တို့ GUI ထဲကိုထည့်မယ်ဗျာ။Label ဆိုတာ widgets ထဲမှာပါတာပေါ့ဗျာ။ widgets ဆိုတာ Label, Button, Textbox စတဲ့ GUI Control တွေအကုန်ပါပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ widget ဖြစ်တဲ့ Label ကိုဘယ်လိုထည့်မလဲကြည့်ရအောင်။

widget_object =widget(Windows_Form_object,**widget's_configuration_option)

widget_object တည်ဆောက်ပြီး ကျွန်တော်တို့တည်ဆောက်ချင်တဲ့ widget method ကိုရေးပြီး နောက်ကမှာ Argument တွေအနေနဲ့ နဂိုတည်ဆောက်ထားတဲ့ Windows_Form_object ကိုရေးပါတယ်။ဒီ Form ပေါ်မှာ ဒီ widget ကိုတည်ဆောက်မယ်လိုပြောလိုက်တာပါဘဲ။ပြီးရင် widget's_configuration တွေထည့်ရေးရပါတယ်။ဒါဆိုရင် Label widget ကိုတည်ကိုယ်တိုင်တည်ဆောက်ကြည့်ရအောင်။

```
from tkinter import *
win = Tk()
win.title("Windows Form")
label = Label(win,text="Label Test Program")
label.pack()
win.mainloop()
```

tkinter import * ဆိုတာ tkinter ထဲက အားလုံးကို import လုပ်လိုက်တာပါ။အဲဒီနောက်မှာ win ဆိုတဲ့နာမည်နဲ့ Form method ဖြစ်တဲ့ Tk() ကို ယူပြီး Form object တည်ဆောက်လိုက်ပါတယ်။နောက်ပြီး Form object ထဲက title method ကိုယူပြီး Windows Form မှာ Title ပေးပါတယ်။ပြီးရင်တော့ ကျွန်တော်တို့ လိုချင်တဲ့ Label ကိုတည်ဆောက်ဖို့ label ဆိုတဲ့ object variable ကိုရေးတယ် အဲထဲမှာ Label method ကိုခေါ်ပြီး Form Object ကို ထည့်ရေးပေးတယ် ဒါက inheritance လုပ်လိုက်တာပေါ့ Form ကို Parent သဘောလုပ်လိုက်တာပါ။ အဲဒီမှာရေးမယ့်စာကို text နဲ့ Label Test Program လို့ရေးပါတယ်။ဒါက Label ပေါ်မှာဖော်ပြမယ့်စာပါ။ဒါဆိုရင် ကျွန်တော်တို့က Label ကိုတော့တည်ဆောက်ပြီးသွားပြီ ဒါပေမယ့် အသုံးပြုလို့မရသေးပါဘူး အသုံးပြုလို့ရအောင် widget object ဖြစ်တဲ့ label ကိုခေါ်ပြီး label.pack() ကို အသုံးပြုပြီး widget ကိုနေရာချခြင်းကိစ္စလုပ်ရပါတယ်။pack() method ကတော့ widget တွေကို နေရာချတဲ့ ကိစ္စလုပ်ပေးတဲ့ method ဖြစ်ပါတယ်။နောက်ပြီး pack() က widget အတွက် မဖြစ်နေသုံးရမယ့် essentially required ဖြစ်ပါတယ်။နောက်ထပ်မဖြစ်မနေသုံးရမယ့် essentially requirement ကတော့ win.mainloop() ဘဲဖြစ်ပါတယ်။ဒါက ဘာလုပ်တယ်ဆိုတာ ကျွန်တော်တို့အပေါ်မှာပြောခဲ့ပြီးပါပြီ။



Run လိုက်ရင် အထက်ပါပုံအတိုင်း Windows Form တစ်ခုတည်ဆောက်နိုင်ပြီး ကျွန်တော်တို့ Label ကိုလဲ အထက်ပါပုံအတိုင်းမြင်ရမှာဖြစ်ပါတယ်။

Widgets တွေက Object စစ်စစ်ဖြစ်ပါတယ်။ဒီလိုဗျာ object တစ်ခုတည်ဆောက်ပြီး widget class တွေစီကနေ object အဖြစ်ရယူထားလို့ဘဲဖြစ်ပါတယ်။widget_object = widget(it's parent,configurations) မြင်သာပါတယ်နော့။

parent ကတော့ Tk() class ကို inheritance လုပ်ထားတဲ့ parent ပါ။configurations မှာတော့ attributes တွေဖြစ်တဲ့ Labels ထဲက Text တွေ Text Colors တွေ နဲ့ Font size တွေ Font Type တွေ စတာ တွေပြင်ဆင်နိုင်ဖို့အတွက် အသုံးပြုနိုင်တဲ့ attributes တွေထည့်နိုင်ပါတယ်။configurations တွေကို widget object ကနေ .config() or .configure() သုံးပြီး attributes တွေထည့်နိုင်ပါတယ်။သူတို့ method နှစ်ခု လုံးကတော့လုပ်ဆောင်ချက်အတူတူပါဘဲ။

```
label = Label(win)
label.config(text='Hello World',bg='red',fg='white')
label.pack()
```

အခုဆိုရင် config() နဲ့သုံးပြီးတော့ text= နဲ့ Hello World ဆိုတဲ့ Text ရေးမယ်။bg ဆိုတာ background color ဖြစ်ပါတယ် red ထည့်ထားပါတယ်။fg ကတော့ foreground color ဖြစ်ပါတယ်။

pack() ကတော့ widgets တွေနေရာချတဲ့ method ဖြစ်ပါတယ်။ဒါကိုမသုံးရင်တော့ တခြား geometry managers တွေအသုံးပြုပြီးနေရာချနိုင်ပါတယ်။

Getting to Know the core Tkinter widgets

အခုကျွန်တော်တို့ Tkinter ရဲ့ widgets တွေအကြောင်းကိုလေ့လာကြမယ်ဗျာ။ကျွန်တော်တို့ Label အကြောင်းကိုတော့ သိခဲ့ပြီးဖြစ်ပါတယ်။အခုတခြား widgets တွေအကြောင်းကို လေ့လာလိုက်ကြရအောင်။

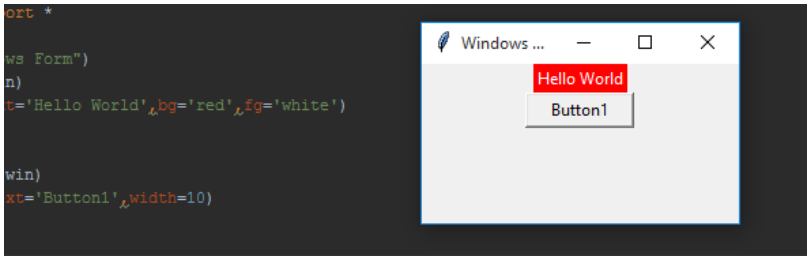
Tkinter မှာ widgets 21 ခုရှိပါတယ်။အဲဒါတွေကအောက်ပါအတိုင်းဖြစ်ပါတယ်။

Toplevel widget	Label widget	Button widget
Canvas widget	Checkbutton widget	Entry widget
Frame widget	LabelFrame widget	Listbox widget
Menu widget	Menubutton widget	Message widget
OptionMenu widget	PanedWindow widget	Radiobutton widget
Scale widget	Scrollbar widget	Spinbox widget
Text widget	Bitmap Class widget	Image Class widget

Button တစ်လုံးထည့်ပြီးတည်ဆောက်ကြည့်ရအောင်။

```
from tkinter import *
win = Tk()
win.title("Windows Form")
label = Label(win)
label.config(text='Hello World',bg='red',fg='white')
label.pack()

button = Button(win)
button.config(text='Button1',width=10)
button.pack()

win.mainloop()
```

button နာမည်နဲ့ object တစ်ခုတည်ဆောက်ပါတယ်။Button method ကိုတည်ဆောက်တယ် win ကို parent အဖြစ်ထားတယ်။button.config(text='Button1',width=10) လို့ရေးတာဟာ Button ပေါ်မှာ Button1 ဆိုတဲ့ စာလုံးဖော်ပြမယ်လို့ရေးတယ် width = 10 ကတော့ Button ရဲ့ Size ကိုသတ်မှတ်လိုက်တာပါ။



Adding widgets to a parent window

Widgets တွေရဲ့အသုံးပြုပုံတချို့ကိုဖော်ပြလိုက်ပါတယ်။အနည်းငယ်ဘဲကွဲပြားမှုရှိပါတယ်။တော်တော်များများက ဆင်တူပါတယ်။

Label(parent, text = “Enter your Password:”)

Button(parent,text=”Search”)

Checkbutton(parent,text=”Remember Me”,variable=v,value=True)

Entry(parent,width=30)

Radiobutton(parent,text=”Male”,variable=v,value=1)

Radiobutton(parent,text=”Female”,variable=v,value=2)

OptionMenu(parent,var,”Select Country”,”USA”,”UK”,”India”,”Other”)

Scrollbar(parent,orient=VERTICAL,command=text.yview)

ကျွန်တော်တစ်ခုခြင်းသုံးပြသွားမယ်။နောက်ပြီး ကျွန်တော်တို့အတူ GUI projects တွေလုပ်ကြတာပေါ့ဗျာ။ဒါဆိုရင် အရင်ဆုံး Frame တစ်ခုတည်ဆောက်ပြီး အဲဒီ Frame ထဲမှာ widgets တွေထည့်မယ် ပြီးရင် geometry managers တွေကိုအသုံးပြီးတော့နေရာချတာပြုလုပ်နိုင်ပါတယ်။

```
frame = Frame(win)
frame.geometry_Managers()
```

အထက်ပါအတိုင်းပါဘဲ frame object ကိုတည်ဆောက်ပြီး Frame() method ကို win ဆိုတဲ့ Tk() class ကို parent အဖြစ်အသုံးပြုထားပါတယ်။အဲဒီနောက်မှာမှ frame ကို geometry_Managers() method တွေကို နေရာခေါ်ပြီးနေရာချရပါတယ်။Geometry Managers ၃ မျိုးရှိပါတယ်။

The Tkinter Geometry Manager

အရင်ကျွန်တော်တို့သုံးခဲ့တဲ့ pack() က Geometry Manager တစ်မျိုးဖြစ်ပါတယ်။ဒါပေမယ့် pack() တခုထဲရှိတာတော့မဟုတ်ပါဘူး အခြား 2 ခုရှိနေပါသေးတယ်။ဒါတွေကိုလေ့လာကြည့်ရအောင်။

The three geometry managers are as follows:

- pack:ဒီmethod ကလည်း GM တစ်ခုပါဘဲ။ဒါပေမယ့် ဒါကိုအသုံးပြုဖို့တော် အတော်စဉ်းစားရမှာဖြစ်ပါတယ်။ဒါက ရိုးရိုးရှင်းရှင်းနဲ့အသုံးပြုနိုင်ပါတယ် ရိုးရှင်းတဲ့ Layout တွေမှာဆိုရင်ပေါ့။ ဒါပေမယ့် ရှုပ်ထွေးပြီးများပြားတဲ့ Layout တွေအတွက်ဆိုရင်တော့ သုံးဖို့ခက်ခဲပြီးရှုပ်ယှက်ခက်စေပါလိမ့်မယ်။
- grid:ဒီmethod ကတော့ အများဆုံးအသုံးပြုတဲ့ method ဖြစ်ပါတယ်။Table ပုံစံမျိုးနဲ့ Layout ချနိုင်ပါတယ်။ဒီ GM method ကတော့ Layout Management ကို အလွယ်ဆုံးပြုလုပ်နိုင်မှာပါ။

- place:ဒါကနောက်ဆုံးတစ်ခုဖြစ်ပြီး popular ဖြစ်တဲ့ GM method တစ်ခုဖြစ်ပါတယ်။ဒီ method က widgets တွေကိုနေရာချတဲ့အလုပ်ကိုအကောင်းဆုံးလုပ်နိုင်တယ်လို့လဲဆိုထားပါတယ်။

The Pack Geometry Manager

ဒီ pack method က စာနဲ့ရှင်းပြရင်နားလည်ဖို့ခက်ခဲပါလိမ့်မယ်။ဒါကြောင့် code ရေးရင်းရှင်းတာက ပိုပြီးရှင်းလင်းမယ်လို့ထင်ပါတယ်။ဒါကြောင့် code base နဲ့သွားကြတာပေါ့ဗျာ။ဒါပေမယ့် သိထား သင့်တာတွေကိုတော့ ကြိုရှင်းထားပရစေ။Tkinter ရဲ့ author Fredrik Lundh ကတော့ windows form ကို ချုံလို့ချဲ့လို့ရတဲ့ sheet တစ်ခုလို့ပုံဖော်ထားပါတယ်။Tk() ကို run လိုက်တဲ့အခါမှာ အကျဉ်းအကျယ်လုပ်လို့ ရတဲ့ windows form လေးကိုပြောတာပါ။pack ကတော့ ဒီ elastic sheet လေးထဲက hole လေးလိုပါဘဲ ပြောရရင် sheet လေးပေါ်မှာအဖောက်လေးတွေဖောက်ထားသလိုပေါ့။default အနေနဲ့ကတော့ pack က ထည့်တဲ့ widget တွေက အပေါ်ဘက်အစွန်းမှာကပ်နေလေ့ရှိပါတယ်။

pack နဲ့ widgets တွေထည့်သွင်းတဲ့အခါမှာ pack manager က အောက်ပါအခြေနေသုံးခုရဲ့ space အတိုင်းကွဲပြားခြားနားပါတယ်။

- The unclaimed space
- The claimed but unused space
- The claimed and used space

အသုံးအများဆုံး pack options တွေကိုအောက်ပါအတိုင်းဖော်ပြလိုက်ပါတယ်။

- side : LEFT,TOP,RIGHT and BOTTOM (ဒါတွေက widget တွေရဲ့နေရာကို alignment လုပ်နိုင်ပါတယ်။)
- fill : X, Y, BOTH and NONE(ဒါကတော့ widget တစ်ခုမဟုတ်တစ်ခုကို size အကြီးအသေး ပြောင်းလဲနိုင်ပါတယ်။)
- expand : Boolean ကတော့ tkinter.YES or tkinter.NO , 1 or 0 , True or False
- anchor : NW,N,NE,E,SE,S,SW,W and CENTER (ဒီ option တွေကတော့ မှုလ သချာ်ဦးတည်ရာညွှန်းကိန်းတွေနဲ့လိုက်လျောညီထွေရှိပါတယ်။)
- Internal padding : ipadx and ipady ကတော့ widgets တွေကို အတွင်းပိုင်းက padding လုပ် ပေးပါတယ်။
- external padding : padx and pady ကတော့ widgets ကိုအပြင်ကနေ padding လုပ်ပါတယ် default တန်ဖိုးကတော့ zero ပါ။

ဒါဆိုရင် နမူနာ program ရေးကြည့်ရအောင်။

```
from tkinter import *
win = Tk()
win.title("Windows Form")
```

```
Label(win,text="Tkinter pack() Example:").pack()
frame = Frame(win)
Button(frame,text='A').pack(side=LEFT,fill=Y)
Button(frame,text='B').pack(side=TOP,fill=X)
Button(frame,text='C').pack(side=RIGHT)
Button(frame,text='D').pack(side=TOP)
frame.pack()

Label(win,text="Tkinter pack() Normal:").pack()
Button(win,text="Button").pack()
Button(win,text="Button").pack()
Button(win,text="Button").pack(side=LEFT,padx=2,pady=2,expand=1)
Button(win,text="Button").pack(side=LEFT,padx=2,expand=1)
Button(win,text="Button").pack(side=LEFT,padx=2,expand=1)
win.mainloop()
```

ဒီ program မှာ frame object ကိုတည်ဆောက်ပြီး Frame method ကိုယူပါတယ် Tk() object ကို parent အနေနဲ့သုံးပါတယ်။အဲဒီ Frame ထဲမှာတော့ Button ၄ခုပါဝင်ပါတယ်။ပထမ Button က side=LEFT ဆိုတာ ကျွန်တော်တို့ Button ကို ဘယ်ဘက်ကိုပို့လိုက်ပါတယ် fill = Y ကတော့ Y ပုံစံဆိုတော့ ထောင်လိုက်မျဉ်းမတ်ပုံစံပေါ့။နောက်ထပ် B ကတော့ TOP ဆိုတော့အပေါ်ပါ X သုံးထားလို့ အလျားလိုက် အလဲ့အတိုင်းပေါ့။နောက်တစ်ခုကတော့ C ပါ RIGHT ဆိုတော့ ညာဘက်ကိုဖြစ်ပါတယ်။D ကတော့ TOP အလည်ပါ။frame.pack() နဲ့ pack နေရာချလိုက်ပါတယ်။နောက်ထပ်ကတော့ ဒီအတိုင်း ဘာ option မှမသတ်မှတ်ထားတဲ့ Button ၂ ခု ကို ထည့်ပါတယ်။side တွေအနေနဲ့ LEFT နဲ့ဘဲထားလိုက်ပါတယ် ဒါက ဘာဖြစ်စေချင်လို့လဲဆိုတော့ အတန်းလိုက်ကလေးဖြစ်စေချင်လို့ပါ။နောက်ထပ်ပါတဲ့ option ကတော့ padx=2 ပါ external padding လုပ်တာပါ Button တစ်ခုနဲ့တစ်ခုကြားကို ခြားတာဖြစ်ပါတယ်။padx ဆိုတော့ အလျားလိုက် x အတိုင်းခြားတာပါ။pady ကတော့ ထောင်လိုက် Y အတိုင်းခြားတာဖြစ်ပါတယ်။expand =1 ကတော့ GUI ကို ချဲ့ချဲ့လုပ်တဲ့ အခါမှာ Button တွေနေရာရွှေ့မသွားအောင် အခုလိုနေရာချထားတဲ့အတိုင်းဘဲဖြစ်အောင်လုပ်လိုက်တာပါ။

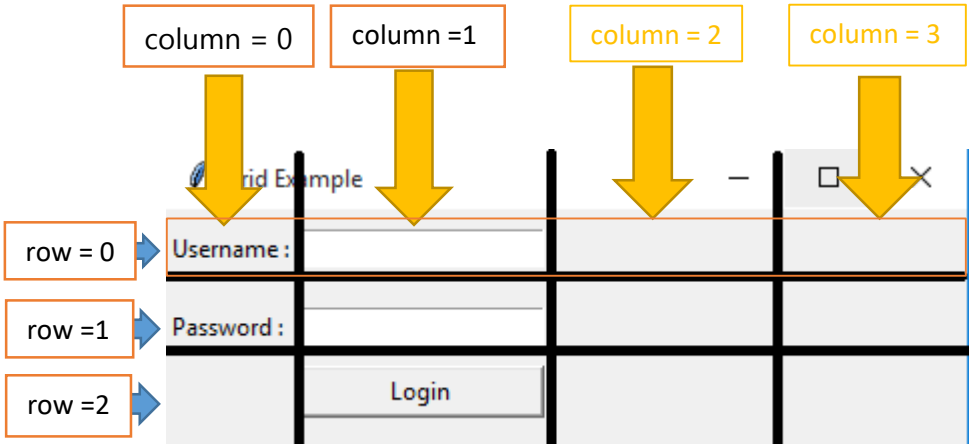
ဒါဆိုရင် pack() အကြောင်းအနည်းငယ်သိသွားပြီဆိုရမှာပါ။ဒါဆိုရင်နောက်ထပ် GM ဖြစ်တဲ့ grid() အကြောင်းဆက်သွားရအောင်။

The grid geometry managers

The grid geometry managers ဟာ Tkinter မှာ နားလည်ဖို့လွယ်ကူပြီ အလွန်လွန်ကောင်းတဲ့ အသုံးအဝင်ဆုံး geometry manager ဖြစ်ကောင်းဖြစ်နိုင်တယ်လို့ Tkinter GUI Application Development Blueprints စာအုပ်မှာဖော်ပြထားပါတယ်။ဒီလောက်ညွှန်နေရင်တော့ ခေသူမဟုတ်လောက်ဘူး။grid အတွက် အဓိကကျတဲ့ idea ကိုပြောရမယ်ဆိုရင် widgets တွေကို တည်ဆောက်ပြင်ဆင်နိုင်ဖို့အတွက် frame contrainer မှာ two-dimensional table ပုံစံမျိုးနဲ့တည်ဆောက်ပါတယ် ဒီ table တွေကို rows တွေ columns တွေအဖြစ်ထပ်မယ်ခွဲလိုက်ပါတယ်။ဒီ table တစ်ခုချင်းစီရဲ့ cell တွေက widgets တွေကိုစီမံခန့်ခွဲ စေနိုင်ပါတယ်။ဒီစကားကိုဆက်ပြောရမယ်ဆိုရင် cell ဆိုတာ widgets တွေဘဲဖြစ်ပါတယ်။ဒီ cell လေးတွေကို table ပေါ်မှာတင်ပြီး ဘယ် row ဘယ် column မှာထားမယ်လို့သတ်မှတ်ပေးခြင်းဖြင့်မိမိလိုရာ ပုံစံ GUI ကို တည်ဆောက်နိုင်ပါတယ်။sticky ကတော့ အဲဒီ cell ထဲမှာရှိတဲ့ widget တွေကို ချိန်ညှိဖို့အတွက်သုံးတာပါ။

```
from tkinter import *

win = Tk()
win.geometry("400x400")
win.title("Grid Example")

frame = Frame(win)

Label(frame,text="Username :").grid(column=0,row=0,sticky=W)
Label(frame,text="Password :").grid(column=0,row=1,sticky=W)
Entry(frame).grid(column=1,row=0,pady=10,sticky=E)
Entry(frame).grid(column=1,row =1,pady = 10,sticky=E)
Button(frame,text="Login").grid(column=1,row=2,sticky=NSEW)
frame.grid()

win.mainloop()
```

ဒီ program မှာ method အသစ်တချို့ပါလာတယ် အရင်ဆုံးတွေ့ရမယ့်အသစ်က geometry ပါ သူ့က Form size ကိုသတ်မှတ်ပေးနိုင်တဲ့ method ဖြစ်ပါတယ်အခုဆိုရင် 400x400 size ရှိတဲ့ Form တစ်ခုပေါ်လာမှာဖြစ်ပါတယ်။နောက်ထပ်ကတော့ frame တည်ဆောက်တယ် အဲ Frame ထဲမှာ Label ၂ ခု Entry ၂ ခု ပြီးတော့ Button ၁ ခုပါတယ်။grid ကိုသုံးထားတယ်။ပထမဆုံး Label က column 0 မှာရှိတယ် Entry တစ်ခုကတော့ column 1 မှာပေါ့ row ကတော့ 0 မှာရှိပါတယ်။နောက်ထပ် Label တစ်ခုကလည်း column 0 ဖြစ်ပါတယ် Entry ကလည်း column 1 မှာပါဘဲ ဒါပေမယ့် row က 1 ဖြစ်သွားပါတယ် ဘာကြောင့်လဲဆိုတော့ row ဆိုတာ အောက် ထောင်လိုက် Y ပုံစံအတိုင်းဆင်းသွားတဲ့ အရာကိုခေါ်တာဖြစ်ပါတယ်။နောက်ထပ် row 2 နဲ့ column 1 မှာတော့ Button တစ်လုံးပါတည်ဆောက်ထားပါတယ်။ပုံကြည့်လိုက်ရင်ရှင်းသွားမှာပါ။



ဒီပုံကိုကြည့်လိုက်တော့ ရှင်းရှင်းလင်းလင်းဖြစ်သွားပြီလို့ထင်ပါတယ်။ဒါဆိုရင် sticky တွေ ဘာကြောင့် သုံးလဲကြည့်ရအောင် sticky = W လို့သုံးလိုက်တာ Left-Hand-Side ဆိုပြီးသတ်မှတ်လိုက်တာဖြစ်ပြီး E ကတော့ Right ပေါ့ဗျာ။Button မှာသုံးထားတဲ့ NSEW ကတော့ column =1 မှာရှိတဲ့ widgets တွေရဲ့ အကျယ် အတိုင်းဆွဲဆန့်ပေးလိုက်တာပေါ့။ကိုယ်တိုင်ချိန်စရာမလိုဘူး NSEW လို့ထည့်လိုက်ရင် ဆွဲဆန့်ပေးသွားမှာပါ။Entry တစ်ခုနဲ့တစ်ခုကြားမှာ padding အခြားလေးဖြစ်စေဖို့ pady ကိုတန်ဖိုးပေးထားပါတယ်။

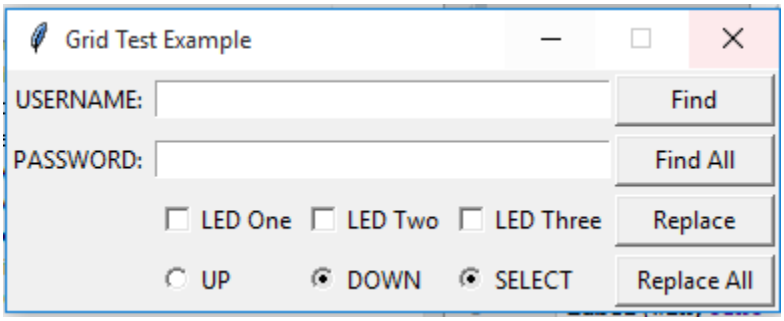
ဒါဆိုရင် grid() အတွက် ယေဘုယ options တွေကတော့ row, column, padx, pady , sticky, rowspan , columnspan. တို့ဘဲဖြစ်ပါတယ်။

sticky အလုပ်လုပ်ပုံက အပေါ်မှာ ပြောခဲ့သလိုဘဲ widgets တွေကိုလိုက်ချိန်ညိပေးတာဗျာ။သေချာရှင်းအောင် လုပ်ကြည့်ပြီးတော့သုံးကြည့်ရအောင်။

```
Entry(frame,width=50).grid(column=1,row=0,pady=10,sticky=W)
Entry(frame).grid(column=1,row =1,pady = 10,sticky=E)
Button(frame,text="Login").grid(column=1,row=2,sticky=NSEW)
```

အခုဆိုရင် Entry တစ်ခုမှာ width =50 ပေးထားပါတယ်။အဲဒီတော့ ဒီ Entry ရဲ့ အကျယ်က 50 ဖြစ်နေတာပေါ့ဗျာ။အခု sticky = w ပေးထားတဲ့အတွက် Left side ဘက်ကပ်ထားတာပေါ့။ဒါပေမယ့်သိပ်မမြင်သာဘူးဗျာ ဒါ ကြောင့် ကျွန်တော် တို့က width = 50 မထားဘဲ ပုံမှန် Entry တစ်ခု တည်ဆောက်လိုက်တယ် ပြီးတော့ ကျွန်တော်က sticky=E ထားလိုက်တော့ widget လေးက ညာဘက်ကိုကပ်သွားမှာဖြစ်ပါတယ်။W ထည့်လိုက်ရင် ဘယ်ဘက်ကို ကပ်နေမှာဖြစ်ပါတယ်။S နဲ့ N တစ်ခုခုထားရင်တော့ Center ဖြစ်နေပါလိမ့်မယ် အပေါ်က widget အတိုင်း side အပြည့်ဖြစ်ချင်ရင်တော့ NSEW ကိုသုံးနိုင်ပါတယ်။widget တစ်ခုကို cell တစ်ခုအဖြစ် တည်ဆောက်လိုက်ပါတယ်။အဲဒီ widget ရဲ့width တွေ hight တွေက အဲဒီ cell ရဲ့ပမာဏဘဲပေါ့ဗျာ အဲဒီ cell အတိုင်းချိန်ညိနိုင်အောင် sticky သုံးတာဖြစ်ပါတယ်။

```
from tkinter import *
win = Tk()
win.title("Grid Test Example")
win.resizable(0,0)
Label(win,text="USERNAME: ").grid(column=0,row=0,sticky='e')
Entry(win,width=30).grid(column=1,row=0,sticky='we',pady=5,columnspan=9)

Label(win,text="PASSWORD: ").grid(column=0,row=1,sticky='e')
Entry(win,width=30).grid(column=1,row=1,sticky='we',pady=5,columnspan=9)

Button(win,text="Find",width=10).grid(row=0,column=10,sticky='e'+'w',pady=2,padx=2)
Button(win,text="Find
All",width=10).grid(row=1,column=10,sticky='e'+'w',pady=2,padx=2)
Button(win,text="Replace",width=10).grid(row=2,column=10,sticky='e'+'w',pady=2,padx=2)
Button(win,text="Replace
All",width=10).grid(row=3,column=10,sticky='e'+'w',pady=2,padx=2)

Checkbutton(win,text="LED One").grid(row=2,column=2,sticky='w')
Checkbutton(win,text="LED Two").grid(row=2,column=3,sticky='w')
Checkbutton(win,text="LED Three").grid(row=2,column=4,sticky='w')

Radiobutton(win,text="UP",value=1).grid(row =3,column=2,sticky='w')
Radiobutton(win,text="DOWN",value=2).grid(row =3,column=3,sticky='w')
Radiobutton(win,text="SELECT",value=3).grid(row =3,column=4,sticky='w')

win.mainloop()
```



အထက်ပါ Code တွေက အထက်ပါ GUI Form တည်ဆောက်တဲ့အခါမှာအသုံးပြုထား တဲ့ code တွေ ဖြစ်ပါတယ်။columnspan = 9 ဆိုတာလေးဘဲရှင်းမယ် အဲဒီ columnspan =9 လို့သတ်မှတ်လိုက်တဲ့အခါမှာ အသတ်မှတ်ခံရတဲ့ Entry က columnspan =9 အထိသတ်မှတ်ခံရသလိုဖြစ်ပြီးတော့ column=10 လို့ အသတ်မှတ်ခံရမယ့် widget က အဲဒီ columnspan = 9 နောက်မှာ သွားပြီးရှိနေမှာဖြစ်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့ GUI တည်ဆောက်တဲ့အခါမှာ အခက်ခဲဖြစ်စရာမရှိတော့ပါဘူး။ဒါက Grid ကိုအသုံးပြုပြီး GUI တည်ဆောက်တဲ့စနစ်ဖြစ်ပါတယ်။ကဲနောက်တစ်ခုတည်ဆောက်ကြည့်ရအောင်။

The place geometry manager

Place geometry manager ကတော့ Tkinter မှာရှားရှားပါးပါးအသုံးပြုတဲ့ အမျိုးအစား မျိုး ဖြစ်ပါတယ်။ဒီ GM မှာတော့ widgets တွေရဲ့ position တွေကို parent frame ရဲ့ (x,y) coordinate ပုံစံနဲ့နေရာ ချအသုံးပြုပါတယ်။place manager ကို place() ကိုအသုံးပြုပြီး access လုပ်လို့ရတယ်။

Place ရဲ့ options တွေကို တစ်ချက်လေ့လာကြည့်ရအောင်၊အသုံးအများဆုံး အရေးကြီးဆုံး ကတော့ absolute positioning(options x =N or y =N) ဖြစ်ပါတယ်။Relative positioning (relx, rely, relwidth, relheight) တို့ဘဲဖြစ်ပါတယ်။ဒါဆိုရင်ရေးကြည့်ရအောင်။

```
from tkinter import *
win = Tk()
win.resizable(0,0)
Button(win,text='Absolute').place(x=70,y=10)
Button(win,text="Relative").place(relx=0.7,rely=0.4,relwidth=0.4,relheight=0.3,width=1
,anchor=E)
win.mainloop()
```

အထက်မှာဖော်ပြထားတဲ့ options တွေကိုချိန်ညှိပြီး GUI တည်ဆောက်နိုင်ပါတယ်။ကျွန်တော်ကတော့ GUI ကို grid() နဲ့ဘဲအမြဲတည်ဆောက်သွားမယ်။Widget တွေကို Style တွေပြင်ဆင်မယ်။အထက်မှာဖော်ပြခဲ့တဲ့ Tkinter program အားလုံးက Layout နေရာချတာဘဲလုပ်ရသေးတယ်။တကယ်အလုပ်လုပ်တဲ့ Function တွေ

ထည့်သွင်းအလုပ်မလုပ်ရသေးပါဘူးဒါကြောင့်ကျွန်တော်တို့တကယ့်အလုပ်လုပ်တဲ့ program ရေးကြရ အောင်ဗျာ။

```
from tkinter import *
from tkinter import ttk

def button_click():
    label.configure(text="Hello World")


if __name__ == "__main__":
    win = Tk()
    win.title("Test Program")
    label = Label(win)
    label.grid()
    button = Button(win,text="press",width=20,command=button_click)
    button.grid()
    win.mainloop()
```

ဒီ program မှာ method တစ်ခုကိုတည်ဆောက်ထားပါတယ် button_click() နာမည်နဲ့ပါ။ဒီထဲမှာလုပ်မယ့် အလုပ်ကိုမပြောသေးဘူးအရင်ဆုံး အောက်က အလုပ်တွေပြောမယ်။if __name__ == "__main__" ကတော့ program တစ်ခုတည်ဆောက်တဲ့နေရာမှာ main program မုန်းသိအောင်လုပ်တဲ့ method ဖြစ်ပါတယ်။ဒီ ထဲမှာ လုပ်ထားတဲ့အလုပ်တွေကတော့ GUI Form တစ်ခုတည်ဆောက်တယ်။title ထည့်တယ်။နောက်ထပ် label ဆိုတဲ့ object တစ်ခုကိုတည်ဆောက်ပြီးတော့ Label(win) ဆိုပြီး Label တစ်ခုကို ဘာစာမှမထည့်ဘဲ တည်ဆောက်လိုက်ပါတယ်။label object ကို grid() နဲ့ဘဲနေရာချလိုက်ပါတယ်။နောက်ထပ် button တစ်လုံးကို အရင်လိုဘဲ object တစ်ခုဆောက်ပြီး ထည့်လိုက်ပါတယ်။options တွေမှာတခြား option တွေကတော့ သိပြီး သားတွေပါ command တစ်ခုဘဲအပိုပါလာပါတယ်။အဲဒီ command ဆိုတာ Button ကိုနှိပ်လိုက်တဲ့အခါမှာ command မှာရှိတဲ့ function ကိုခေါ်ပေးနိုင်တဲ့ option ဖြစ်ပါတယ်။ဒီတော့ command မှာ ကျွန်တော်တို့ တည်ဆောက်ထားတဲ့ method ကိုလှမ်းခေါ်လိုက်ပါတယ်။အဲဒီ method ထဲမှာလုပ်တဲ့အလုပ်က label object ကို configure() method နဲ့options တွေပြင်ဆင်မှာဖြစ်ပါတယ်။ဒီထဲမှာတော့ Button ကိုနှိပ်တဲ့ အခါမှာ Hello World ဆိုတဲ့စာသားဖော်ပြအောင်ပြုလုပ်ထားတာဖြစ်ပါတယ်။

Messagebox in Tkinter

ကျွန်တော်တို့ Tkinter မှာ Messagebox ဘယ်လိုပြကြမလဲဆိုတာကို ဆွေးနွေးသွားမှာဖြစ်ပါတယ်။ ကျွန်တော် လက်လှမ်းမှီသလောက် control widgets တွေကိုတစ်ခုခြင်းဖော်ပြသွားမှာဖြစ်ပါတယ်။Messagebox ကိုသုံးဖို့အတွက်အရင်ဆုံး tkinter ထဲက messagebox ဆိုတဲ့ method ကို import လုပ်ရပါတယ်။

```
from tkinter import *
from tkinter import messagebox as msgbox
def press_button():
    msgbox._show("I am Messagebox Title","Hello,I'm Message")



if __name__ == "__main__":
    win = Tk()
    win.title("Messagebox Showing")
    win.resizable(0, 0)
    win.geometry("200x200")
    frame = Frame(win)
    Label(frame, text="Press Button").grid(column=0, row=0, columnspan=2)
    Button(frame, text='Press', width=15).grid(column=3, row=0, pady=5)
    frame.grid()
    win.mainloop()
```

အထက်က Program လိုပါဘဲ အရင်ဆုံး from tkinter import * ဆိုပြီးအားလုံးကို import လုပ်တယ် ပြီးတော့ from tkinter import messagebox နဲ့ messagebox ကို import လုပ်ပါတယ် as ဆိုပြီး msgbox ကိုလုပ်လိုက်ပါတယ်။messagebox ကို as msgbox နဲ့ကိုယ်စားပြုလိုက်ပါတယ်။ဒါဆိုရင် messagebox ကိုအသုံးပြုလို့ ရပါပြီ အခု def နဲ့ function တစ်ခုတည်ဆောက်ပြီး ဒီ function ကိုအခေါ်ခံရရင်လုပ်မယ့်အလုပ်ကတော့ msgbox._show() ဆိုတာပါ ဒါက messagebox ပေါ်လာဖို့သုံးတဲ့ method ပါ ။ဒီ method ကိုသုံးဖို့ အာ Arguments တွေသုံးနိုင်လဲကြည့်ရအောင်။

_show(title,message,**option)တို့ဖြစ်ပါတယ်။ Messagebox ကို Error တွေနဲ့ ကျွန်တော်တို့သိစေချင်တဲ့ အချက်လက်တွေကိုဖော်ပြချင်လို့အသုံးပြုတာပါ။

```
from tkinter import *
from tkinter import messagebox as msgbox

def msg_button():
    msgbox._show("I am Messagebox Title","Hello,I'm Message")

def asq():
    msgbox.askquestion("I am Askquestion Box","I am Askquestion")

def yes_no():
    msgbox.askyesno("I am Yes or No","I'm ask_yes_no")

def ask_ok_cancel():
    msgbox.askokcancel("I'm aksokcancel","I'm askokcancel")

def ask_retry_cancel():
    msgbox.askretrycancel("I'm askretrycancel","I'm AskretryCancel")

def ask_yes_no_cancel():
    msgbox.askyesnocancel("I'm Ask Yes No Cancel","I'm Ask Yes No Cancel")

def show_warning():
    msgbox.showwarning("I'm Show_Warning","I'm Show Warning")
```

```
if __name__ == "__main__":
    win = Tk()
    win.title("Messagebox Showing")
    win.resizable(0, 0)

    frame = Frame(win)
    Label(frame, text="Press").grid(column=0, row=0, columnspan=2)
    Button(frame, text='Message', width=15,command=msg_button).grid(column=3, row=0,
pady=5)

    Label(frame,text='Press').grid(column=0,row=1,columnspan=2)

Button(frame,text='Askquestion',command=asq).grid(column=3,row=1,sticky=NSEW,pady=5)

    Label(frame, text='Press').grid(column=0, row=2, columnspan=2)
    Button(frame, text='Askquestion',command=yes_no).grid(column=3, row=2,
sticky=NSEW,pady=5)

    Label(frame, text='Press').grid(column=0, row=3, columnspan=2)
    Button(frame, text='askyesno',command=ask_ok_cancel).grid(column=3, row=3,
sticky=NSEW, pady=5)

    Label(frame, text='Press').grid(column=0, row=4, columnspan=2)
    Button(frame, text='askokcancel',command=ask_ok_cancel).grid(column=3, row=4,
sticky=NSEW, pady=5)

    Label(frame, text='Press').grid(column=0, row=5, columnspan=2)
    Button(frame, text='askretrycancel',command=ask_retry_cancel).grid(column=3,
row=5, sticky=NSEW, pady=5)

    Label(frame, text='Press').grid(column=0, row=6, columnspan=2)
    Button(frame, text='askyesnocancel',command=ask_yes_no_cancel).grid(column=3,
row=6, sticky=NSEW, pady=5)

    Label(frame, text='Press').grid(column=0, row=7, columnspan=2)
    Button(frame, text='showwarning',command=show_warning).grid(column=3, row=7,
sticky=NSEW, pady=5)

    frame.grid()
    win.mainloop()
```

အထက်က program မှာ Button ၈ ခုတည်ဆောက်ပြီးမတူညီတဲ့ပုံစံ ၈ ခုနဲ့ပြသပေးတဲ့ Message ၈ မျိုးကို ဖော်ပြလိုက်ပါတယ်။code တွေကအရမ်းလွယ်လို့နားလည်မယ်ထင်ပါတယ်။ဒါဆိုရင်နောက်တစ်ခုစီ သွားရအောင်။

Tkinter Button State

Button State ဆိုတာ ကျွန်တော်တို့ control လုပ်မယ့် Button လေးတွေ disable မိုန်ပြီးနိုပ်လို့မရတဲ့အခြေနေနဲ့ active ထင်းနေပြီးနိုပ်လို့ရတဲ့အခြေနေ နှစ်ခုကိုပြောတာပါ။ဒီလိုလုပ်ဖို့အတွက်ဆိုရင် .config() ထဲက state = 'disable' or state = 'active' နှစ်ခုနဲ့ရေးနိုင်ပါတယ်။အောက်မှာ နမူနာ code လေးနဲ့ဖော်ပြလိုက်ပါတယ်။

```
from tkinter import *

def button1_state():
    btn1.config(state='disable')

def button2_state():
    btn2.config(state='disable')

def button3_state():
    btn1.config(state='active')
    btn2.config(state='active')


if __name__ =="__main__":
    win = Tk()
    win.title("Button State Tutorial")
    btn1 = Button(win, text='Button 1',command=button2_state)
    btn1.grid(column=0, row=0, padx=5)

    btn2 = Button(win, text='Button 2',command=button1_state)
    btn2.grid(column=3, row=0, padx=5)

    btn3 = Button(win, text='All Active', command=button3_state)
    btn3.grid(column=1, row=2, padx=5,sticky=N)
    win.mainloop()
```

ဒီ code လေး အလုပ်လုပ်ပုံကတော့ Button 1 ကိုနှိပ်ရင် Button 2 က disable ဖြစ်ပြီးမှိန်သွားမယ် နှိပ်လို့မရတော့ဘူး၊Button 2 ကိုနှိပ်ရင်တော့ Button 1 မှိန်သွားပြီးနှိပ်လို့မရဘူး၊Button 3 ကိုနှိပ်လိုက်ရင်တော့ Button 1 ရော Button 2 ရောဘယ်ဟာဘဲမှိန်နေနေ ပြန်ပြီး active အခြေနေပြန်ရောက်ပါမယ်။

Combo box widgets

Combo box ဆိုတာ တန်ဖိုးတွေရွေးလို့ရတဲ့ drop down control widget လေးတစ်ခု ဖြစ်ပါတယ်။ အရမ်းအသုံးဝင်တဲ့ widgets လေးတစ်ခုလဲဖြစ်ပါတယ်။ကျွန်တော်တို့ combo box ကိုအသုံးပြုတဲ့အခါမှာ tkinter ထဲက တိုက်ရိုက်အသုံးပြုလို့မရပါဘူး messagebox လိုဘဲ tkinter ထဲကမှ ttk ဆိုတဲ့ module ကို import လုပ်ပြီးသုံးမှရပါတယ်။ttk က လည်း အလန်းစား module တစ်ခုပါဘဲ။ttk ဆိုတာ themed tk လို့ အဓိပ္ပါယ်ရပါတယ် ကျွန်တော်တို့ရဲ့ GUI ကို ပိုပြီးလှအောင်ပိုပြီးကောင်းတယ် advanced widgets တွေပါဝင်တဲ့ extension တစ်ခုဖြစ်ပါတယ်။ဒီ Combobox ကလည်း ttk ထဲမှာဘဲပါဝင်ပါတယ်။

```
from tkinter import *
from tkinter import ttk as tk,messagebox as msgbox

def click_1():
    msgbox._show("Combobox Chosen",nameChosen.get())


if __name__ == "__main__":
    win = Tk()
    nameChosen = tk.Combobox(win, width=12)
    nameChosen['values'] = ("Devid", "San", "John", "Wany")
    nameChosen.grid(column=0, row=0)
    nameChosen.current(0)
```

```
    Button(win, text="Combo", command=click_1).grid(column=1, row=0)

    win.mainloop()
```

ဒီ program မှာ tkinter ထဲက ttk ကို as tk အဖြစ်ရော messagebox ကို as msgbox အဖြစ်ရော သတ်မှတ်လိုက်တယ်။ပြီးတော့ click_1 ဆိုတဲ့ function တစ်ခုတည်ဆောက်လိုက်ပါတယ်။ကျွန်တော်တို့အရင်တိုင်း GUI တစ်ခုတည်ဆောက်တယ် အဲဒီထဲမှာ nameChosen ဆိုတဲ့ object တစ်ခုနဲ့ tk.Combobox() ဖြစ်ပါတယ်။အဲဒီထဲမှာ nameChosen['valuse'] = ဆိုပြီး tuple တစ်ခုတည်ဆောက်ပြီး ကျွန်တော်တို့ ထည့်ချင်တဲ့ Data တွေထည့်ပါတယ်။ပြီးရင် grid() နဲ့နေရာချတယ်။nameChose.current() ကတော့ ဘယ်ကနေစမယ်လို့ပြောတာပါ။ဒီထဲမှာ 0 ထည့်ထားတော့ tuple ရဲ့ index 0 ဖြစ်တဲ့ Devid ကစမယ်လို့ပြောတာပါ။နောက်ထပ် widget တစ်ခုကတော့ button ဖြစ်ပါတယ် အဲဒီ Button ကိုနှိပ်ရင် click_1 ဆိုတဲ့ function ကိုခေါ်တယ်။အဲဒီ function ထဲမှာကတော့ msgbox._show() နဲ့ nameChosen.get() ကိုအသုံးပြုပြီး Combobox ကရွေးလိုက်တဲ့ နာမည်ကို messagebox မှာဖော်ပြမှာဖြစ်ပါတယ်။

Check Button with different initial states

အခုပြောမှာကတော့ Check Button တွေ အကြောင်းဖြစ်ပါတယ်။Check Button တွေတည်ဆောက်မယ် ပြီးရင် သူတို့ရဲ့ different မတူညီတဲ့ states တွေကိုဖော်ပြသွားမှာဖြစ်ပါတယ်။Check Button ကိုအသုံး ပြုဖို့ဆိုရင် IntVar() လို့ခေါ်တဲ့ method တစ်ခုက အရေးကြီးသလို variable လို့ခေါ်တဲ့ widget ရဲ့ Option တစ်ခုကလည်းအရေးကြီးပါတယ်။ဒါဆိုရင် ဘယ်လိုသုံးလဲ ဆိုတာ program နဲ့တွဲကြည့်ရအောင်။

```
from tkinter import *
from tkinter import ttk as tk,messagebox as msgbox

def click_1():
    if check_button_2.get() == 1:
        msgbox._show("Combobox Chosen","Button 2 is Checked")
    elif check_button_2.get() ==0:
        msgbox._show("Combobox Chosen","Button 2 is Unchecked")

if __name__ == "__main__":
    win = Tk()
    check_button_2 = IntVar()
    c_b1 = Checkbutton(win,text='Button_1',state='disabled')
    c_b1.grid(column=0,row=0)
    c_b2 = Checkbutton(win, text='Button_2',variable=check_button_2)
    c_b2.grid(column=2, row=0)
    c_b3 = Checkbutton(win, text='Button_3')
    c_b3.grid(column=3, row=0)
    Button(win, text="Combo", command=click_1).grid(column=1, row=0)
    win.mainloop()
```

ကျွန်တော်တို့ အရင်ဆုံး import လုပ်သင့်တာတွေလုပ် GUI တည်ဆောက် အဲဒီထဲမှာ object တစ်ခုတည်ဆောက်ပြီး check button method ကိုခေါ်တယ် အဲဒီထဲမှာ variable လို့ခေါ်တဲ့ option တစ်ခု ကို ထည့် ပြီး variable နာမည်တစ်ခုပေးတယ် ပြီးတော့ ကျွန်တော်တို့က အဲဒီ variable ကို Intvar() ဖြစ်ကြောင်း ပြောပေးပါတယ်။ဒါကဘာလုပ်ဖို့လဲဆိုတော့ ဒီ check button မှာ check လုပ်ရင် Int အနေနဲ့ 1 ဖြစ်

ပြီးတော့ Uncheck ဆိုရင်တော့ 0 ဖြစ်မှာဖြစ်ပါတယ်။ဒီတော့ variable နဲ့ IntVar() က check လုပ်တာ မလုပ်တာ ကို စစ်ဆေးပေးဖို့အတွက်မဖြစ်မနေလိုတဲ့လုပ်ငန်းစဉ်ဖြစ်ပါတယ်။ပြီးတော့ တည်ဆောက်ထား တဲ့ variable ကို .get နဲ့ရယူစစ်ဆေးနိုင်ပါတယ် အပေါ်မှာ check_button_2.get() == 1: == 0: စတာနဲ့ စစ် ဆေးပါတယ်။ဒါဆိုရင် Check Button အလုပ်လုပ်ပုံကိုလည်းသိရပါပြီး တခြား widgets တွေစီသွားရအောင်။

Using radio button with Change Form Background Color

ဒီတစ်ခေါက် Radio Button အသုံးပြုရင်းနဲ့ Background Colors တွေပါပြောင်းတဲ့ အလုပ်ကို လုပ်သွား ပါ့မယ်။Radio Button အသုံးပြုပုံကလည်း Check Button နဲ့မခြားပါဘူး။variable ကိုအသုံးပြုရသလို .get() နဲ့ဘဲရယူနိုင်ပါတယ်။တစ်ခုပိုပါလာတာကတော့ value လို့ခေါ်တဲ့ option ဘဲဖြစ်ပါတယ်။

```
from tkinter import *
from tkinter import ttk as tk

def click_1():
    var_ch = rechr.get()
    if var_ch == 1: win.config(bg='blue')
    elif var_ch == 2: win.config(bg='red')
    elif var_ch == 3: win.config(bg='green')



if __name__ == "__main__":
    win = Tk()
    win.geometry("500x500")
    win.resizable(0,0)

    rechr = IntVar()
    rdo1 =tk.Radiobutton(win,text="Blue",variable=rechr,value=1,command=click_1)
    rdo1.grid(column=0,row=0)


    rdo2 = tk.Radiobutton(win, text="Red",variable=rechr,value=2,command=click_1)
    rdo2.grid(column=1, row=0)


    rdo3 =tk.Radiobutton(win, text="Green",variable=rechr,value=3,command=click_1)
    rdo3.grid(column=2, row=0)

    win.mainloop()
```

Radio Button တွေအားလုံးကို variable = rechr ဆိုပြီး variable တမျိုးထဲပေးထားပါတယ်။value ကတော့ ကွဲ ပြားပါတယ်။value =1 , 2 , 3 ဖြစ်ပါတယ်။command = click_1 ဆိုတဲ့ method တစ်ခုထဲကိုဘဲ ခေါ်ထား ပါတယ်။Radio Button တွေက variable ကို .IntVar() နဲ့ဘဲ Int ရယူပါတယ်။ပြီးတော့ .get() နဲ့ရယူပြီး if နဲ့ စစ်ဆေးပါတယ်။အဲဒီထဲမှာ if 1 ဝင်လာရင်လုပ်မယ့် အလုပ်က win.config(bg= ' red') လို့ရေးတဲ့အတွက် bg ဆိုတာ background လို့ပြောတာပါ win ဆိုတာ Form ဖြစ်ပါတယ်။အဲဒီတော့ကျွန်တော်တို့ config လုပ် တဲ့အတိုင်းအရောင်ပြောင်းလဲမှာဖြစ်ပါတယ်။အော်...ပြောဖို့တစ်ခုမေ့နေတာ tk မသုံးဘဲ ဒီအတိုင်း

radiobutton သုံးရင်သိပ်မလှပါဘူး။tk.Radiobutton သုံးတဲ့အခါအောက်ပါအတိုင်းဘဲပိုမိုက်တာလေး သုံး နိုင်ပါတယ်။

Using Scrolled Text Widgets

Scrolled Text ဆိုတာ Text View Widget တစ်ခုပါဘဲ။သာမာန် Entry နဲ့ကွာခြားတာတစ်ခုကတော့ သူထက်ကျော်လွန်တဲ့ပမာဏ တစ်ခုကိုရောက်သွားတဲ့ scroll လုပ်နိုင်တဲ့ အပေါ်အောက်ဆွဲနိုင်တဲ့ scroll bar တစ်ခုပေါ်လာပါတယ်။

```
from tkinter import *
from tkinter import ttk as tk,scrolledtext
```

ကျွန်တော်တို့ tkinter ထဲက scrolledtext ကို import လုပ်ရပါမယ်။ဒါမှသုံးလို့ရနိုင်မှာဖြစ်ပါတယ်။

```
s_text = scrolledtext.ScrolledText(win, width=50, height=10, wrap=WORD)
s_text.grid(column=0,row=1,pady=10,padx=50)
```

အထက်ပါ အတိုင်းအသုံးပြုရပါတယ် scrolledtext.ScrolledText(win,width=50,height=10) အဖြစ်သုံးရပါတယ်။width က အကျယ် ဖြစ်ပြီး height က အမြင့်ပါ။wrap = WORD လို့အသုံးပြုရပါတယ်။

Label Frame Widget

Label Frame ကို GUI တွေကိုအလှဆင်တည်ဆောက်ဖို့အတွက်အသုံးပြုနိုင်ပါတယ်။အခုလောလောဆယ် ကျွန်တော်တို့က grid နဲ့ fram ကို အသုံးပြုပြီး အလှစဉ်နေရာချနေကြပါတယ် ဒီ LabelFrame ကို အသုံး ပြုပြီး GUI တည်ဆောက်ရင် ပိုပြီးအထောက်ကူပြုမယ့်အားသာချက်တွေရှိနိုင်ပါတယ်။ဒါဆိုရင်စပြီး တည်ဆောက်ကြည့်ရအောင်။

```
label_frame = tk.LabelFrame(win,text='Label Group')
label_frame.grid(column=0,row=0,padx=5,pady=5)

lab1 = tk.Label(label_frame,text='Label One')
lab1.grid(column=0,row=0)

lab2 = tk.Label(label_frame,text='Label Two')
lab2.grid(column=1,row=0)

lab3 = tk.Label(label_frame,text='Label Three')
lab3.grid(column=2,row=0)
```

Frame ကဲ့သို့ဘဲ သူ့ကိုနေရာချပြီး သူ့ရဲ့ object ကို Label တွေမှာထည့်သွင်းအသုံး ပြုရမှာ ဖြစ်ပါတယ်။ ကျွန်တော်တို့ win ဖြစ်တဲ့ win Form ပေါ်ကို label_frame ထပ်ပြီး အဲဒီ LabelFrame အပေါ်ကို တခြားသော Label တွေထည့်သွင်းအသုံးပြုမှာဖြစ်ပါတယ်။

Menu and Windows Form Image

အခုသင်ခန်းစာကတော့ new , edit , view စတဲ့ Tabs တွေထည့်နိုင်တဲ့ Menu နဲ့ Form Image ငှက်တောင်မွေးအပြာလေးနေရာမှာ ကျွန်တော်တို့စိတ်ကြိုက် icon လေးထည့်ဖို့ဘဲလိုပါတယ်။အဲ.. နောက် တစ်ခုပြောမှာရှိပါသေးတယ် os module ထဲက os.getcwd() method အသုံးပြုမှုဘဲဖြစ်ပါတယ်။

```
from tkinter import *
from tkinter import messagebox as msgbox,sys
import os
print(os.getcwd())

def click_menu():
    msgbox._show("Show Menu","I am New Menu")

def exit_click():
    sys.exit()

win = Tk()
win.title("Menu Tutorial")
win.iconbitmap(os.getcwd()+r'/CD.ico')

menubar = Menu(win)
file_menu = Menu(menubar,tearoff=1)
menubar.add_cascade(label='File',menu=file_menu)
edit_menu = Menu(menubar,tearoff=0)
menubar.add_cascade(label='Edit',menu=edit_menu)

file_menu.add_command(label="New",accelerator='Ctrl +
N',compound='left',command=click_menu)
file_menu.add_command(label="Open",accelerator='Ctrl +
N',compound='left',command=click_menu)
file_menu.add_command(label="Save",accelerator='Ctrl +
N',compound='left',command=click_menu)
file_menu.add_command(label="Save as",accelerator='Ctrl +
N',compound='left',command=click_menu)
file_menu.add_command(label="Exit",accelerator='Ctrl +
N',compound='left',command=exit_click)
win.config(menu=menubar)
win.mainloop()
```

အထက်က code ကအပြည့်စုံဘဲဖြစ်ပါတယ်။ဒီ code အလုပ်လုပ်ပုံက Windows Form လေးတစ်ခု ပေါ်မှာ File နဲ့ Edit Menu ပေါ်မှာ Tabs နှစ်ခုရှိနေပါတယ်။File ထဲမှာ New Open Save Save as Exit ဆိုတဲ့ Drop Down Menu ပါဝင်ပါတယ်။အဲဒီ Menu ထဲမှာ အခြားသော New Open Save .. တွေကို ကလစ် လုပ်တဲ့အခါမှာ Messagebox တစ်ခုကြွလာပြီး Message ပေါ်လာမှာဖြစ်ပါတယ်။အဲလို Message ပေါ်ဖို့ Function တည်ဆောက် ပြီး အဲဒီ Function ကိုခေါ်သုံးတာဖြစ်ပါတယ်။နောက်တစ်ခုက Exit ကိုနှိပ်တဲ့ အခါမှာ exit function ကိုခေါ်ပြီး Application ပိတ်သွားမှာဖြစ်ပါတယ်။

```
win.iconbitmap(os.getcwd()+r'/CD.ico')
```

ဒီအပေါ်က method လေးက iconbitmap() ကတော့ Windows From မှာ Image ထည့်ဖို့ဖြစ်ပါတယ်။ဒါပေမယ့် Image Format အနေနဲ့ .ico ဆိုတဲ့ Format တစ်ခုကိုဘဲလက်ခံနိုင်ပါတယ်။

```
import os
```

os module ကို import လုပ်ထားတာဖြစ်ပါတယ်။ဘာကြောင့်လဲဆိုတော့ os module က Folder တွေတည် ဆောက်တာ os မှာ Folder တွေ remove လုပ်တာနဲ့ Folder နေရာတွေလမ်းကြောင်းတွေ ကိုရယူတာစတာတွေလုပ်နိုင်ပါတယ်။ဒီတော့ ကျွန်တော်က အခု CD.ico ဆိုတဲ့ဖိုင်လေးကို ဒီ program folder ထဲမှာဘဲထည့်ထားပါတယ် အဲဒီ program ထဲက နေရာလမ်းကြောင်းကိုသွားဖို့အတွက် os.getcwd() ကိုအသုံး ပြုရပါတယ်။r '/CD.ico' ဆိုတာကတော့ CD.ico ရှိတဲ့နေရာကိုထပ်ယူတာဖြစ်ပါတယ်။

```
from tkinter import messagebox as msgbox,sys
```

ဒီမှာ import လုပ်ထားတာ messagebox ကတော့ အထူးဆန်းမဟုတ်ပေမယ့် sys ကတော့ ထူးနေမှာပါ sys က system module တစ်ခုဖြစ်ပါတယ်။exit() ဆိုတဲ့ Application ပိတ်တဲ့ Method က sys module ထဲမှာ ဘဲရှိပါတယ်။အဲဒီတော့ Application ပိတ်ဖို့အတွက် sys ကိုသုံးတာဖြစ်ပါတယ်။

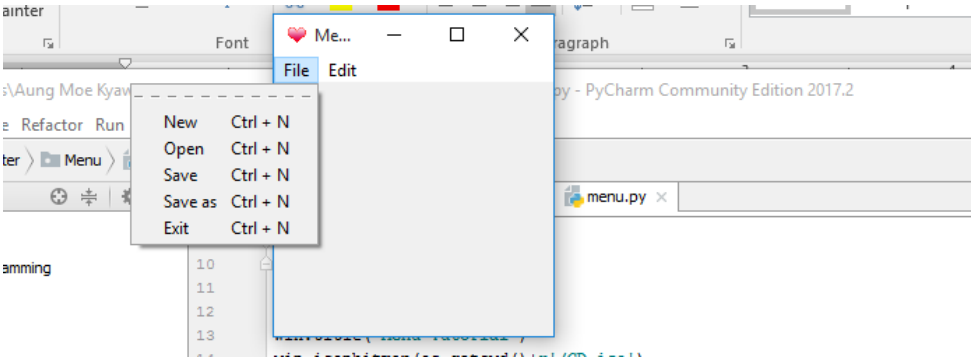
```
menubar = Menu(win)
file_menu = Menu(menubar,tearoff=1)
menubar.add_cascade(label='File',menu=file_menu)
edit_menu = Menu(menubar,tearoff=0)
menubar.add_cascade(label='Edit',menu=edit_menu)

file_menu.add_command(label="New",accelerator='Ctrl +
N',compound='left',command=click_menu)
win.config(menu=menubar)
```

အထက်ပါ method များ code များက menu တည်ဆောက်ဖို့သုံးတဲ့ code တွေဖြစ်ပါတယ်။အရင်ဆုံး object တစ်ခုကိုတည်ဆောက်ပြီး Menu() method ခေါ်ပါတယ်ပြီးတော့ parent အဖြစ် win ကို inheritance လုပ်ပြီး win ပေါ်တင်လိုက်ပါတယ်။နောက် object တစ်ခုတည်ဆောက်ပြီးတော့ Menu method ထဲကိုဘဲ စောစောက win ပေါ်တင်ထားတဲ့ object ကို parent ထဲထည့်ပြီး menu အဖြစ်ကြေညာလိုက်ပါတယ်။win ပေါ်တင်ထားတဲ့ object ကို .add_cascade() method နဲ့ label ထဲမှာ menu မှာပေါ်စေချင်တဲ့ Name ထည့်ပြီး menu file object ကို ထည့်ရပါတယ်။tearoff =1 ကတော့ --- လေးတွေပေါ်စေပါတယ် အဲဒါကို separator လို့ ခေါ်ပါတယ်။0 ထားရင်တော့ dot လေးတွေပျောက်သွားပါလိမ့်မယ် ဒါပေမယ့် ဒီ function က multi os cross supported တော့မဟုတ်ပါဘူး တချို့သော Linux Os တွေမှာအလုပ်မလုပ်တတ်ပါဘူး။အထက်ပါလုပ်ဆောင်

ချက်အားလုံးက Tabs တွေတည်ဆောက်မှု့ဘဲဖြစ်ပါတယ်။တကယ်နှိပ်လို့ရတဲ့ Drop Down List တွေကို ဖန်တီးဖို့အတွက်က

```
file_menu.add_command(label="New",accelerator='Ctrl + N',command=click_menu)
```

File Menu ထဲကို .add_command() method နဲ့ထည့်ရပါတယ်။အဲဒီမှာ command နဲ့ button တွေလိုဘဲ function တွေခေါ်နိုင်တဲ့အလုပ်ကိုလုပ်ပေးပါတယ်။



Tkinter Tab Widget

Windows Form ထဲမှာဘဲ Tabs တွေခွဲပြီး Form အမျိုးမျိုးနဲ့ control widgets တွေထည့်သွင်းနိုင်ပါတယ်။ဒါကိုလုပ်ကြည့်ဖို့ Tab တွေတည်ဆောက်ရအောင်။

```
from tkinter import *
from tkinter import ttk as tk

win = Tk()
win.geometry("300x300")
tabcontrol = tk.Notebook(win)

tab1 = tk.Frame(tabcontrol)
tabcontrol.add(tab1,text='Tab 1')
tabcontrol.grid()

btn = tk.Button(tab1,text='Button1')
btn.grid()

tab2 = tk.Frame(tabcontrol)
tabcontrol.add(tab2,text='Tab 2')
tabcontrol.grid()

btn = tk.Button(tab2,text='Button2')
btn.grid()

tab3 = tk.Frame(tabcontrol)
tabcontrol.add(tab3,text='Tab 2')
tabcontrol.grid()

btn = tk.Button(tab3,text='Button3')
btn.grid()
```

```
tabcontrol.pack(expand=1,fill='both')
win.mainloop()
```

ဒါကတော့ Code အပြည့်အစုံဖြစ်ပါတယ်။ကဲရှင်းလိုက်ရအောင်။

```
tabcontrol = tk.Notebook(win)
```

ဒီ code ကတော့ object တစ်ခုတည်ဆောက်ပြီး Tab function သုံးဖို့အတွက် tk ထဲက Notebook() method ကိုခေါ်လိုက်တာပါဘဲ win ကိုတော့ parent အဖြစ်ထားပါတယ်။

အဲဒီနောက်မှာ

```
tab1 = tk.Frame(tabcontrol)
tabcontrol.add(tab1,text='Tab 1')
tabcontrol.grid()
```

tab1 ဆိုတဲ့ object ကိုတည်ဆောက်ပြီး tk.Frame() ထဲမှာ tabcontrol ကိုparent ထားပါတယ်။နောက်တော့ tab ထည့်ဖို့အတွက် tabcontrol.add() နဲ့ Frame ရော text ရောထည့်ပါတယ်။ပြီးတော့နေရာချတယ် tabcontrol.grid() နဲ့ပါ။ဒါမျိုးအများကြီးတည်ဆောက်နိုင်ပါတယ်။

```
btn = tk.Button(tab1,text='Button1')
btn.grid()
```

ကတော့ tab တစ်ခုခြင်းစီရဲ့အထဲမှာ Button , Label အားလုံးထည့်နိုင်တဲ့အရာဖြစ်ပါတယ်။နောက်တစ်ခုအရေးကြီးတာက Windows Form အတိုင်း Tab ရဲ့ပမာဏဖြစ်အောင် pack() method ကိုအသုံးပြုရမှာပါ။

```
tabcontrol.pack(expand=1,fill='both')
```

expand=1,fill= 'both' ကိုသုံးပြီး Windows Form အပြည့်သုံးနိုင်မှာပါ။

Using a spin box control

Spin box ဆိုတာကတော့ ကျွန်တော်တို့ထည့်ချင်တဲ့တန်ဖိုးတွေကို generate လုပ်ပေးတဲ့ Box လေးတစ်ခုဖြစ်ပါတယ်။သုံးရတာလွယ်ကူပြီး အရမ်းအသုံးဝင်ပါတယ်။ဘယ်လိုတည်ဆောက်လဲကြည့်ရအောင်။

```
from tkinter import *

def _spin():
    value = spin.get()
    print(value)


win = Tk()
win.geometry("300x300")
spin = Spinbox(win,from_=0,to=10,command= _spin)
spin.grid(column=0,row=0)
win.mainloop()
```

အထက်ကတော့ code အပြည့်စုံဘဲဖြစ်ပါတယ်။

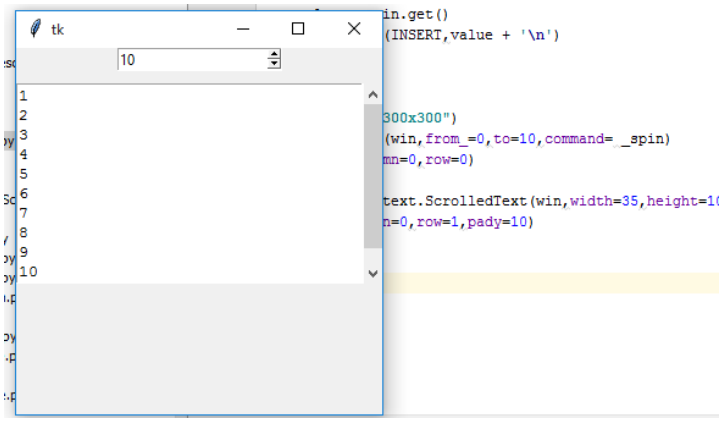
```
spin = Spinbox(win,from_=0,to=10,command= _spin)
```

အခြားသော widgets တွေကဲ့သို့ဘဲ object တစ်ခုတည်ဆောက်ပြီး Spinbox() method ခေါ်တည်ဆောက်ရပါတယ်။from _=0 ဆိုတာ 0 ကနေစမယ်လို့ပြောပြီး to = 10 ထိလို့ပြောတာဖြစ်ပါတယ်။command ကတော့ method call တာဖြစ်ပါတယ်။spinbox ထဲက value တွေကိုတော့ get() နဲ့ဘဲရယူပါတယ်။

ဒါဆိုရင် ကျွန်တော်တို့ Scroll Text ထဲကို ကျွန်တော်တို့ Spin Box ထဲက Datas တွေဘယ်လိုထည့်မလဲကြည့် ရအောင်။

```
from tkinter import *
from tkinter import scrolledtext
def _spin():
    value = spin.get()
    scr.insert(INSERT,value + '\n')


win = Tk()
win.geometry("300x300")
spin = Spinbox(win,from_=0,to=10,command= _spin)
spin.grid(column=0,row=0)

scr = scrolledtext.ScrolledText(win,width=35,height=10,wrap=WORD)
scr.grid(column=0,row=1,pady=10)

win.mainloop()
```

sipn box နဲ့ scrolledtext နဲ့တည်ဆောက်မယ် ပြီးရင်ကျွန်တော်တို့ spin box value ကို spin.get() ကိုဖမ်းယူပြီးတော့ value ထဲထည့်မယ် အဲဒီ value ကို scrolledtext object ထဲကို insert() method နဲ့ ထည့်သွင်းရပါတယ်။ဘေးတိုက်မသွားဘဲ တစ်ကြောင်းခြင်းဆင်းသွားချင်တဲ့အတွက် value + '\n' ကိုသုံးပါတယ်။



နောက်ထပ် spinbox ထဲကို Data ထည့်ချင်ရင် spinbox object ကိုခေါ်ပြီး spinbox_object['value'] = () နဲ့ tuple အနေနဲ့ ထည့်သွင်းနိုင်ပါတယ်။

Passing arguments to callbacks

ကျွန်တော်တို့ widgets တွေနောက်မှာ command နဲ့ချိတ်ဆက်ပြီး functions တွေကိုခေါ်တာကို callbacks လုပ်တယ်လို့ခေါ်ပါတယ်။ရှေ့မှာ command နဲ့ callbacks ကိုအသုံးပြုတဲ့အခါမှာ simple function တွေကိုဘဲပြောရရင် arguments တွေမပါတဲ့ Functions တွေကိုခေါ်ခဲ့ကြတာဖြစ်ပါတယ်။သာမန် callbacks တွေက function မှာ Argument ထည့်ရင်သုံ့အလိုလိုအလုပ်တွေလုပ်နေပြီးပြဿနာတွေ တတ်နေပါတယ်။ဒီလိုမျိုးမဖြစ်အောင် Argument တွေ Passing လုပ်တဲ့ အခါမှာ lambda ကိုသုံးရပါတယ်။ဒါဆို လက် တွေ့လုပ်ကြည့်ရအောင်။

```
from tkinter import *
from tkinter import messagebox as msgbox


def call_backs(name):
    msgbox._show('Call_Backs With Arguments','Hello '+name)


win = Tk()
win.title("Call Backs With Arguments")
win.resizable(0,0)
win.geometry("200x200")
btn=Button(win,text="Press",command=lambda:(call_backs("Myanmar Developer!")))
btn.grid()
win.mainloop()
```

call_backs(name) ဆိုတဲ့ function ထဲမှာ Argument အနေနဲ့ name ကိုထည့်ထားပါတယ်။အဲဒီ ထဲမှာ လုပ်မယ့်အလုပ်က msgbox._show() နဲ့ Hello + name လို့ခေါ်ထားတယ်။function ကိုဖြတ်ပြီးဂင်လာမယ့် name နဲ့ Hello ကို တွဲအဖြေထုတ်မယ်လို့ပြောထားတာပါ။

command ကိုခေါ်တဲ့အခါမှာ ရိုးရိုးမခေါ်ဘဲနဲ့ command=lambda:(function(Argument)) ဆိုတဲ့ ပုံစံနဲ့ ခေါ်ပါတယ်။

Event Binding

Tkinter application တွေကို Loop အတွင်းမှာတည်ဆောက်ထားတာဖြစ်ပါတယ်။ဘယ်လို လဲဆိုတော့ mainloop ထဲမှာ တည်ဆောက်ထားတာပါ။အဲဒီတော့ဒီ Application တွေက Events တွေဘာဖြစ်လာမလဲဆိုတာလဲစောင့်ကြည့်နိုင်ပါတယ်။Events ဆိုတာကတော့ Key Board က ဘယ်ခလုပ် နှိပ်မလဲစောင့်ကြည့်တာ Mouse ကနေဘယ်နေရာကိုနှိပ်မလဲဆိုတာ စောင့်ကြည့်တုန့်ပြန်နိုင်တဲ့လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။

Tkinter က programmer ကို events တွေနဲ့အလုပ်လုပ်နိုင်အောင် အထောက်ပံ့ပေးထားပါတယ်။ Widget ကို bind ဆိုတဲ့ Python Function ကိုသုံးပြီး event ဖမ်းယူနိုင်ပါတယ်။

widget.bind(event,handler)

အထက်ပါအတိုင်း widget ကို event လို့သတ်မှတ်လိုက်ရင် အကျိုးသက်ရောက်မှုက အဲဒီ define လုပ်ထားတဲ့ widget ပေါ်မှာသက်ရောက်ပါတယ်။Handler ဆိုတာကတော့ event object ဘဲဖြစ်ပါတယ်။ဒါဆိုလက် တွေ့ရေးကြည့်ရအောင်။

```
from tkinter import *
import sys

win = Tk()

def hello(event):
    print("Single Click,Button-I")
def quit(event):
    print("Double Click,so let's stop")
    sys.exit()

label = Label(win,text="Click A Text!!!")
label.pack()
label.bind('<Button-1>',hello)
label.bind('<Double-1>',quit)

win.mainloop()
```

functions ၂ ခုတည်ဆောက်ပါတယ် အဲဒီမှာ event object ကို ဖမ်းထားတယ် ပြီးတော့ လုပ်မယ့် အလုပ်တွေ ရေးတယ်ပေါ့။နောက်ပြီး Label တစ်ခုတည်ဆောက်တယ် label object ကို bind() နဲ့ တွဲတယ် ဘာကိုဖမ်းမှာလဲဆိုတော့ <Button-1> ပေါ့ တစ်ချက်နှိပ်ရင် hello ဆိုတဲ့ event function ကိုခေါ်မယ်။<Double-1> ဆိုတော့ Double Click လုပ်ရင် quit ကိုခေါ်မယ်လို့ပြောလိုက်တာပါ။ရေးကြည့်ပါအုန်း။ဒါဆိုနောက် ထပ်တစ်ခုထပ်ကြည့်ရအောင်။

```
from tkinter import *

win = Tk()
def motion(event):
    print('Mouse Posistion:(%s,%s)'%(event.x,event.y))
    return

msg = Message(win,text="what ever you do")
msg.config(bg='lightgreen',font=('times',24,'italic'))
msg.bind('<Motion>',motion)
msg.pack()
win.mainloop()
```

ဒီမှာတော့ msg ပေါ်ကို Mouse ရွှေ့တဲ့အခါမှာ positions ပြချင်လို့ <Motion> ကို ဖမ်းထားပါတယ်။ Motion နဲ့ တွဲဖမ်းတဲ့ event object လုပ်ဆောင်ချက်ကတော့ event.x , event.y ဖြစ်ပါတယ် event.x က x ဝင်ရိုးအတိုင်းသွားတဲ့ position ပေါ့။event.y ကတော့ y ဝင်ရိုးအတိုင်းသွားတာဖြစ်ပါတယ်။

Event Types

Event Types အများကြီးရှိပါတယ်။အပေါ်မှာက <Button-1> ,<Double-1>,<Motion> တို့ကို ဘဲအသုံးပြုပြခဲ့ပါတယ်တကယ်တော့ အများကြီးရှိတာပါ။

Event Types	Event Description
<Button>	ဒါက Mouse Button Event ကိုဖမ်းတာပါ Left Click Mouse ဆိုရင် <Button-1> ကိုအသုံးပြုရပြီး Middle Button ဆို <Button-2> Right Button ဆို ရင် <Button-3> နဲ့ဖမ်းယူရပါတယ်။<Button-4> က scroll Up ဖြစ်ပြီး <Button-5> ကတော့ Scroll Down ဖြစ်ပါတယ်။
<Motion>	Mouse ရွှေ့လျားမှု့ကို ဖမ်းဖို့အတွက်သုံးတာဖြစ်ပါတယ်။Left , middle , right ကို <B1-Motion> <B2-Motion> <B3-Motion> ဆိုပြီးသုံးခုနဲ့ ဖမ်းနိုင်တယ် သူ့ရဲ့ current positions တွေကို event.x , event.y ဆိုပြီး ဖမ်းယူနိုင်ပါတယ်။
<ButtonRelease>	Event x,y တွေကို Lef Middle,Right ကို <ButtoRelease-1>, <ButtonRelease-2>, <ButtonRelease-3> ဆိုပြီးသုံးခုနဲ့လည်း ဖမ်း ယူ နိုင် ပါတယ်။
<Double-Button>	<Button> လုပ်ဆောင်ချက်နဲ့တူပါတယ် ဒါပေမယ့် ဒါက Double Click လုပ်မှ အလုပ်လုပ်မယ့် လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။ Left, Middle , Right ကို အရင်လိုဘဲ <Double-Button-1>2,3 နဲ့သတ်မှတ်ပါတယ်။<Button-1> နှစ်ခါဆက်တိုက်နှိပ်တာက <Double-Button-1> နဲ့အတူတူပါဘဲ။
<Enter>	ဒါက Enter Key နှိပ်တာကိုပြောတာမဟုတ်ဘူးနော် Mouse Pointer ကို Widget အတွင်းကိုထည့်လိုက်တာကိုပြောတာပါ။Keyboard က Enter Key နှိပ်တာကိုပြောတာမဟုတ်ပါဘူး။
<Return>	ဒီ Return ကမှ Keyboard က Enter Key နှိပ်တာကို အလုပ်လုပ်တဲ့ Function ဖြစ်ပါတယ်။Cancel(break), BackSpace, Tab,Return(Enter), Shift_L, Alt_L, Pause, Caps_Lock, Escape, Prior(page up) Next (Page Down) End,Home,Left,Up,Right,Down,Print,Insert,Delete,F1 to F12 …. အကုန်အသုံးပြုနိုင်ပါတယ်။
<keyPress>	ဒီ KeyPress ကတော့ char တွေကို passing လုပ်ပြီးတန်ဖိုးဖမ်းပေး ပါတယ် <KeyPress-b> ဆိုရင် b အသေးကို Keyboard က ရိုက်တဲ့အခါမှာ အလုပ် လုပ်မှာဖြစ်ပါတယ်။

<Leave>	Mouse Pointer တွေ widgets ပေါ်က ထွက်သွားရင်လုပ်မယ့် အလုပ်ဖြစ်ပါတယ်။
<FocusIn>	Widgets တွေရဲ့ child တွေကိုရွေးချယ်ဖို့တွက်ဖြစ်ပါတယ်။ <FocusOut> ကတော့ widget တစ်ခုကနေအခြား widget တစ်ခုကို ပြောင်းရင်အလုပ်လုပ်မယ့် လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။

Dialogbox with Tkinter

Dialogbox ဆိုတာအထူးဆန်းမဟုတ်ပါဘူး ကျွန်တော်တို့ဖိုင်တွေရွေးချယ်တဲ့အခါ သို့မဟုတ် Notepad မှာစာတွေရေးပြီး save ဖို့လုပ်တဲ့အခါ မှာသို့မဟုတ် Photoshop , Word , Excel စတာတွေမှာ ဖိုင် Open လုပ်တဲ့အခါ File Save လုပ်ဖို့နေရာရွေးတဲ့အခါမျိုးတွေမှာ သုံးတဲ့ Windows Explorer လို Box မျိုးကို ပြောတာပါ။ဘာအလုပ်လုပ်လဲဆိုရင် File နေရာလမ်းကြောင်းရွေးတာ File Save ဖို့လမ်းကြောင်းရွေးတာ မျိုးလုပ်ဖို့ပေါ့။

ဒါကိုသုံးဖို့အတွက် အရင်ဆုံး tkinter.filedialog ကို import လုပ်ရပါတယ်။သုံးရမယ့် Dialog အမျိုးအစား ၇ မျိုး ရှိပါတယ်။

askopenfile	File Object ကို return ပြန်ပေးတဲ့ dialog အမျိုးအစား
askopenfilename	File Name ဖြစ်တဲ့ String ကို retrun ပြန်ပေးပါတယ်။ဒါပေမယ့် File Object ကိုတော့ဖွင့်လို့ရမှာမဟုတ်ပါဘူး။
askopenfilenames	File Name တွေကို တစ်ခုထက်ပိုပြီး List လိုမျိုး Return ပြန်ပေးပါတယ်။
askopenfiles	File Object တွေကို တစ်ခုထက်ပိုပြီး List ပုံစံနဲ့ Return ပြန်ပေးတာမျိုးပါ။
asksaveasfile	ဒါကတော့ Save as လုပ်ဖို့ File name ကိုရစေပြီးတော့ File object ကို Return ပြန်တဲ့ အမျိုးအစားပါဘဲ။
askdirectory	Directory အမျိုးစား File တွေကိုနေရာပြနိုင်ပြီး Directory Name တွေကို return ရစေပါတယ်။

askopenfile နဲ့ asksaveasfile မှာတော့ mode တွေသုံးရပါတယ်။'r' သို့မဟုတ် 'w' ပေါ့ဗျာ။r ကတော့သိတဲ့အတိုင်း read ပေါ့ w ကတော့ write ပါ။file ကိုဖွင့်တဲ့ dialog ဖြစ်တဲ့ askopenfile က r သုံးပြီး file တွေ save တာဖြစ်တဲ့ asksaveasfile ကတော့ w သုံးပါတယ်။

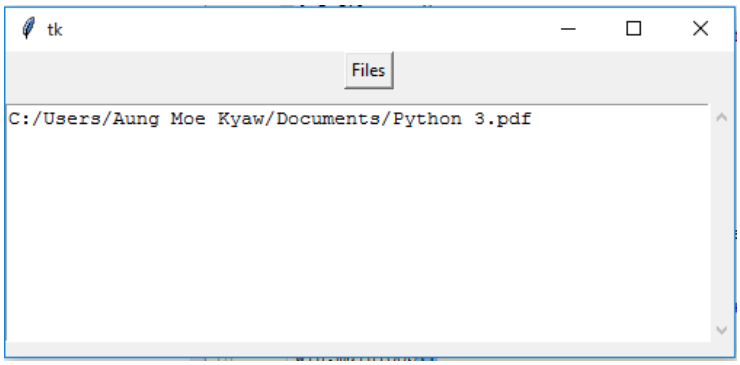
file_object = filedialog.askopenfile(mode='r')

file_object = filedialog.asksaveasfile(mode='w')

ပုံစံနဲ့သွားပါတယ်။အခြားသော dialog ၅ မျိုးမှာတော့ mode တွေမပါပါဘူး ဒါပေမယ့် parent,title,message ,defaultextension, filetypes, initialdir, initialfile, and multiple ကတော့ အားလုံးနီးပါးမှာပါပါတယ်။parent ဆိုတာကတော့ ကိုယ်တည်ဆောက်မယ့် Frame parent ပါ title ကတော့ dialogbox မှာပေါ်မယ့် title ပြီးတော့ message က ကိုယ်ပေးချင်တဲ့ message ပေးဖို့ defaultextension ကတော့ ကိုယ့်သတ်မှတ်ချင်တဲ့ file extension အတွက်ပါ။initialdir ကတော့ dialogbox ကိုဖွင့်လိုက်တဲ့အခါမှာ ဘယ် Directory က စမှာလဲ လို့သတ်မှတ်ပေးနိုင်တာဖြစ်ပါတယ်။ဒါကတော့အသုံးအများဆုံး options တွေကိုဖော်ပြလိုက်တာဖြစ်ပါတယ်။ဒါ ဆိုရင်စရေးကြည့်ရအောင် File ထားတဲ့လမ်းကြောင်းလေးကို ကျွန်တော်တို့ scrolltext ထဲမှာထည့်တဲ့ program လေးရေးပို့မယ်။

```
from tkinter import *
from tkinter import filedialog,scrolledtext

def fileopen():
    fi_open = filedialog.askopenfilename(defaultextension='.pdf',filetypes=[('All
File','*.*'),("PDF","*.pdf")])
    scr.insert(INSERT, fi_open + '\n')


win = Tk()
btn_d = Button(win,text="Files",command=fileopen)
btn_d.grid()
scr = scrolledtext.ScrolledText(win, width=60, height=10, wrap=WORD)
scr.grid(column=0, row=1, pady=10)
win.mainloop()
```

tkinter import filedialog,scrolledtext ဆိုပြီး module နှစ်ခုကို import လုပ်ပါတယ် ပြီးတော့ fileopen() ဆိုတဲ့ function တစ်ခုတည်ဆောက်တယ်။ပြီးတော့ fi_open ဆိုတဲ့ file object တစ်ခု တည်ဆောက် ပြီး askopenfilename() ကိုအသုံးပြုပြီး file object ကိုရယူထားတယ် defaultextension='.pdf ' လို့ ထည့် ထားပါတယ်။ကျွန်တော်တို့လက်ခံမယ့် defaultextension ပါ။ပြီးတော့ ဖွင့်လို့ရမယ့် filetypes မှာ All File ဆို ပြီး *.* ဆိုတာ ဖိုင် extension အားလုံးကိုဖွင့်နိုင်မယ့် All File ဖြစ်ပါတယ်။နောက်တစ်ခုကတော့ "PDF" , "*.pdf" လို့ရေးထားတာဖြစ်တဲ့အတွက် .pdf ဖိုင်တွေကလွဲရင်ဖွင့်လို့ရမှာမဟုတ်ပါဘူး။အဲဒီထဲမှာ လုပ်ထားတဲ့ အလုပ်က scrolledtext object ဖြစ်တဲ့ scr ထဲကို insert နဲ့ (INSERT,fi_open +'\n') ဖြစ်ပါတယ်။file object ကို ထည့်မယ် ပြီးရင် နောက်တစ်ကြောင်းဆင်းလို့ ရိုးရိုးရှင်းရှင်းပြောလိုက်တာမျိုးပါ။Button တစ်လုံးတည်ဆောက် ထားပြီး အဲဒီ Button ကိုနှိပ်ရင် fileopen() function ကိုခေါ်မှာဖြစ်ပါတယ်။



Tkinter Tree View Using ttk.Treeview

TreeView ဆိုတာ Scrolledtext လိုပါဘဲ Column လေးတွေနဲ့ဖွဲ့စည်းထားပြီးတော့ အဲဒီထဲမှာ တန်ဖိုးတွေထည့်နိုင် ကြည့်ရှုနိုင်တယ်။ပိုကောင်းတာ Scrolledtext နဲ့မတူတာက Item တွေကို Select လုပ်လို့ရပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့တည်ဆောက်ကြည့်ရအောင်။

```
from tkinter import *
from tkinter import ttk as gui

win = Tk()
win.geometry('500x300')
treeveiw = gui.Treeview(win,height=5,columns=2)
treeveiw.grid(row=4,column=0,columnspan=4)
treeveiw.heading("#0",text='No',anchor=W)
treeveiw.heading("#1",text='Name',anchor=W)
win.mainloop()
```

ဒါကတော့ TreeView တစ်ခုကိုတည်ဆောက်လိုက်တာဘဲဖြစ်ပါတယ်။ treeview ဆိုတဲ့ object ကိုတည်ဆောက်တယ်။ပြီးရင် Treeview ကိုခေါ်တယ် သူ့ထဲမှာ parent ဖြစ်တဲ့ win ပါတယ် height=5 သတ်မှတ်ပေးနိုင်တယ် နောက်ပြီး columns=2 ဆိုပြီး columns အရေတွက်ထည့်တယ်။ပြီးတော့ grid() နဲ့နေရာချတယ်ပေါ့ဗျာ။Columns ထည့်တာကတော့ heading() နဲ့ထည့်ပါတယ်။

TreeView ထဲကို Data ဘယ်လို့ထည့်လဲဆိုရင်တော့

```
treeveiw.insert('',0,text='1',values="KyawZayAung")
treeveiw.insert('',1,text='2',values="KyawKyawWin")
```

text နဲ့ values ဆိုပြီး နှစ်ခုထဲမှာထည့်နိုင်ပါတယ်။

TreeView ထဲမှာရှိတဲ့ Items တွေကိုဘယ်လိုယူလဲဆိုရင်တော့

```
items = treeveiw.get_children()
```

ဒါလေးကိုအသုံးပြုရပါတယ်။Tuple အနေနဲ့တန်ဖိုးရလာမှာဖြစ်ပါတယ်။ဒါကြောင့် items[] ဆိုပြီး index တန်ဖိုးတွေနဲ့ယူနိုင်ပါတယ်။treeview ထဲက items တွေကို Delete လုပ်ချင်တဲ့အခါမှာ ဒီ tuple တန်ဖိုးတွေကို အသုံးပြုပြီးလုပ်နိုင်ပါတယ်။

```
treeveiw.delete(items[0])
```

ဒီလိုဆိုအပေါ်ဆုံးဖြစ်မယ့် item 0 ကို Delete လုပ်သွားမှာဖြစ်ပါတယ်။ဒါဆိုရင် ကျွန်တော်တို့တော်တော်ပြည့်စုံသလောက်ဖြစ်သွားပြီထင်ပါတယ်။ကျွန်တော်တို့ Tkinter ကို OOP ပုံစံနဲ့ဘယ်တည်ဆောက်မလဲဆိုတာ ဆွေးနွေးသွားမယ်ပြီးရင် ကျွန်တော်တို့ Program လေးတချို့လက်တွေးရေးကြမှာဖြစ်ပါတယ်။

Data and Classes

ဒီအခန်းမှာတော့ tkinter variable တွေအသုံးပြုနည်းရယ် object-oriented programming example တွေရယ်ကိုဆွေးနွေးသွားပါ့မယ်။အရင်ကျွန်တော်တို့ StringVar() နဲ့ tkinter variable အကြောင်းကို အနည်းငယ်သိခဲ့ပြီးပါပြီ။အခုထပ်ပြောမယ်ပေါ့ဗျာ။အခုပြောမှာက tkinter GUI ကနေပြီး variable ထဲကို data တွေကို get တွေ set တွေသုံးပြီးဘယ်လိုထည့်မလဲဆိုတာပြောမှာပါ။Java မှာဆို getter setter နဲ့ ပုံစံတူပါတယ်။StringVar() လိုဘဲ တခြားသုံးလို့ရတဲ့က variable method တွေရှိပါတယ်။ဒါတွေကတော့

strVar = StringVar()	String Value တွေကိုယူဖို့အသုံးပြုပါတယ်။Default Value ကတော့ Empty ဖြစ်ပါတယ်။
intVar = IntVar()	Integer Value တွေအတွက်ဖြစ်ပါတယ်။default value က 0 ပါ။
dbVar =DoubleVar()	Float Value တွေအတွက်ဖြစ်ပါတယ်။default value က 0.0 ဖြစ်ပါတယ်။
blVar = BooleanVar()	True or False Boolean Value တွေအတွက်ဖြစ်ပါတယ်။0 က False ဖြစ်ပြီး 1 က True ဖြစ်ပါတယ်။

ဒါဆိုရင် String Variable တစ်ခုကို get , set သုံးပြီးယူတဲ့ လက်တွေ့ Program တစ်ခု ရေးပြီး ရှင်းပြ ပါမယ်။

```
import tkinter as tk
from tkinter import *
win = tk.Tk()
# Assign Tkinter Variable to strData variable
strData = tk.StringVar()

# Set strData variable
strData.set('Hello StringVar')

#get strData variable
getData = strData.get()

print(getData)
```

Tkinter ကနေ variable တွေယူပုံပြမှာမို့ tkinter ကို import လုပ်ထားတယ် tk ဆိုတဲ့ နာမည်နဲကပေါ့။ပြီးတော့ win ဆိုပြီး tk.Tk() ဖြစ်ပါတယ်။ဒါက Tk method ကိုခေါ်လိုက်တာဖြစ်ပါတယ်။ဒီ program မှာ tkinter object ကို အသုံးပြုပေမယ့် GUI Form တော့မတည်ဆောက်ပါဘူး။strData ဆိုတဲ့ variable ထဲကို tk.StringVar() နဲ့ String သိုလှောင်မယ်ပြောလိုက်တယ်။ပြီးတော့ strData.set() ဆိုပြီး set() method ကိုအသုံးပြုပြီး Hello StringVar ဆိုတဲ့ String စာသား ကို ထည့်လိုက်တယ်ပြီးတော့ getData ဆိုတဲ့ variable ထဲကို strData.get() နဲ့ data ရယူလိုက်ပါတယ်။ပြီးတော့ print နဲ့ထုတ်ကြည့်တဲ့အခါမှာ ကျွန်တော်တို့ထည့်လိုက်တဲ့ String ကိုပြန်တွေ့မှာဖြစ်ပါတယ်။

```
# Assign Tkinter Variable to strData variable
strData = tk.StringVar()

#Print tk.StringVar() After set value
print(strData)
```

ဒီ tk.Stringvar() ကို ဒီအတိုင်း print ထုတ်မယ်ဆိုရင်တော့ PY_VAR0 ဆိုပြီးအဖြေထွက်ပါတယ်။ဒီက ဘာအမျိုးစားလဲသိချင်ရင် type() နဲ့စစ်ကြည့်တာပေါ့ဗျာ။

```
print(type(strData))
```

ဒါဆိုရင် tkinter.StringVar ဆိုတာကိုတွေ့ရမှာပါ။ဒါကအထူး return ပြန်တဲ့ပုံစံတစ်ခုပေါ့။ဒါဆိုရင် get() နဲ့ တွဲယူကြည့်ရင်တော့ လက်တွေ့ဆန်တဲ့ Type အမျိုးစားထွက်လာမှာပါ။

```
# Assign Tkinter Variable to strData variable
strData = tk.StringVar()

#Print tk.StringVar() After set value
print(strData.get())
print(type(strData.get()))
```

ဘာမှမပါတဲ့ Empty အဖြေထွက်လာမယ် Type မှာ str လို့အဖြေထွက်လာပါတယ်။ဒါကိုကြည့်ခြင်းအားဖြင့် StringVar() လို method တွေက .get() .set() နဲ့တွဲဖက်သုံးရပါတယ်။

How to get Data From widgets

Widgets တွေစီကနေ StringVar() IntVar() တွေယူချင်ရင်ဘယ်လိုယူရလဲဆိုတာ ဆွေးနွေးသွားမယ်။ widgets တွေစီကနေ get() နဲ့ရယူနိုင်ပါတယ်။အောက်မှာ နမူနာရေးထားတဲ့ program ကိုလေ့လာကြည့်လိုက်ပါ။

```
from tkinter import *

win = Tk()
def call_back():
    strData = spin.get()
```

```
    print(strData)

spin = Spinbox(win,value=(1,2,3,4,5),command=call_back)
spin.grid()


win.mainloop()
```

Global and Local Variable

Global Variable ဆိုတာ functions တွေ Class တွေပြင်ပမှာတည်ဆောက်ထားတဲ့ variable တွေဖြစ်ပါတယ်။ဒါပေမယ့် ပြဿနာရှိတာက Global နဲ့ Local ရောနှောသွားတဲ့အခါမှာ ကျွန်တော်တို့ မလိုလားအပ်တဲ့ရလဒ်တွေရလာစေပါတယ်။ဥပမာ program တစ်ခုနဲ့ကြည့်ရအောင်။

```
GLOBAL_CONST = 42


def Usingglobal():
    print(GLOBAL_CONST)


Usingglobal()
```

GLOBAL_CONST = 42 ဆိုပြီး variable တစ်ခုကိုတည်ဆောက်ထားပါတယ်။function တစ်ခုတည်ဆောက် ပြီး အဲဒီထဲမှာဘဲ variable ထဲကတန်ဖိုးကို print လုပ်ထားပါတယ်။အခုလိုအဖြေထုတ်ကြည့်လိုက်တဲ့အခါမှာ 42 လို့အဖြေထွက်ပါတယ်။တကယ်လို့ဗျာ ကျွန်တော်တို့က အပေါ်မှာ ပေးခဲ့တဲ့နာမည်အတိုင်း function ထဲမှာ local variable ကိုနာမည်ပေးမိလိုက်ပြီး တခြားတန်ဖိုးထည့်လိုက်တဲ့အခါမှာ အဲဒီထည့်လိုက်တဲ့တန်ဖိုးဘဲ အဖြေထွက်သွားမှာပါ။ဒါဆိုရင် ကျွန်တော်တို့လုပ်ချင်တဲ့အလုပ်အတိုင်းတိတိကျကျလုပ်ချင်မှလုပ်ပါတော့လိမ့်မယ်။

```
GLOBAL_CONST = 42


def Usingglobal():
    GLOBAL_CONST = 777
    print(GLOBAL_CONST)


Usingglobal()
```

ဒီမှာ global variable နဲ့နာမည်တူ variable တည်ဆောက်ပြီး 777 တန်ဖိုးပေးလိုက်တဲ့အခါမှာ အဖြေထွက်လာတဲ့အခါမှာလဲ 777 ဖြစ်နေပါတယ်။ဒါမျိုးမဖြစ်အောင် ကျွန်တော်တို့ သတိလက်လွှတ်ဖြစ်ပြီးတော့ ဒီလို အမှားမျိုး မဖြစ်ရအောင်။global ဆိုတဲ့ python keyword တစ်ခုကိုအသုံးပြုပါမယ်

```
GLOBAL_CONST = 42

def Usingglobal():
    GLOBAL_CONST = 777
    global GLOBAL_CONST
    print(GLOBAL_CONST)

Usingglobal()
```

အခုလိုရေးလိုက်တဲ့အခါမှာ ကျွန်တော်တို့ local ထဲမှာ global နဲ့ နာမည်တူ variable တည်ဆောက်ပြီး တန်ဖိုး ပြောင်းလဲလို့မရတော့ပါဘူး။ဒါကိုအသုံးပြုခြင်းအားဖြင့် အစောပိုင်းကတည်းက Global တည်ဆောက်ထားပြီးသားဆိုတာသတိပေးပါလိမ့်မယ်။

How Coding in classes can imporove the GUI

အရှေ့ပိုင်းတွေတုန်းက သာမာန် script ရေးနည်းအတိုင်းရေးခဲ့တယ်။ဒါက မြန်မြန်နဲ့ရိုးရိုးရှင်းရှင်းရှိတဲ့ ပုံစံဖြစ်ပါတယ်။ဒါပေမယ့် တခါတရံ Project အကြီးတွေရေးတဲ့အခါမှာ Code တွေက များသထက် များလာပါတယ်။ဒီလိုများလာတဲ့အခါမှာ advance ရေးဟန်ဖြစ်တဲ့ OOP လိုအပ်လာပါတယ်။

OOP အသုံးပြုလို့ရမယ့်ကောင်းကျိုးတွေကို အပေါ်မှာလဲပြောခဲ့ပြီးသားပါ။ဒါပေမယ့်ပြန်ပြောပါ့မယ်။OOP က functions တွေ method တွေခွဲပြီး code တွေတည်ဆောက်နိုင်လို့ code တွေရှုပ်ယှက်ခတ်တာကို လျော့ ချ နိုင်ပါတယ်။ဒီလို code တွေက တပိုင်းစီခွဲထားနိုင်တဲ့အတွက် Error တတ်ခဲ့ရင် ရှာရလွယ်သလို မကြိုက်လို့ ပြင်မယ်ဆိုရင်တောင် လွယ်လွယ်ကူကူသူ့အပိုင်းလေးနဲ့သူခွဲပြင်နိုင်မှာဖြစ်ပါတယ်။တကယ်တော့ OOP က အရမ်းအခက်ကြီးမဟုတ်ပါဘူး class တွေတည်ဆောက်ပြီး Object ဖွဲ့ရေးတဲ့ပုံစံပါဘဲ။Python မှာ OOP ရေး ဖို့ အတွက် variables တွေအကုန်လုံးကို self နဲ့စတင်ရပါလိမ့်မယ်။

ပထမစရေးရေးချင်မှာတော့ OOP ပုံစံက စိတ်ရှုပ်စရာကောင်းပါလိမ့်မယ် ပြီးတော့ self တွေအပိုထည့်ပြီးရေးရတာလဲ လက်ညောင်းတယ်လို့ထင်ရင်ထင်ပါလိမ့်မယ်။ဒါပေမယ့်အခက်ခဲကြုံလာတဲ့အခါမှာ အကျိုးကျေးဇူးများတယ်လို့လက်တွေ့ယုံရမှာဖြစ်ပါတယ်။နောက်ပြီး စစရေးရေးခြင်းမှာ အများအနည်းငယ်တွေ့မယ် ဒါကို ကျွန်တော်တို့ပြုပြင်ယူသွားလို့ရတယ်။နောက်ပြီး ပုံစံမကျတာတွေဖြစ်ရင်ဖြစ်မယ် ဒါကို ကျွန်တော်တို့ စိတ်တိုင်းကျပြုပြင်ပြီးရေးရင်းရေးရင်းနဲ့ ကျွန်တော်တို့ OOP ကိုရင်းနှီးကျွမ်းဝင်လာမှာဖြစ်ပါတယ်။ကျွန်တော်လည်း Professional Developer တစ်ယောက်မဟုတ်သေးပါဘူး လေ့လာဆဲ လေ့လာတုန်းဖြစ်ပါတယ်။OOP ရေးတဲ့အခါမှာ အချို့မကျဆဲလူတစ်ဦးပါဘဲ ဒါပေမယ့် ကျွန်တော်စိတ်ဓာတ်မကျဘူး ဆက်လုပ်တယ် အခုထိလဲဆက်လုပ်ဆဲဘဲ ဒါကြောင့် မိတ်ဆွေတို့လဲစိတ်ဓာတ်မကျပါနဲ့လို့ပြောချင်ပါတယ်။စေတနာနဲ့ပါ။ဆရာလုပ်တာ မဟုတ်ပါဘူး။ကျွန်တော်တို့ ဘယ်လိုလေးရေးရင်တော့ပိုလွယ်မယ် ဘယ်လိုလေးရေးရင်တော့ပိုကောင်းမယ် ပိုပြီးရှင်းသွားမယ်ဆိုတာ ကိုတွေးပြီးရေးရင်းနဲ့ အချို့းကျတဲ့ Developer တယောက်ဖြစ်လာမယ်လို့ ယုံကြည် ပါတယ်။

```
from tkinter import *
from tkinter import ttk as gui
class OOP():
    def __init__(self):
        self.win = Tk()
        self.win.title("Python GUI")


    def clickMe():
        action.configure(text="Hello" + name.get())



    def widgets(self):
        tabControl = gui.Notebook(self.win)
        tab1 = gui.Frame(tabControl)
        tabControl.add(tab1,text="Tab 1")

        tab2 = gui.Frame(tabControl)
        tabControl.add(tab2,text='Tab 2')

        tabControl.pack(expand=1,fill='both')


if __name__ == "__main__":
    oop = OOP()
    oop.win.mainloop()
```

ဒီမှာ Class OOP ကိုတည်ဆောက်ပြီးတော့ GUI တစ်ခုတည်ဆောက်ထားပါတယ်။ဒီထဲမှာ widgets(self) clickMe(): ဆိုတဲ့ Functions နှစ်ခုတည်ဆောက်ပါတယ်။ဒီမှာ Error တချို့ရှိတဲ့ Function တစ်ခုရှိပါတယ် ဒါပေမယ့် Run လိုက်တဲ့အခါမှာ ဘာ Error မှမတတ်ဘဲ Run ပါလိမ့်မယ်။

```
from tkinter import *
from tkinter import ttk as gui
class OOP():
    def __init__(self):
        self.win = Tk()
        self.win.title("Python GUI")


    def clickMe():
        action.configure(text="Hello" + name.get())



    def widgets(self):
        tabControl = gui.Notebook(self.win)
        tab1 = gui.Frame(tabControl)
        tabControl.add(tab1,text="Tab 1")

        tab2 = gui.Frame(tabControl)
        tabControl.add(tab2,text='Tab 2')
```

ကြည့်လိုက်ပါ def clickMe() ဖြစ်ပါတယ်။Error တတ်ကြောင်း Pycharm မှာ အနီရောင်နဲ့ပြနေပေမယ့် ဘာကြောင့် Run တဲ့အခါမှာ ပြဿနာ မတတ်လဲဆိုတော့။Functions တွေက ခေါ်မှအလုပ်လုပ်ပါတယ် အခုက ဒီ function က အခေါ်မခံရသေးပါဘူး ဘာကြောင့် Error တတ်လဲဆိုရင် self မပါလို့ပါ။Class တွေထဲမှာရှိတဲ့ variable တွေနဲ့ method တွေမှာ self ပါသင့်ပါတယ်။ဒီ self ကြောင့်ဘဲ method တွေကနေ function လို့ အခေါ်ခံရတာဖြစ်ပါတယ်။ကျွန်တော်တို့ဟာ Function နဲ့ Method မကွဲဘူးဖြစ်နေပေမယ့်ဒီမှာတော့ ရှင်းရှင်းလင်းလင်းသိရတော့မှာပါ self ပါဂင်တဲ့ method တွေကို function လို့ခေါ်ပါတယ်။

```
def clickMe(self):
    self.action.configure(text="Hello" + self.name.get())
```

ဒီလိုမျိုးပြင်ဆင်လိုက်ရင် Error မရှိတော့ဘူး။ဒီမှာက OOP() class ထဲမှာ constructor ဖြစ်တဲ့ __init__ ထဲမှာ GUI Form တစ်ခုတည်ဆောက်ထားပါတယ်။တခြား Functions တွေကတော့ တည်ဆောက်တယ် ဒါပေမယ့် constructor ထဲမှာ ထည့်ပြီးမခေါ်ထားတော့ အလုပ်လုပ်မှာမဟုတ်ပါဘူး။ဒါဆိုရင်ကျွန်တော်တို့ Main လို့သတ်မှတ်တဲ့ class ရဲ့ constructor ထဲကနေ တခြား class တွေ function တွေကိုခေါ်ရမယ်။

Python က Class တွေအများကြီးတည်ဆောက်ပြီး Program ကိုတည်ဆောက်နိုင်တယ်။ဒါပေမယ့် တခါ တရံ Project အရမ်းကြီးလို့ Code တွေအများကြီးဖြစ်တဲ့အခါမှာ တချို့ Class တွေကို မူလ module ရဲ့အပြင်မှာ တည်ဆောက်ပြီးလှမ်းခေါ်ယူရတာမျိုးရှိတယ်။ဒါပေမယ့် မဖြစ်မနေလုပ်ရမယ့် အရာတော့မဟုတ်ပါဘူး။ဒါက Python ရဲ့အားသာချက်လို့ပြောလို့ရပါတယ် အရမ်းကို flexible ဖြစ်တဲ့ Programming အမျိုးစားဖြစ်ပါတယ်။

```
from tkinter import ttk as tk
from tkinter import *

class GUI():
    def __init__(self):
        self.win = Tk()
        self.win.title("Python GUI")
        self.win.geometry("300x300")
        self.create_widgets()


    def create_widgets(self):
        self.entry = Entry(self.win)
        self.entry.pack()




if __name__ == "__main__":
    gui = GUI()
    gui.win.mainloop()
```

နမူနာ Program လေးဖြစ်ပါတယ်။GUI class မှာ __init__ ပါတယ်။အဲဒီထဲမှာ win Form တစ်ခုတည်ဆောက်ထားပါတယ်။နောက်ထပ် function တစ်ခုတည်ဆောက်ပြီးအဲဒီထဲမှာ Entry widget တည်ဆောက်ထားတယ် အဲဒီ function ကို __init__ ထဲကနေလှမ်းခေါ်ထားတယ်။ဒါကလုံးဝနားလည်လွယ်ပါတယ်နော်။ဒါဆိုရင် ကျွန်တော်တို့လက်တွေ့ Program တွေရေးသွားတော့မယ် အဲဒီရေးသွားတဲ့ထဲမှာ OOP တွေတောက်လျောက်သုံးပြသွားပါ့မယ်။

Multiple Fun Projects

ကျွန်တော်တို့ Program တစ်ချို့ရေးကြရအောင် Android Phone နဲ့တွဲပြီး Phone Informations တွေပြတဲ့ Program အရင်ဆုံးရေးကြရအောင်။

Android Phone Informations Tool

ADB နဲ့ Python နဲ့တွဲဖက်ပြီးအသုံးပြုလို့ရပါတယ်။ဒီ Program က Phone ပညာရှင်တွေအတွက်တော့ အရမ်းအသုံးဝင်မယ်လို့ထင်ပါတယ်။ကျွန်တော်ကတော့ Phone ပညာရှင်မဟုတ်တဲ့အတွက် သိပ်ပြီး အလုပ်လုပ် နိုင်တဲ့ Tools တွေတော့မရေးနိုင်ပါဘူးလေ့လာဆဲပါ။ပညာရှင်တွေအတွက်အသုံးဝင်မယ် ထင်လို့ လေ့လာပြီးဖော်ပြလိုက်ပါတယ်။New Project ယူလိုက်ပြီး အဲဒီထဲမှာ adb ဆိုတဲ့ folder လေးကိုကူးထည့်လိုက် ပါပြီးရင် New ထဲက Python File ကိုရွေးပြီးတော့ နာမည်တစ်ခုပေးပြီးတော့တည်ဆောက်လိုက်ပါ။ပြီး Code ရေးကြရအောင်။Code အပြည့်စုံကို ရှင်းပြပြီးတဲ့အခါမှာ ဖော်ပြပါမယ် အခုတော့ တစ်ခုခြင်းဘဲရှင်းပြတော့မယ်။

```
from tkinter import *
from tkinter import ttk as gui
import os,time,threading,subprocess
from tkinter import messagebox as msgbox
```

ပထမဆုံး ကျွန်တော်တို့လိုအပ်တဲ့ modules တွေကို import လုပ်ပါတယ်။tkinter ကိုလုပ်တယ် ttk ကို import လုပ်ထပ်တယ် ပြီးတော့ os ရယ် time ရယ် threading ရယ် subprocess ရယ်ပါ import လုပ်လိုက်ပါတယ်။ဒါ တွေဘာအတွက်လိုတယ်ဆိုတာကို သုံးရင်းရှင်းပြသွားမယ်နော်။

```
adb = os.path.dirname(os.path.abspath(__file__))
ADB =adb+r'\adb\adb.exe'
Fastboot =adb+r'\adb\fastboot.exe'
batteryinfo = 'shell dumpsys battery|findstr'
```

အရင်ဆုံး variable adb ရယ် ADB ဆိုတာရယ် Fastboot ရယ် batteryinfo ဆိုတာရယ်တည်ဆောက်ပါတယ် ဒါတွေက ဘာအတွက်လဲဆိုရင် ကျွန်တော်တို့ Android ဖုန်းနဲ့ Computer နဲ့ချိတ်ဆက်ဖို့ ADB လို့ခေါ်တဲ့ Android Debugging Bridge ဆိုတဲ့ software လေးရှိတဲ့ အဲဒီ software လေးက Android ပိုင်ရှင်ဖြစ်တဲ့ Google ကိုယ်တိုင် ထုတ်ပေးထားတာဖြစ်ပါတယ်။ဒီကောင်လေးက Command Line Tool လေးပါ ဒါလေး နဲ့ ကျွန်တော်တို့ App တွေကို Install လုပ်နိုင်တယ်၊Android Phone Information တွေကြည့် နိုင်တယ် Lock တွေဖြုတ်နိုင်တယ် Myanmar Font တွေထည့်နိုင်ပါတယ်။ဒါပေမယ့် Advance Level တွေဖြစ်တဲ့ Font ထည့်တာ Lock ဖြုတ်တာ စတဲ့လုပ်ငန်းစဉ်တွေက ဖုန်း service ပညာရှင်တွေအတွက် တော့ အခက်ခဲမရှိပေမယ့် ကျွန်တော်တို့နဲ့ကအလှမ်းဝေးပါတယ်။ဒါကြောင့် ကျွန်တော်က Phone Information တချို့ ကြည့် ဖို့ နဲ့ Restart ချတာ Recovery ဝင်တာလောက်ဘဲလုပ်ထားတဲ့ ဆော့ဝဲလ်လေးကို Python နဲ့ရေးပြ သွားမှာ ပါ ဖုန်း service ပညာရှင်တွေ Android Phone Service Tool တွေထုတ်ချင်တဲ့ ညီအကိုတွေအတွက် အထောက် ကူဖြစ်မယ်လို့ထင်ပါတယ်။အဲဒီ ADB က ကျွန်တော်တို့ py python code file နဲ့ တွဲပြီး adb ဆိုတဲ့ Folder လေး ထည့်ထားပါတယ်။အဲဒီ Folder လေးထဲက adb.exe ကိုသွားချိတ်ရမှာဖြစ်ပါတယ်။အဲဒီလို ချိတ်ဖို့ အတွက် os module က အထောက်ပံ့ပေးပါတယ်။

```
adb = os.path.dirname(os.path.abspath(__file__))
```

ဒီ စာကြောင်းလေးက ကျွန်တော်တို့ software file ဘယ်နားလေးမှာရှိရှိ အဲဒီနေရာကိုဖော်ပြပေးတဲ့ code ဖြစ်ပါတယ်။ဒါပေမယ့် ဒါက adb.exe စီကိုမရောက်သေးဘူးဒါကြောင့် ထပ်ပြီး ADB ဆိုတဲ့ variable ထဲမှာ ထပ်ပြီး လမ်းကြောင်းချိတ်ဆက်ထားပါတယ်။

```
ADB =adb+r'\adb\adb.exe'
```

ဖိုင်လမ်းကြောင်းတွေမှာ r လေးထည့်ပေးရပါတယ်။အခုဆိုရင် adb.exe နဲ့ချိတ်ဆက်ပြီးပါပြီ။တကယ် တော့ ကျွန်တော်တို့ os.path.dirname(os.path.abspath(__file__))+r'\adb\adb.exe' ဆိုပြီး တိုက်ရိုက်ချိတ် လို့ရပါတယ်။အခုမှစတင်လေ့လာမယ့်လူတွေအတွက် ကျွန်တော်က တစ်ကြောင်းချင်းဘဲရေးထားတာပါ။အောက်က Fastboot တို့ဘာတွေကလဲ အတူတူဘဲ fastboot.exe ကိုချိတ်ဆက်တာဖြစ်ပါတယ်။fastboot ဘာ အသုံးဝင်တယ်ဘာညာဆိုတာသိချင်ရင် android ဆရာတွေကိုမေးပါ။batteryinfo က ခဏ ထားလိုက်အုန်း adb command တွေက ဘယ်လိုသုံးလဲဆိုတာအရင်ပြောပါမယ်။adb tool လေးရှိတဲ့ Folder လေးမှာ Shift ကို နှိပ်ပြီး Right Click နှိပ်လိုက်ပါ ပြီးရင် Open command window here ဆိုတာလေးကိုနှိပ်လိုက်ပါ။နောက်ပြီးဖုန်းကို USB နဲ့ချိတ်ပါ Setting ထဲက Developer options ထဲမှာ USB debugging ကိုဖွင့်ထားပါ။အဲဒီလိုလုပ်မှ ADB command တွေ Run နိုင်မှာဖြစ်ပါတယ်။နောက်တစ်ခုက Phone Driver တွေထည့်ထားဖို့လဲလိုပါတယ်။ကျွန်တော့် DVD ခွေထဲမှာ တချို့ကိုထည့်ပေးလိုက်ပါတယ်။

adb shell getprop ro.product.brand

```
D:\PythonCode\PythonProgram\Projects\android_adb_tool\adb>adb shell getprop ro.product.brand
Xiaomi

D:\PythonCode\PythonProgram\Projects\android_adb_tool\adb>
```

လို့ရိုက်လိုက်ပါ။ကိုယ့်ရဲ့ဖုန်းထုတ်လုပ်တဲ့ကုမ္ပဏီပေါ်လာပါလိမ့်မယ်။ဒီလိုမျိုး Command တွေနဲ့သုံး ပါတယ် ဒီလိုမျိုး Command တွေကို ကျွန်တော်တို့ Program မှာ ထည့်ရေးပြီးတော့ ပြန်ရလာတဲ့ ကုမ္ပဏီတံဆိပ် ဖုန်းအမျိုးအစားစတာတွေကို Label မှာဖော်ပြအောင်တည်ဆောက်မှာဖြစ်ပါတယ်။

```
class GUI():
    def __init__(self):
        self.win = Tk()
        self.win.title("Android Information Tool")
        self.win.geometry("300x300")
        self.win.resizable(0,0)
        self.widget_add()
        self.Theme()
        self.Runtime()
```

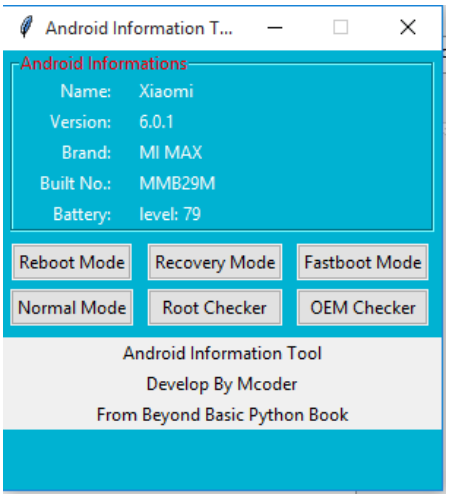
GUI ဆိုတဲ့ class တစ်ခုတည်ဆောက်လိုက်တယ် အဲဒီ class ထဲမှာ __init__ နဲ့ constructor တစ်ခုတည်ဆောက်ပြီး Windows Form တည်ဆောက်ပါတယ်။ကျွန်တော်ပြောတဲ့အတိုင်းပါဘဲ class ထဲမှာ တည်ဆောက်တဲ့ variable မှန်သမျှ self နဲ့စရပါတယ်။ဒီမှာ win form တည်ဆောက်တဲ့အပြင် self.widget_add() , self.Theme() , self.Runtime() ဆိုတဲ့ class သုံးခု constructor ထဲမှာခေါ်ထားပါသေးတယ် ပြော

တဲ့သဘောက constructor ကိုခေါ်ခံရတာနဲ့ အလုပ်လုပ်မှာဖြစ်ပါတယ်။ဒီ Functions တွေကဘာတွေ လုပ် လဲ ကြည့်ရအောင်။အရင်ဆုံး self.widget_add() ပေါ့။

```
def widget_add(self):
    self.text_frame =LabelFrame(self.win,text="Android Informations")
    self.frame_text =Frame(self.text_frame)

    self.text1 = Label(self.frame_text, text="Name: ")
    self.text1.grid(column=0, row=0, padx=5, sticky=E)

    self.text2 = Label(self.frame_text, text="--------------------")
    self.text2.grid(column=1, row=0, padx=5, sticky=W)

    self.text3 = Label(self.frame_text, text="Version: ")
    self.text3.grid(column=0, row=1, padx=5, sticky=E)

    self.text4 = Label(self.frame_text, text="--------------------")
    self.text4.grid(column=1, row=1, padx=5, sticky=W)

    self.text5 = Label(self.frame_text, text="Brand: ")
    self.text5.grid(column=0, row=2, padx=5, sticky=E)

    self.text6 = Label(self.frame_text, text="--------------------")
    self.text6.grid(column=1, row=2, padx=5, sticky=W)

    self.text7 = Label(self.frame_text, text="Built No.: ")
    self.text7.grid(column=0, row=3, padx=5, sticky=E)

    self.text8 = Label(self.frame_text, text="--------------------")
    self.text8.grid(column=1, row=3, padx=5, sticky=W)

    self.text9 = Label(self.frame_text, text="Battery: ")
    self.text9.grid(column=0, row=4, padx=5, sticky=E)

    self.text10 = Label(self.frame_text, text="--------------------")
    self.text10.grid(column=1, row=4, padx=5, sticky=W)

    self.frame_text.grid(padx=10)
    self.text_frame.pack(expand='no',fill='x',side=TOP,padx=5)

    #Button Widgets
    self.button_frame = Frame(self.win,pady=5)
    self.btn1 = gui.Button(self.button_frame, text="Reboot Mode",command=self.restart)
    self.btn1.grid(column=0, row=0,pady=3,padx=5)

    self.btn2 = gui.Button(self.button_frame, text="Recovery
Mode",command=self.recovery_mode)
    self.btn2.grid(column=1, row=0,pady=3,padx=5)

    self.btn3 = gui.Button(self.button_frame, text="Fastboot
Mode",command=self.fastboot_restart)
    self.btn3.grid(column=2, row=0,pady=3,padx=5)

    self.btn4 = gui.Button(self.button_frame, text="Normal
Mode",command=self.normal_mode)
    self.btn4.grid(column=0, row=1, pady=3, padx=5,sticky=NSEW)

    self.btn5 = gui.Button(self.button_frame, text="Root
Checker",command=self.su_check)
    self.btn5.grid(column=1, row=1, pady=3, padx=5,sticky=NSEW)
```

```
    self.btn6 = gui.Button(self.button_frame, text="OEM
Checker",command=self.oem_check)
    self.btn6.grid(column=2, row=1, pady=3, padx=5,sticky=NSEW)

    self.button_frame.pack(expand='no', fill='x', side=TOP)

    Label(self.win,text="Android Information
Tool").pack(expand='no',fill='x',side=TOP)
    Label(self.win, text="Develop By Mcoder").pack(expand='no', fill='x', side=TOP)
    Label(self.win, text="From Beyond Basic Python Book").pack(expand='no', fill='x',
side=TOP)
```

Code တွေက များလို့စိတ်မရှုပ်ပါနဲ့နော် အားလုံးက GUI ပေါ်မှာနေရာချတာဘဲ ဖြစ်ပါတယ် အထူးအထေ ပြောစရာမလိုပါဘူးအရှေ့ခန်းတွေမှာလဲဖော်ပြခဲ့ပြီးဖြစ်ပါတယ်။sticky = E ဆိုတာ Right ဘက် အညီစာသားညိတာဖြစ်ပါတယ် W ကတော့ Left ဘက်ကိုအညီပေါ့။NSEW ကတော့ အပေါ်က widget ရဲ့ပမာဏ အတိုင်း အောက်က widgets တွေကို ညိတာဖြစ်ပါတယ်။အောက်ပါပုံစံအတိုင်း GUI ပုံစံချတာဖြစ်ပါတယ်။



ဒါဆိုရင် self.Theme() ထဲမှာ ဘာတွေပါလဲလေ့လာကြည့်မယ်။ဘာမှမဟုတ်ဘူး လှအောင် GUI မှာ အရောင် လေးတွေသီးသန့် Function ခွဲတည်ဆောက်ထားတာဖြစ်ပါတယ်။

```
def Theme(self):
    self.win.config(bg='#00b2d2')
    self.text_frame.config(bg='#00b2d2',fg='red')
    self.frame_text.config(bg='#00b2d2')
    self.text1.config(bg='#00b2d2',fg='#ffffff')
    self.text2.config(bg='#00b2d2',fg='#ffffff')
    self.text3.config(bg='#00b2d2',fg='#ffffff')
    self.text4.config(bg='#00b2d2',fg='#ffffff')
    self.text5.config(bg='#00b2d2',fg='#ffffff')
    self.text6.config(bg='#00b2d2',fg='#ffffff')
    self.text7.config(bg='#00b2d2',fg='#ffffff')
    self.text8.config(bg='#00b2d2',fg='#ffffff')
    self.text9.config(bg='#00b2d2',fg='#ffffff')
    self.text10.config(bg='#00b2d2',fg='#ffffff')
    self.button_frame.config(bg='#00b2d2')
```

အထက်ပါအတိုင်းဘဲ widget တွေကိုခေါ်ပြီး config() နဲ့ bg ဆိုတာ background color ဖြစ်ပြီး fg ဆိုတာ foreground color အနေနဲ့သုံးတာဖြစ်ပါတယ်။ဒါဆိုရင်နောက် function တစ်ခုက self.Rumtime() ဖြစ်ပါတယ်။အဲဒီထဲမှာလုပ်မယ့်အလုပ်ကတော့

```
def Runtime(self):
    self.interval = 1
    thread = threading.Thread(target=self.run, args=())
    thread.daemon = True
    thread.start()

def run(self):
    while True:
        self.adb_working()
        time.sleep(self.interval)
```

ဒါကို thread လုပ်ခေါ်ပါတယ်။Threading Module နဲ့ Time module ကိုအသုံးပြုတည်ဆောက် ထားတာဖြစ်ပါတယ်။ဒီ Program ပြီးရင် Thread အကြောင်းကိုသီးသန့်ရှင်းပြပါမယ်။အခုသုံးထားတဲ့ Thread ကလည်း ကျွန်တော်ရေးထားတာမဟုတ်ပါဘူး နိုင်ငံခြားက Developer တစ်ယောက်ရေးထားတာကိုအသုံးပြုထားတာ။ သူ့ ရဲ့အလုပ်ကတော့ software ကိုနောက်ကွယ်မှာ အမြဲတမ်း အလုပ်လုပ်နေစေဖို့ဖြစ်ပါတယ်။ဒီထဲမှာ 1 စက္ကန့်ခြားတစ်ခါ self.adb_working() အလုပ်လုပ်မယ်လို့ပြောထားပါတယ်။ကျွန်တော်တို့ အမြဲ Run နေစေချင်တဲ့ adb.working() ဆိုတဲ့ function ကြီးကဘာတွေများလုပ်ထားတာတုန်း တစ်ချက်ကြည့်ရအောင်။

```
def adb_working(self):
    self.adb_device_wait = subprocess.getoutput('%s shell getprop
ro.product.brand'%ADB)

    if self.adb_device_wait == "error: device not found":
        pass
    elif self.adb_device_wait == "error: device unauthorized. Please check the
confirmation dialog on your device.":
        pass

    else:
        self.device_name = subprocess.getoutput('%s shell getprop
ro.product.brand'%ADB)
        try:
            self.device_raw = self.device_name.split()
            self.device_name_1 = self.device_raw


        except:
            pass

        self.text2.config(text=self.device_name_1)

        self.device_version = subprocess.getoutput('%s shell getprop
ro.build.version.release' % ADB)
        try:
            self.version_raw = self.device_version.split()
            self.device_version_1 = self.version_raw


        except:
            pass
```

```
        self.text4.config(text=self.device_version_1)

        self.brand_name = subprocess.getoutput('%s shell getprop ro.product.model' %
ADB)
        try:
            self.brand_raw = self.brand_name.split()
            self.device_brand_1 = self.brand_raw


        except:
            pass

        self.text6.config(text=self.device_brand_1)

        self.brand_build = subprocess.getoutput('%s shell getprop ro.build.display.id'
% ADB)
        try:
            self.build_raw = self.brand_build.split()
            self.device_build_1 = self.build_raw


        except:
            pass

        self.text8.config(text=self.device_build_1)

        self.brand_battery = subprocess.getoutput('%s shell dumpsys battery|findstr
"level"' %(ADB))
        try:
            self.battery_raw = self.brand_battery.split()
            self.device_battery_1 = self.battery_raw


        except:
            pass

        self.text10.config(text=self.device_battery_1)
```

ဒီထဲမှာလုပ်ထားတဲ့အလုပ်တွေက adb command တွေရိုက်ထားပြီး အဲဒီက return ပြန်ရလာတဲ့ value တွေကို Label တွေထဲကိုထည့်ဖို့လုပ်ထားတာဘဲဖြစ်ပါတယ်။တစ်ခုကိုရှင်းပြလိုက်ရင်နောက် လုပ်ထားတာတွေအကုန်နားလည်သွားမှာပါ။

```
self.adb_device_wait = subprocess.getoutput('%s shell getprop ro.product.brand'%ADB)

if self.adb_device_wait == "error: device not found":
    pass
elif self.adb_device_wait == "error: device unauthorized. Please check the
confirmation dialog on your device.":
    pass

else:
```

ပထမဆုံးကျွန်တော်က device နဲ့ချိတ်လားမချိတ်လားသိရအောင်ရယ် device နဲ့ချိတ်ဆက်တာသေချာမှ နောက် အလုပ်တွေဆက်လုပ်ဖို့ကို အရင်ဆုံး

```
self.adb_device_wait = subprocess.getoutput('%s shell getprop ro.product.brand'%ADB)
```

variable ဖြစ်တဲ့ self.adb_device_wait ထဲကို subprocess.getoutput() နဲ့ adb command ရိုက်ထားပါတယ်။subprocess.getoutput() ထဲမှာ adb command တွေရိုက်နိုင်ပြီးတော့ သူကနေ return ပြန်လာတဲ့

string တွေကို variable ထဲကိုသိမ်းလိုက်တာဖြစ်ပါတယ်။အဲဒီမှာ Run လိုက်တဲ့အခါမှာ ဖုန်းနဲ့ computer ဟာ ချိတ်ဆက်မထားဘူးဆိုရင် error: device not found လို့ပေါ်မှာဖြစ်ပါတယ်။ဒီလိုမှမဟုတ်ဘဲ USB debugging ဖွင့်မထားတာမျိုး android phone မှာ permission မပေးတာမျိုးဆိုရင်တော့

```
error: device unauthorized. Please check the confirmation dialog on your device.
```

လို့ပေါ်မှာဖြစ်ပါတယ်။ဒီနှစ်ခုကို if နဲ့စစ်ထားတယ်။ဒီနှစ်ခုဖြစ်နေခဲ့ရင် pass လို့ပြောလိုက်တာ ဘာအလုပ်မှ မလုပ်ဘူး ကျော်သွားလို့ပြောလိုက်တာပါဘဲ။else ထဲမှာတော့ ကျွန်တော်တို့လုပ်ချင်တဲ့အလုပ်တွေကို လုပ်မှာဖြစ်ပါတယ်။ဘာကြောင့်လဲဆိုတော့ အပေါ်ကနှစ်ခုလုံးဖြစ်မနေဘူးဆိုရင်သေချာပေါက် ကျွန်တော်တို့ ဖုန်းနဲ့ ဆော့ဝဲလ်နဲ့ချိတ်ဆက်နေပြီလို့ယူဆရမှာဖြစ်ပါတယ်။

```
self.device_name = subprocess.getoutput('%s shell getprop ro.product.brand'%ADB)
```

နောက်တစ်ခုကတော့ self.device_name ထဲမှာ subprocess.getoutput() နဲ့ command ကို ရိုက်ထားပါတယ် ဒါက ဘယ်လိုရိုက်တာလဲဆိုတော့ adb shell getprop ro.product.brand ဆိုပြီးရိုက်တာပါ။%s လို့ခေါ်တဲ့ special argument %ADB နဲ့ adb.exe လမ်းကြောင်းလေးကိုထည့်ပြီး ကျွန်တော်တို့ ဆက်ရေးရမယ့် shell…. brand ဆိုတဲ့ command ကိုထည့်လိုက်တာဖြစ်ပါတယ်။ရလာတဲ့ value ကို self.device_name ထဲက တိုက်ရိုက်သုံးရင် ပုံစံတကျမဖြစ်ဘဲရှိနေပါတယ်။ဒါကြောင့် split() ကိုသုံးပြီး မလိုတဲ့ space တွေ \n တွေကိုဖယ်ပြီးသုံးမှအဆင်ပြေမှာဖြစ်ပါတယ်။ဒါပေမယ့် ဘာတန်ဖိုးမှမရှိတာကို split() သုံးမိရင် error တွေတတ်လာမှာပါဥပမာ Phone နဲ့ PC နဲ့မချိတ်ဘဲဆော့ဝဲလ်ဖွင့်တဲ့အချိန်မျိုးမှာပေါ့ ဒီလို Error တွေမတတ်ချင်လို့ try: except: နဲ့ Error ဖမ်းထားပါတယ်။အဲဒီထဲမှာ subprocess က ရလာတဲ့တန်ဖိုးကို split()နဲ့ခွဲထုတ်ပြီးပြန်ရယူထားတဲ့တန်ဖိုးကို GUI ပေါ်က Label ထဲမှာ config() နဲ့ထည့်လိုက်တာဖြစ်ပါတယ်။ဒီတော့ ဒီ function ကို အမြဲတမ်း Running လုပ်နေအောင် Threading လုပ်ထားတော့ ဖုန်းရဲ့ပြောင်းလဲနေတဲ့ Battery အားတွေ ဖုန်းပြောင်းချိတ်တဲ့အခါမှာ ပြောင်းလဲသွားမယ့် model no. တွေ product name တွေက အမြဲ ရနေမှာဖြစ်ပါတယ်။အောက်မှာ function အသေးလေးတွေတည်ဆောက်ပါတယ်။ဒါတွေကတော့ Button တွေကို နှိပ်ရင် အလုပ်လုပ်ဖို့ဘဲဖြစ်ပါတယ်။subprocess.getoutput() နဲ့ဘဲ ကွန်မန့်တွေရေးထားတာကိုတွေ့ရမှာပါ။ပြီးတဲ့အခါမှာ if __name __ =="__main__": ထဲမှာ gui =GUI() ဆိုပြီး GUI class ရဲ့ object ကိုဆောက်တယ်။နောက်ပြီး GUI class ထဲက win ဆိုတဲ့ object ကို mainloop() နဲ့အမြဲ GUI ဇပါနေ အောင် ပြုလုပ်ထားပါတယ်။Code အပြည့်စုံကို အောက်မှာဖော်ပြထားပါတယ်။

```
from tkinter import *
from tkinter import ttk as gui
import os,time,threading,subprocess
from tkinter import messagebox as msgbox

adb = os.path.dirname(os.path.abspath(__file__))
ADB =adb+r'\adb\adb.exe'
Fastboot =adb+r'\adb\fastboot.exe'
batteryinfo = 'shell dumpsys battery|findstr'

class GUI():
    def __init__(self):
        self.win = Tk()
        self.win.title("Android Information Tool")
        self.win.geometry("300x300")
```

```
        self.win.resizable(0,0)
        self.widget_add()
        self.Theme()
        self.Runtime()



    def widget_add(self):
        self.text_frame =LabelFrame(self.win,text="Android Informations")
        self.frame_text =Frame(self.text_frame)

        self.text1 = Label(self.frame_text, text="Name: ")
        self.text1.grid(column=0, row=0, padx=5, sticky=E)

        self.text2 = Label(self.frame_text, text="--------------------")
        self.text2.grid(column=1, row=0, padx=5, sticky=W)

        self.text3 = Label(self.frame_text, text="Version: ")
        self.text3.grid(column=0, row=1, padx=5, sticky=E)

        self.text4 = Label(self.frame_text, text="--------------------")
        self.text4.grid(column=1, row=1, padx=5, sticky=W)

        self.text5 = Label(self.frame_text, text="Brand: ")
        self.text5.grid(column=0, row=2, padx=5, sticky=E)

        self.text6 = Label(self.frame_text, text="--------------------")
        self.text6.grid(column=1, row=2, padx=5, sticky=W)

        self.text7 = Label(self.frame_text, text="Built No.: ")
        self.text7.grid(column=0, row=3, padx=5, sticky=E)

        self.text8 = Label(self.frame_text, text="--------------------")
        self.text8.grid(column=1, row=3, padx=5, sticky=W)

        self.text9 = Label(self.frame_text, text="Battery: ")
        self.text9.grid(column=0, row=4, padx=5, sticky=E)

        self.text10 = Label(self.frame_text, text="--------------------")
        self.text10.grid(column=1, row=4, padx=5, sticky=W)

        self.frame_text.grid(padx=10)
        self.text_frame.pack(expand='no',fill='x',side=TOP,padx=5)

        #Button Widgets
        self.button_frame = Frame(self.win,pady=5)
        self.btn1 = gui.Button(self.button_frame, text="Reboot
Mode",command=self.restart)
        self.btn1.grid(column=0, row=0,pady=3,padx=5)

        self.btn2 = gui.Button(self.button_frame, text="Recovery
Mode",command=self.recovery_mode)
        self.btn2.grid(column=1, row=0,pady=3,padx=5)

        self.btn3 = gui.Button(self.button_frame, text="Fastboot
Mode",command=self.fastboot_restart)
        self.btn3.grid(column=2, row=0,pady=3,padx=5)

        self.btn4 = gui.Button(self.button_frame, text="Normal
Mode",command=self.normal_mode)
        self.btn4.grid(column=0, row=1, pady=3, padx=5,sticky=NSEW)
```

```
        self.btn5 = gui.Button(self.button_frame, text="Root
Checker",command=self.su_check)
        self.btn5.grid(column=1, row=1, pady=3, padx=5,sticky=NSEW)

        self.btn6 = gui.Button(self.button_frame, text="OEM
Checker",command=self.oem_check)
        self.btn6.grid(column=2, row=1, pady=3, padx=5,sticky=NSEW)

        self.button_frame.pack(expand='no', fill='x', side=TOP)

        Label(self.win,text="Android Information
Tool").pack(expand='no',fill='x',side=TOP)
        Label(self.win, text="Develop By Mcoder").pack(expand='no', fill='x',
side=TOP)
        Label(self.win, text="From Beyond Basic Python Book").pack(expand='no',
fill='x', side=TOP)
```

```
    def Theme(self):
        self.win.config(bg='#00b2d2')
        self.text_frame.config(bg='#00b2d2',fg='red')
        self.frame_text.config(bg='#00b2d2')
        self.text1.config(bg='#00b2d2',fg='#ffffff')
        self.text2.config(bg='#00b2d2',fg='#ffffff')
        self.text3.config(bg='#00b2d2',fg='#ffffff')
        self.text4.config(bg='#00b2d2',fg='#ffffff')
        self.text5.config(bg='#00b2d2',fg='#ffffff')
        self.text6.config(bg='#00b2d2',fg='#ffffff')
        self.text7.config(bg='#00b2d2',fg='#ffffff')
        self.text8.config(bg='#00b2d2',fg='#ffffff')
        self.text9.config(bg='#00b2d2',fg='#ffffff')
        self.text10.config(bg='#00b2d2',fg='#ffffff')
        self.button_frame.config(bg='#00b2d2')


    def adb_working(self):
        self.adb_device_wait = subprocess.getoutput('%s shell getprop
ro.product.brand'%ADB)

        if self.adb_device_wait == "error: device not found":
            pass
        elif self.adb_device_wait == "error: device unauthorized. Please check the
confirmation dialog on your device.":
            pass

        else:
           self.device_name = subprocess.getoutput('%s shell getprop
ro.product.brand'%ADB)
           try:
               self.device_raw = self.device_name.split()
               self.device_name_1 = self.device_raw


           except:
               pass

           self.text2.config(text=self.device_name_1)
```

```
            self.device_version = subprocess.getoutput('%s shell getprop
ro.build.version.release' % ADB)
            try:
                self.version_raw = self.device_version.split()
                self.device_version_1 = self.version_raw


            except:
                pass

            self.text4.config(text=self.device_version_1)

            self.brand_name = subprocess.getoutput('%s shell getprop ro.product.model'
% ADB)
            try:
                self.brand_raw = self.brand_name.split()
                self.device_brand_1 = self.brand_raw


            except:
                pass

            self.text6.config(text=self.device_brand_1)

            self.brand_build = subprocess.getoutput('%s shell getprop
ro.build.display.id' % ADB)
            try:
                self.build_raw = self.brand_build.split()
                self.device_build_1 = self.build_raw


            except:
                pass

            self.text8.config(text=self.device_build_1)

            self.brand_battery = subprocess.getoutput('%s shell dumpsys battery|findstr
"level"' %(ADB))
            try:
                self.battery_raw = self.brand_battery.split()
                self.device_battery_1 = self.battery_raw


            except:
                pass

            self.text10.config(text=self.device_battery_1)

    def Runtime(self):
        self.interval = 1
        thread = threading.Thread(target=self.run, args=())
        thread.daemon = True
        thread.start()

    def run(self):
        while True:
            self.adb_working()
            time.sleep(self.interval)

    def restart(self):
            subprocess.getoutput('%s reboot'%ADB)

    def su_check(self):
```

```
        su_checker = subprocess.getoutput('%s shell su'%ADB)

        try:
            su_ch_raw = su_checker.split(":")
            msgbox._show("Root Checker", "Device Root :" + su_ch_raw[2])

        except:
            pass

    def oem_check(self):
        oem_check = subprocess.getoutput('%s oem get-bootinfo'%Fastboot)
        print(oem_check)

    def fastboot_restart(self):
        fastboot_mode = subprocess.getoutput('%s reboot bootloader'%ADB)

    def recovery_mode(self):
        recovery_mode = subprocess.getoutput('%s reboot recovery'%ADB)

    def normal_mode(self):
        subprocess.getoutput('%s reboot'%Fastboot)




if __name__ == "__main__":
    gui = GUI()
    gui.win.mainloop()
```

What is Threading and Thread

Thread ဆိုတာ execution ပြုလုပ်တဲ့ unit တစ်ခုဘဲဖြစ်ပါတယ်။Thread က operating system ရဲ့ အလုပ်လေးတွေကို အစီစဉ်တကျလုပ်နိုင်စွမ်းရှိပါတယ်။Threads ကို သာမန်အားဖြင့် computer script ရဲ့ fork ဒါမှမဟုတ် ၂ ခုသို့မဟုတ် နှစ်ခုထက်ပိုတဲ့အပြိုင် program တွေရဲ့လုပ်ငန်းစဉ်တွေနဲ့တည်ဆောက်ထားပါတယ်။Threads တွေမှာ အမြဲလို processes တွေမှာပါဝင်ပါတယ်။တစ်ခုထက်ပိုတဲ့ Thread တွေက တူညီတဲ့ process အတွင်းရှိနေနိုင်ပါတယ်။ပြောချင်တဲ့သဘောက process တစ်ခုမှာ thread ဟာ တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုပြီးရှိနေနိုင်တယ်လို့ပြောတာဖြစ်ပါတယ်။ဒီ threads တွေဟာ process ရဲ့ state နဲ့ memory တွေကို share နိုင်ပါတယ်။ပြောရရင် thread တွေက code ဒါမှမဟုတ် အမိန့် instructions တွေနဲ့ သူတို့ရဲ့ variables တန်ဖိုးတွေကို share နိုင်ပါတယ်။

Threads အမျိုးအစား နှစ်မျိုးရှိပါတယ်။

- Kernel threads
- User-space threads or user threads

တို့ဘဲဖြစ်ပါတယ်။

Kernel threads တွေဆိုတာ operating system OS ရဲ့ အစိတ်အပိုင်းတွေဖြစ်ပြီးတော့ user threads တွေကတော့ kernel ရဲ့အစိတ်အပိုင်းမဟုတ်ပါဘူး။သေချာတာကတော့ user threads တွေက programming language ရဲ့ function concept တွေရဲ့ extension တွေကိုမြင်နိုင်ပါတယ်။ဒါကြောင့် thread အမျိုးစားဖြစ်တဲ့ user thread က function သို့မဟုတ် procedure call နဲ့တူပါတယ်။ဒါပေမယ့် သာမန် functions တွေနဲ့တော့ကွာခြားပါတယ် အဓိကအားဖြင့် return ပြန်တဲ့ ဝိသေသ လက္ခဏာတွေကခြားနားတာ ဖြစ်ပါတယ်။

Process တိုင်းမှာ အနည်းဆုံး Thread တစ်ခုတော့ရှိပါတယ်။ဥပမာ အဲဒီ process run နေတာမျိုးပေါ့။Process တစ်ခု က threads တွေအများကြီးကိုစတင်နိုင်ပါတယ်။OS က ဒီ threads တွေကို parallel ပုံစံမျိုးနဲ့ စီမံ ဆောင်ရွက်နိုင်ပါတယ်။Single Processor တစ်ခုဘဲသုံးတဲ့ machine ပေါ်မှာ ဒီ အပြိုင်အသုံးပြုတဲ့ parallelism က အစီစဉ်တကျဆောင်ရွက်ခြင်းနဲ့ timeslicing လုပ်ငန်းစဉ်ကိုရယူနိုင်ပါတယ်။

Threading ရဲ့ ကောင်းကျိုးများကိုဖော်ပြရမယ်ဆိုရင်တော့

- Multithreaded programs တွေက computer systems တွေပေါ်မှာ multiple CPUs တွေကို အသုံးချ ပြီးမြန်မြန်ဆန်ဆန် run နိုင်ပါတယ်။ဒီ threads တွေက တကယ့်ကို တပြိုင်ထဲဖြစ်ပေါ်နေတဲ့ လုပ်ငန်း စဉ်တွေကိုလုပ်နိုင်တာဖြစ်ပါတယ်။
- Program တွေဟာ input လုပ်လိုက်တာတွေကို မဆိုင်းမတွဘဲချက်ခြင်းလုပ်နိုင်ပါတယ်။ဒါက တကယ့် ကို single နဲ့ multiple CPU တွေပေါ်မှာ မှန်ကန်စွာလုပ်နိုင်ခြင်းဖြစ်ပါတယ်။
- Process တစ်ခုရဲ့ threads တွေက global variables ရဲ့ memory တွေကို share နိုင်ပါတယ်။global variable တစ်ခုက thread တစ်ခုပေါ်မှာ ပြောင်းလဲသွားတယ်ဆိုရင် ဒီအပြောင်းလဲက တခြား threads တွေနဲ့လိုက်လျောညီထွေဖြစ်ပါတယ်။thread တစ်ခုမှာ local variables တွေလည်း ရှိနိုင် ပါတယ်။

OS တစ်ခုအတွက် threads တွေကို ကိုင်တွယ်ရတာက Processes တွေကို ကိုင်တွယ်ရတာထက် အများကြီး ရိုးရှင်းပါတယ်။ဒါကြောင့်လဲ တစ်ခါတလေမှာ LWP (Light-Weight Process) လို့ခေါ်တာဖြစ်ပါတယ်။

Threads in Python

Python အတွက် Threads တွေအသုံးပြုနိုင်တဲ့ modules နှစ်မျိုးရှိပါတယ်။

- thread
- threading

ဖြစ်ပါတယ်။

Thread module ကို deprecated လုပ်ဖို့ အတော်သင့်အချိန်အကြာကြီးမှာ ဆုံးဖြတ်ခဲ့ပြီးဖြစ်သလို အသုံးပြုသူများကလည်း threading module ကိုအစားထိုးအသုံးပြုဖို့အားပေးခဲ့ပါတယ်။Python 3 မှာ thread module ဟာ သုံးလို့မရတော့ပါဘူး။ဒါပေမယ့် လုံးဝမပါတော့တာမဟုတ်ပါဘူး thread ဆိုတဲ့နာမည်ကနေ _thread ကို နာမည်ပြောင်းသွားတာဖြစ်ပါတယ်။ဒါကြောင့် Python 3 အသုံးပြုသူများဟာ _thread ကိုအသုံးပြုရမှာ ဖြစ်ပါတယ်။

Thread or _thread module ဟာ function ကဲ့သို့ အသုံးပြုနိုင်ပါတယ်။threading module ကတော့ object oriented နည်းလမ်းကို အထမြောက်စေပါတယ်။thread အားလုံးဟာ object ပေါ်မှာ အကျိုးသက်ရောက်ပါတယ်။

The Thread Module Example

Functions ရဲ့ ခွဲခြမ်းစိတ်ဖြာတဲ့ thread တွေကို ပြီးမြောက်အောင်လုပ်ဆောင်ရန်အတွက် _thread module က လုပ်ဆောင်ပေးနိုင်ပါတယ်။ဒီလိုမျိုးလုပ်ဆောင်ဖို့အတွက် ကျွန်တော်တို့က _thread module ထဲက thread.start_new_thread() ဆိုတဲ့ function ကိုအသုံးပြုပြီးလုပ်ဆောင်နိုင်ပါတယ်။

Start_new_thread(function,args[, kwargs])

ဆိုပြီးအသုံးပြုနိုင်ပါတယ်။function ဆိုတာကတော့ thread လုပ်မယ့် function အတွက်ဖြစ်ပြီးတော့ args နဲ့ kwargs လို့ခေါ်တဲ့ list or tuple args တွေကိုအသုံးပြုနိုင်ပါတယ်။start_new_thread() က thread အသစ်တစ်ခုနဲ့သူ့ရဲ့ ဝိသေသလက္ခဏာတွေကို return ပြန်ပေးပါတယ်။thread က function နဲ့ argument or list arguments တွေကို excutes လုပ်နိုင်ပါတယ်။function က returns လုပ်တဲ့အခါမှာ thread က silently exits အနေနဲ့ထွက်သွားမယ် ပြီးတော့တခြား threads ကတော့ ဆက်ပြီး run နေမှာဖြစ်ပါတယ်။ဒါဆိုရင် လက်တွေ့ program တစ်ခုရေးကြည့်မယ်။

```
from _thread import start_new_thread

def heron(a):
    eps = 0.0000001
    old =1
    new =1
    while True:
        old,new = new,(new + a/new)/2.0
        print(old,new)
        if abs(new - old )<eps:
            break
    return new
start_new_thread(heron,(5,))
start_new_thread(heron,(99,))
start_new_thread(heron,(999,))
c = input()
```

ကျွန်တော်က python 3 ကိုအသုံးပြုတာဖြစ်တဲ့အတွက် _thread ကိုအသုံးပြုပါတယ် python 2 မှာကတော့ thread ကိုအသုံးပြုပါတယ်။_thread module ထဲက start_new_thread ကို import လုပ်ပါတယ်။ပြီးတော့

function တစ်ခုတည်ဆောက်တယ် heron နာမည်နဲ့ပါ argument တစ်ခုပါတယ် a ဖြစ်ပါတယ်။ဒီထဲက အလုပ် တွေက eps , old , new ဆိုပြီး variable 3 ခုပါပြီး while True ဆိုတဲ့ အမြဲတမ်း Running လုပ်နေမယ့် အလုပ်ထဲမှာ old,new ဆိုပြီး variables အသစ် ၂ ခုတည်ဆောက်တယ်။ပြီးတော့အဲဒီ variables ၂ ခုထဲကို old ထဲကို new ထည့်တယ်။new ထဲကိုတော့ (new +a/new)/2.0 တန်ဖိုးကိုထည့်ပါတယ် ပြီးရင် old, နဲ့ new ကို print လုပ်ပါတယ်။နောက်ပြီး if နဲ့ abs() ကိုအသုံးပြုပြီး new – old ကရတဲ့အဖြေကို ဒသမကိန်းပြောင်းပြီး eps ထက်ငယ်ပြီဆိုရင် break လို့ရေးထားတာဖြစ်လို့ လုပ်ငန်းစဉ်ရပ်တန့်သွားပါမယ်လို့ပြောထားပါတယ်။အဲဒီနောက်မှာတော့ start_new_thread() နဲ့ thread 3 ခုတည်ဆောက်ပြီး မတူညီတဲ့ argument တွေကို passing လုပ်ပေးပြီး execute လုပ်ပေးပါတယ်။ဒီ Program ကို execute လုပ်လိုက်တဲ့အခါမှာ thread 3 ခု run သွားပါတယ်။ a နေရာမှာ 5 ထည့်လိုက်တာရယ် 99 ရယ် 999 ရယ်သုံးခုပါ ဒီသုံးခုပြီးတဲ့အခါမှာ 5 ထည့်လိုက်လို့ရတဲ့ thread ရဲ့အဖြေတွေကို abs(new -–old) <eps အခြေနေမဖြစ်မခြင်း break မဖြစ်တဲ့အတွက် လုပ်ပါတယ်။အဲဒီအခြေနေမှန်သွားတဲ့အခါမှာ program break ဖြစ်ပြီး thread တစ်ခုထွက်သွားမယ်။ပြီးတော့ တခြား thread ဖြစ်တဲ့ 99 က if နဲ့စစ်ထားတဲ့ program break မဖြစ်မခြင်းလုပ်မယ် ပြီးရင်ရပ်သွားမယ် နောက်ထပ် 999 ကထပ်လုပ်မယ် ရပ်သွားမယ် ဒီလိုပုံမျိုးနဲ့လာပါတယ်။ဒီနေရာမှာ c = input() ဆိုပြီးထည့်ထားတာ program ကလုပ်ငန်းတွေပြီးရင်ချက်ခြင်းရပ်သွားမှာမလိုလားလို့ဖြစ်ပါတယ် Key တစ်ခုခုကို input လုပ်ပေးမှသာ program ရပ်ပါလိမ့်မယ်။

ဒါက thread module သုံးပြီး program တည်ဆောက်တာ သက်သက်ဘဲ ကျွန်တော်တို့ တကယ်သုံးမှာက threading module ကိုသုံးမယ် thread module က မသုံးတော့တာကြောပြီလို့တချို့ website တွေမှာဖတ်ရတယ်။

Threading Module with Time

Threading ကိုဘာကြောင့်အသုံးပြုလဲဆိုတာ အပေါ်မှာပြောခဲ့ပြီးပါပြီ ကျွန်တော်တို့ run နေစေချင်တဲ့ functions တွေဟာ program နဲ့ထပ်တူအချိန်ပြည့် run နေစေချင်လို့ဖြစ်ပါတယ်။အတိုချုံးပြောရရင် threads က programs တွေကို multiple tasks တွေကို တချိန်ထဲမှာ execute လုပ်ဖို့ အတွက် အသုံးပြု နိုင်တာဖြစ်ပါတယ်။

```
import threading

class MyThread(threading.Thread):
    def __init__(self,x):
        self.__x = x
        threading.Thread.__init__(self)

    def run(self):
        print(self.__x)


for x in range(10):
    MyThread(x).start()
```

ဒီ program မှာ threading module ကို import လုပ်ပြီးတော့ MyThread နာမည်နဲ့ class တစ်ခု တည် ဆောက်ထားပြီး threading.Thread ကို parent ထားပါတယ်။ပြီးတော့ __init__ နဲ့ argument x ကို private variable self.__x ထဲကိုထည့်ပါတယ်။ပြီးတော့ threading.Thread ကို __init__() နဲ့ရေးထား တော့ ဒီ class ကိုခေါ်ခံရတာနဲ့ ဒီ threading.Thread ကလည်း အလုပ်လုပ်မှာဖြစ်ပါတယ်။ပြီးတော့ run() ထဲမှာ print self.__x ကိုအဖြေထုတ်ထားခဲ့ပါတယ်။ပြီးတော့ for loop နဲ့ threads 10 ခုကိုတည်ဆောက်ပြီး Class ကိုခေါ် x ကို passing လုပ်ပြီးပေးပြီး .start() နဲ့ thread တွေစတင်စေပါတယ်။run() က threading.Thread ထဲက function ဖြစ်ပါတယ် ဒါကြောင့်အထူးခေါ်စရာမလိုဘဲအလုပ်လုပ်ပါတယ်။

Threads တွေက တစ်ခါ run ပြီးရင် ရပ်ဖို့မလိုပါဘူး။နောက်ပြီး အချိန်နဲ့ပြုလုပ်ထားတဲ့အမျိုးစားတွေလဲ ဖြစ်နိုင်ပါတယ်။thread ကို အချိန် ဘယ်နှစက္ကန့်တိုင်းမှာ ပြန်ပြီး လုပ်နေမယ်ဆိုတာကိုသတ်မှတ်ထားနိုင်ပါတယ်။

Timed Threads

Python မှာ Timer class က Thread class ရဲ့ subclass ဖြစ်ပါတယ်။ဒါကသူတို့ရဲ့ ပြုမူပုံတွေက ဆင်တူတယ်လို့ဆိုလိုချင်တာပါ။ကျွန်တော်တို့က timer ကိုသုံးပြီး အချိန်နဲ့ timed threads တွေကို ဖန်တီး နိုင်ပါတယ်။Timer တွေကိုလဲ thread တွေလိုပါဘဲ စတင်ချင်ရင် .start() နဲ့စပြီးခေါ်ရပါတယ်။ကျွန်တော် တို့ 5 စက္ကန့်နေပြီးရင် thread တစ်ခုတည်ဆောက်တဲ့ပုံစံ program လေးရေးကြည့်ရအောင်။

```
from threading import *

def hello():
    print("Hello I am Thread")

t = Timer(5.0,hello)
t.start()
```

ဒီ program မှာ def နဲ့ method တစ်ခုတည်ဆောက်ပြီး အလုပ်တစ်ခုလုပ်ထားတယ် နောက်ပြီး t ဆိုတဲ့ object မှာ Timer() ကိုခေါ်ပြီး ပထမ args က သတ်မှတ်မယ့်အချိန်ကိုရေးခိုင်းပြီး ဘယ် function ကိုစတင်မလဲဆိုတာက ဒုတိယ args မှာရေးရပါတယ် ပြီးတော့ object t ကို start() နဲ့စတင်ခိုင်းထားပါတယ်။

ကျွန်တော်တို့ threads တွေကို အဆုံးမရှိ ထပ်ခါထပ်ခါလုပ်နေအောင်လို့ အသုံးပြုနိုင်ပါတယ်။ဘယ်လို လုပ်မလဲဆိုတာကို နမူနာ program နဲ့ကြည့်ရအောင်။

```
from threading import *
import time
def hello():
    while True:
        print("Hello World")
        time.sleep(3.0)

t = Timer(1.0,hello)
t.start()
```

ဒီ program လေးမှာ method တစ်ခုတည်ဆောက်ထားပြီး အဲဒီ method နာမည်က hello() ဖြစ်ပါတယ် သူ့ ထဲမှာ while True နဲ့အမြဲတမ်းလုပ်နေမယ့်အလုပ်ပါတယ် print("Hello World") ဖြစ်ပါတယ်။ဘယ်လို လုပ်မှာ လဲဆိုတော့ time.sleep() ကိုသုံးပြီး 3.0 စက္ကန့်တိုင်း ထပ်ခါထပ်ခါ Hello World ကိုလုပ်နေမှာဖြစ်ပါတယ်။ ပြီး တော့ ဒီ thread ကိုဖန်တီးဖို့ Timer(1.0,hello) ဆိုပြီးရေးထားပါတယ်။ဒါက 1 စက္ကန့်ကြာပြီးတာနဲ့ hello ဆိုတဲ့ thread က စတင်ပါတယ် 3 စက္ကန့်ခြားတစ်ခါလုပ်နေပါတယ် ကျွန်တော်တို့ program ကို ပိတ်မပြစ်မ ချင်းလုပ်နေပါတော့တယ်။

Android Information Tool's Thread

Android Information Tool မှာကျွန်တော် Thread တစ်ခုသုံးပြခဲ့ဖူးတာမှတ်မိမှာပါ။အဲဒီ thread ကိုရှင်းပြပါ့မယ်။ဒီ thread program က ကျွန်တော်ရေးတာမဟုတ်ပါဘူး နိုင်ငံခြားက Developer တစ်ယောက် ရေးထားတာဖြစ်ပါတယ်။

```
import threading,time
from threading import *
from tkinter import *

class thread(threading.Thread):

    def __init__(self,x=1):
        self.interval = x
        thread = threading.Thread(target=self.run, args=())
        thread.daemon = True
        thread.start()

    def run(self):
        while True:
            print("Hello World")
            time.sleep(self.interval)



if __name__ == "__main__":
    tk = Tk()
    tc = thread()
    tk.mainloop()
```

ဒီ program မှာ thread ဆိုတဲ့ class တစ်ခုကိုတည်ဆောက်ထားပြီး __init__ ပါဂင်ပါတယ်။ဒီထဲမှာ args တစ်ခု ပါဂင်ပါတယ် ဒီ args မှာ default value 1 ကိုထည့်ပေးထားလို့ ကျွန်တော်တို့က class ကိုခေါ်တဲ့အခါမှာ args မထည့်ပေးလဲရပါတယ်။ပြီးဗတာ့ thread object ကိုတည်ဆောက်တယ် threading.Thread() နဲ့ thread တစ်ခုကိုတည်ဆောက်လိုက်ပြီး target ဆိုတာကတော့ လုပ်ချင်တဲ့ function ကိုခေါ်လိုက်တာပါ args=() မှာ ဘာမှမထည့်ဘူး ပြီးတော့ run() မှာ while True: နဲ့ ဘယ်တော့မှမရပ်မယ့်လုပ်ဆောင်ချက်တစ်ခုထည့် တယ် ပြီးတော့ print("Hello World") လို့တည်ဆောက်လိုက်ပါတယ်။တကယ်တော့ Hello World နေရာမှာ ကျွန်တော်တို့ အမြဲလုပ်နေစေချင်တဲ့ အလုပ်ကိုထည့်ရေးထားတာပါ။ပြီးတော့ time.sleep() နဲ့ __init__ ရဲ့

args ကိုထည့်ပေးထားပါတယ်။thread.daemon= True ကတော့ daemonize လုပ်တာပါ True or False value ပေးနိုင်ပြီး start() နဲ့ thread မစတင်ခင်မှာ daemonize လုပ်ရပါတယ်။တကယ်လို့မပေးထားဘူးဆိုရင် သူ့ရဲ့ default value က false ဖြစ်နေပါလိမ့်မယ် မပါလဲဘာမှမဖြစ်ဘူး Runtime Error ကိုကာကွယ်တဲ့သဘော ဖြစ်ပါတယ်။ဒါကို ကျွန်တော်တို့ Tk windows Form ရဲ့ mainloop() ထဲမှာထည့် execute လုပ်တဲ့အခါမှာ အမြဲတမ်း Run နေတဲ့ Running Thread တစ်ခုရမှာဖြစ်ပါတယ်။

English to Myanmar Dictionary

ကျွန်တော်တို့ Python နဲ့ Words Dictionary တည်ဆောက်ပုံအကြောင်းကိုသွားရအောင် sqlite3 database ကိုအသုံးပြုပြီး Words Dictionary တစ်ခုတည်ဆောက်ပါမယ်။တည်ဆောက်တဲ့နေရာမှာ dictionary database တော့ရှိရမှာဖြစ်ပြီးတော့ အဲဒီ Database ထဲမှာ English စကားလုံးနဲ့ မြန်မာလို အဓိပ္ပါယ်တွဲရက်ရှိရမှာဖြစ်ပါတယ်။လက်တွေ့ဘဲတည်ဆောက်ရအောင်။

```
from tkinter import *
from tkinter import ttk as gui
from tkinter import scrolledtext as sc_text
from tkinter import messagebox as msgbox
import sqlite3
```

လိုအပ်တာတွေကို import လုပ်ပါတယ်။ပြီးတော့ ကျွန်တော်တို့ class တစ်ခုတည်ဆောက်ပြီး အဲဒီ class ရဲ့ constructor ထဲမှာ ကျွန်တော်တို့ win form တည်ဆောက်တွေပြုလုပ်ပါမယ်။

```
class interface:
    def __init__(self):
        self.win = Tk()
        self.win.title("English To Myanmar Dictionary")
        self.win.geometry("600x400")
        self.win.resizable(0,0)
        self.win.config(bg='#6098ff')
        self.add_widgets()
        self.database()
```

ဒီ __init__ ထဲမှာ လိုအပ်တဲ့ functions နှစ်ခုကိုခေါ်ထားပါသေးတယ် အဲဒါက ဘာလဲဆိုတော့ self.add_widgets နဲ့ self.database() ဖြစ်ပါတယ်။self.add_widgets() ကိုအရင်ဆုံးကြည့်ရအောင်။

```
def add_widgets(self):
    self.frame_1 = Frame(self.win,bg='#1a6bff',height=30)
    Label1 = Label(self.frame_1,text="WordsPy Eng-Mm
Dictionary",fg='#ffffff',bg='#1a6bff')
    Label1.config(font='AdobeGothicStd-Bold')
    Label1.grid()
    self.frame_1.pack(expand='no',fill='x',side=TOP)

    self.frame_2 = Frame(self.win,bg='#6098ff')
    self.entry = Entry(self.frame_2,width=84)
    self.entry.bind('<Return>',self.event_entry)
    self.entry.grid(column=0, row=0,padx=3,pady=5,sticky=NSEW)
    self.btn_se = gui.Button(self.frame_2,text="Search Word",command=self.showdata)
    self.btn_se.grid(column=1,row=0,padx=4,pady=5,sticky=NSEW)
    self.frame_2.pack(expand='no',fill='x',side=TOP)
```

```
    self.sctxt = sc_text.ScrolledText(self.win)
    self.sctxt.config(font='Zawgyi-One')
    self.sctxt.pack(expand=1,fill='x',side=TOP,padx=4,pady=5)
```

ဒီ function ကတော့ widgets တွေတည်ဆောက်ထားတဲ့ function ဖြစ်ပါတယ်။ကျွန်တော်တို့ Button တွေ Text Entry တွေ Frame တွေကိုနေရာချထားပါတယ်။ဒီမှာ မြန်မာစကားဖော်ပြဖို့အတွက် scrolledText ကို ကျွန်တော်တို့က font='Zawgyi-One' ကြေညာထားပါတယ်။ဒါလေးကိုမဖြစ်မနေလုပ်ရမယ် မဟုတ်ရင် မြန်မာစာပေါ်မှာမဟုတ်ဘူး။နောက်ထပ် တစ်ခုကတော့ database() တည်ဆောက်ထားတဲ့ function ဘဲဖြစ်ပါတယ်။

```
def database(self):
    try:
        self.con = sqlite3.connect('words_db.db')
        self.cursor = self.con.cursor()
        self.cursor.execute('CREATE TABLE IF NOT EXISTS words(word Text,mean Text)')
    except:
        msgbox._show('Error Message','Database Access Error')
```

ဒီ Class ကတော့ ဘာမှမဟုတ်ဘူး Database တစ်ခုတည်ဆောက်တယ်ပြီးတော့ words နာမည်နဲ့ Table တစ်ခုတည်ဆောက်ပြီးတော့ word နဲ့ mean ဆိုတဲ့ column နှစ်ခုကိုလည်းတည်ဆောက်လိုက်ပါတယ်။ တကယ်လို့ Database နဲ့မချိတ်မိဘူးဆိုရင် Error Message ပြန်ဖို့ messagebox ကို except နဲ့ဖမ်းထားပါသေးတယ်။ကျွန်တော်တို့ add_widgets() function ထဲမှာ Entry ကို bind နဲ့ event ဖမ်းထားသလို Serach Word ဆိုတဲ့ Button လေးကိုနှိပ်ရင်လုပ်မယ့် အလုပ်ကိုလည်း ကျွန်တော်တို့ ထည့်ထားပါသေးတယ်။အောက်မှာကြည့်လိုက်ပါ။

```
self.entry = Entry(self.frame_2,width=84)
self.entry.bind('<Return>',self.event_entry)
self.entry.grid(column=0, row=0,padx=3,pady=5,sticky=NSEW)
self.btn_se = gui.Button(self.frame_2,text="Search Word",command=self.showdata)
self.btn_se.grid(column=1,row=0,padx=4,pady=5,sticky=NSEW)
self.frame_2.pack(expand='no',fill='x',side=TOP)
```

entry bind

Button Command

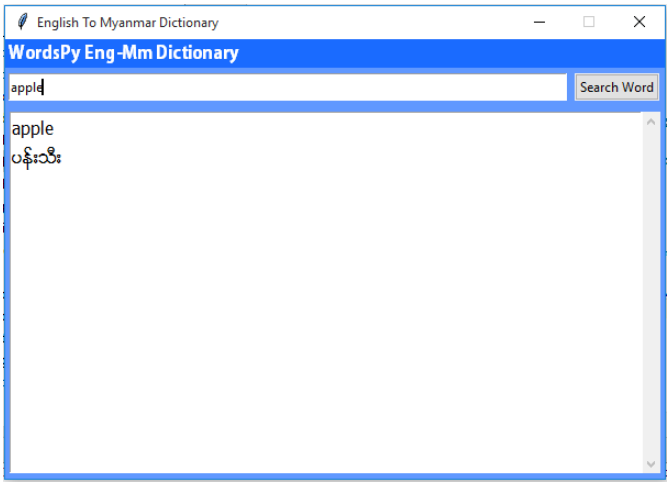
Self.entry.bind('<Return>',self.event_entry) ဆိုတာ Rentrun event ဆိုတာ Enter ကိုနှိပ်ရင် လုပ်မယ့် အလုပ်လို့ ကျွန်တော် အရှေ့ကEvent အခန်းမှာပြောပြီးပါပြီ။self.event_entry ဆိုတာကတော့ Entry widget ထဲမှာ Enter နှိပ်တဲ့အခါမှာ ခေါ်ဖို့အသုံးပြုတဲ့ function ဖြစ်ပါတယ်။အဲဒီ function ထဲမှာဘာတွေလုပ်ထားလဲကြည့်ရအောင်။

```
def event_entry(self,event):
    self.showdata()
```

ကျွန်တော် entry ထဲမှာ Enter နှိပ်ရင်ခေါ်မယ့် function ထဲမှာ self.showdata() ဆိုတဲ့ function ကိုခေါ်ထားပါတယ်။အဲဒီလိုဘဲ Button ကိုနှိပ်ရင်လဲ အဲဒီ function ကိုဘဲခေါ်ထားတာဖြစ်ပါတယ်။ဒါဆိုရင် အဲဒီ function ထဲမှာဘာလုပ်မလဲကြည့်ရအောင်။

```
def showdata(self):
    self.sctxt.delete(1.0,END)
    self.word_data = self.cursor.execute('SELECT * FROM words WHERE
word="%s"'%(self.entry.get()))
    for words_list in self.word_data:
        self.sctxt.insert(INSERT,words_list[0] +'\n'+words_list[1] +'\n')
```

showdata ထဲမှာ လုပ်ထားတဲ့အလုပ်က ပထမဆုံး self.sctxt.delete(1.0,END) ဆိုတာ ScrolledText ထဲမှာ ရှိတဲ့ စာသားတွေကိုအရင်ဆုံး clear လုပ်ရှင်းဖြစ်လိုက်တာဖြစ်ပါတယ်။ပြီးတော့ self.word_data ဆိုတဲ့ variable ထဲမှာ self.cursor.execute() နဲ့ database ထဲက ရတဲ့ data ကိုသိုလှောင်ဖို့တည်ဆောက်လိုက်ပါတယ်။SELECT * FROM words ဆိုတာ words ဆိုတဲ့ Table ထဲက အရာအားလုံးကို မှတ်သားလိုက်တယ်လို့ပြောတယ် ဘယ်လိုမှတ်သားတာလဲဆိုတော့ WHERE နဲ့ word = %s ဆိုပြီးဖြစ်ပါတယ်။WHERE ဆိုတာ Condition ပါဘဲ WHERE word = "apple" ဆိုရင် apple row မှာရှိတဲ့ data တွေကိုရွေးချယ်ယူမှာ ဖြစ်ပါတယ်။ဒီမှာတော့ %s နဲ့ self.entry.get() ထဲကဆိုတော့ Entry widget ထဲကိုရိုက်လိုက်တဲ့စာသားကို Database ထဲကရွေးထုတ်မယ်လို့ပြောတာဖြစ်ပါတယ်။ပြီးတော့ for loop ကိုသုံးပြီး words_list ထဲကို word_data ထဲက တန်ဖိုးတွေ Loop ပတ်ထုတ်လိုက်တယ်။ပြီးတော့ sctxt.insert နဲ့ scrolledText ထဲကို word_list[0] နဲ့ word_list[1] ရဲ့အဖြေတွေကိုထုတ်လိုက်တဲ့အခါမှာအောက်ပါအတိုင်းရမှာဖြစ်ပါတယ်။



ဒါဆိုရင် ကျွန်တော်တို့ Dictionary တစ်ခုကိုတည်ဆောက်နိုင်ပြီဖြစ်ပါတယ်။ကျွန်တော်တို့နောက်ဆုံး အခန်းဖြစ်တဲ့ .exe ဖိုင် သို့မဟုတ် execute ဖိုင်ထုတ်မယ့်အဆင့်ဖြစ်ပါတယ်။Windows ဆော့ဝဲလ်ကိုဖြစ်စေ Linux ကိုဖြစ်စေ execute ဖိုင်ရှိမှ ချက်ခြင်း Click နှိပ် Run နိုင်မှာဖြစ်ပါတယ်။ဒီ execute file တွေတည်ဆောက်ဖို့အတွက် PyInstaller ဆိုတဲ့ Python Module ကိုအသုံးပြုတည်ဆောက်မှာဖြစ်ပါတယ်။တခြား အသုံးဝင် Functions တွေကိုအသုံးပြုရင်းနဲ့ ကျွန်တော်တို့ Application တွေထုတ်ကြည့်ရအောင်။

.Py to Execute .EXE Creation Windows Application

ကျွန်တော်တို့ .py ဆိုတဲ့ python script ကနေ .exe ပြောရရင် ကျွန်တော်တို့ Windows ပေါ်မှာအသုံး ပြုနိုင်တဲ့ Application ဖန်တီးရမှာဖြစ်ပါတယ်။အဲဒီ Application ဖန်တီးဖို့အတွက် ကျွန်တော်တို့ Python module လေးတစ်ခုလိုပါတယ် အဲဒါကတော့ Pyinstaller ဖြစ်ပါတယ်။ထုံးစံအတိုင်း pip ကနေပြီးတော့ ဘဲ install လုပ်မယ်။

သူ့ရဲ့ website ကတော့ http://www.pyinstaller.org/ ဖြစ်ပါတယ်။

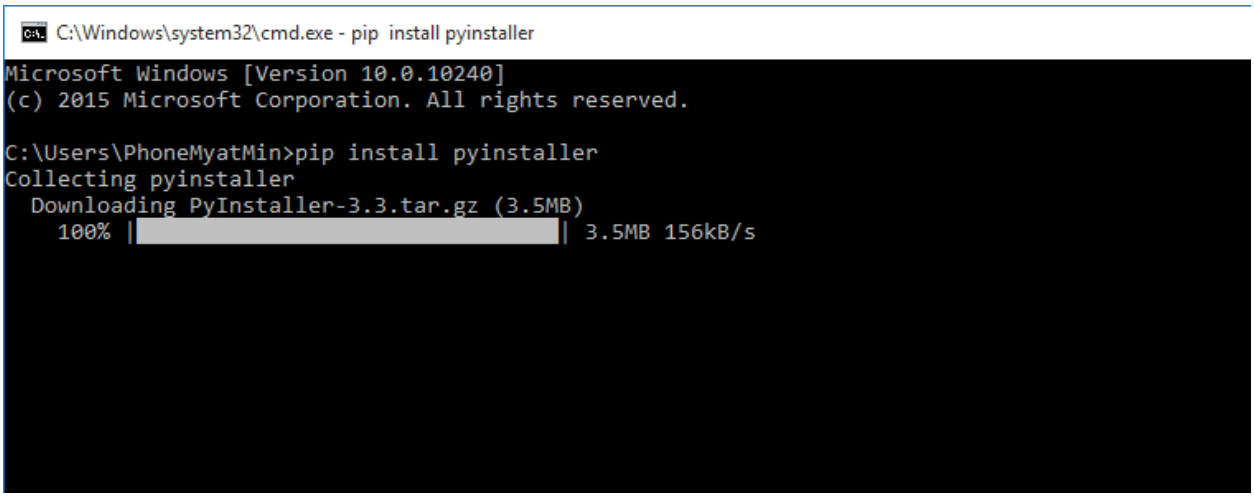
Features of Pyinstaller

- PyInstaller ဟာ Python Programs တွေကို standard executables အဖြစ်တည်ဆောက် ပေး ပါတယ်။ဥပမာ- .exe ပေါ့။ဒီ ဖိုင်တွေက computer ပေါ်မှာ click ဖွင့်ပြီးသုံးတန်းသုံးနိုင်တဲ့ဖိုင်မျိုး ဖြစ် ပါတယ်။ဒီ PyInstaller ရဲ့အားသာချက်တစ်ခုက သူ executable လုပ်ထားတဲ့ကဖိုင်တွေက Python Installed လုပ်မထားတဲ့ Computer တွေပေါ်မှာကိုတောင် အသုံးပြုလို့ရတာဖြစ်ပါတယ်။
- Multi-platform : Works Under
    1. Windows (32bit and 64bit)
    2. Linux(32bit and 64bit)
    3. Mac OS X(32bit and 64bit)
    4. Contributed Support for FreeBSD , Solaris,HPUX,AIX
- Multi-Python Version : Python 2.7 ကစပြီး Python 3 နဲ့အခုရောက်နေတဲ့ 3.6 အထိ အထောက်ပံ့ပေးပါတယ်။
- Flexible packaging mode:
    1. Single Directory: executable ကို Build တဲ့အခါမှာ directory တစ်ခုကိုတည်ဆောက်ပါ တယ်။အဲဒီအထဲမှာဘဲ executable file နဲ့အတူလိုအပ်တဲ့external binary modules တွေ ဖြစ်တဲ့ (.dll,.pyd,.so) စတာတွေပါဝင်ပါတယ်။
    2. Single File: executable file build တဲ့အခါမှာ တခြား external dependency တွေမလို အပ် ဘဲသူ့ဘာသာရပ်တည်နိုင်အောင်တည်ဆောက်ပေးထားပါတယ်။
    3. Custom:ကျွန်တော်တို့ဟာ PyInstaller ကိုအသုံးပြုပြီး packaging mode တွေကို အလိုရှိသလို ပြုလုပ်နိုင်ပါတယ်။
    4. သူ့မှာ လုပ်ဆောင်ချက်တွေကတော့ အများကြီးဖြစ်ပါတယ်။ဒါပေမယ့် ကျွန်တော်တို့ အား လုံး ကိုမကြည့်တော့ဘူး ကျွန်တော်တို့ရဲ့ Python Program ကို Computer အတွက် executable အဖြစ်ဘယ်လိုထုတ်မလဲဆိုတာဘဲကြည့်ရအောင်။

ဒါဆိုရင် အရင်ဆုံး pip ကိုသုံးပြီး PyInstaller ကို install လုပ်ရအောင်။

CMD or Terminal မှာ

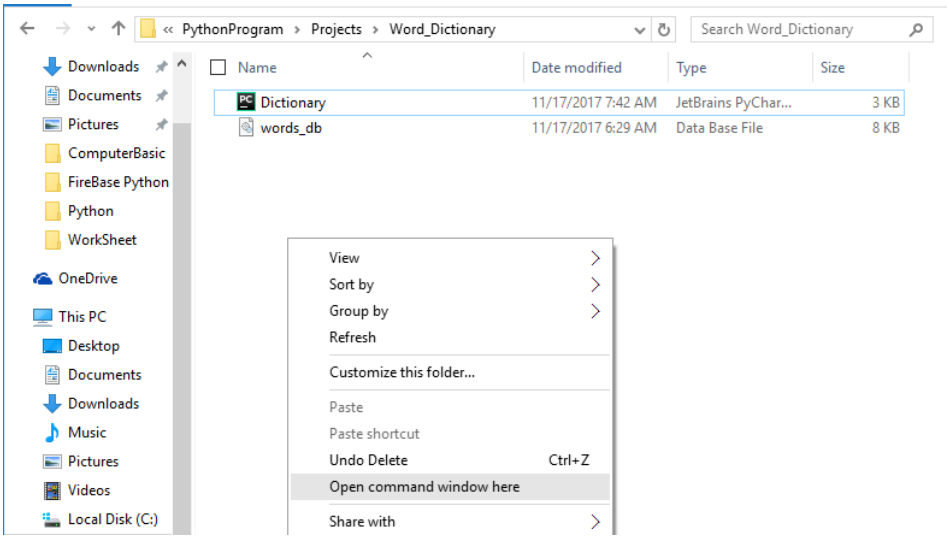
pip install pyinstaller လို့ရိုက်ပြီး Enter နှိပ်လိုက်ပါ။အင်တာနက်ရှိဖို့တော့လိုအပ်တယ်နော်။



အထက်ပါအတိုင်း install လုပ်သွားပါလိမ့်မယ်။ဒါဆိုရင် ကျွန်တော်တို့သုံးလို့ရပါပြီ။

PyInstaller ရဲ့အခြေခံလုပ်ဆောင်ချက်လေးတွေကိုရှင်းပြပါမယ်။.py ကနေပြီးတော့ executable ဖိုင်အဖြစ် ပြောင်းတယ်။အဲဒီလိုပြောင်းတဲ့နေရာမှာ Single File ဖြစ်အောင်ရယ်။ပြောင်းတဲ့ဖိုင်မှာ ကိုယ်လိုချင်တဲ့ Icon လေးဖြစ်ဖို့ရယ်။နောက်ထပ်က Console လို့ခေါ်တဲ့ နောက်ကနေ နောက်ခံ CMD or Terminal ကြီး ပေါ်မနေဖို့ရယ်လုပ်ဆောင်နိုင်ပါတယ်။ဒါဆိုရင်လုပ်ကြည့်ရအောင်။

အရင်ဆုံးကျွန်တော်တို့ရဲ့ .py file ကို executable ပြောင်းဖို့ဘယ်လိုလုပ်မလဲပြောရအောင်။အရင်ဆုံး ကျွန်တော်တို့ပြောင်းချင်တဲ့ .py file လေးရှိတဲ့နေရာမှာ shift + Right Click နှိပ်လိုက်ပါ။ကျွန်တော် ကတော့ English Myanmar Dictionary Program ကိုဘဲ .exe ပြောင်းပါမယ်။ဒါကြောင့်သူရှိတဲ့ Folder မှာ Shift ကို ဖိပြီး Right Click နှိပ်ပြီး CMD ကိုခေါ်လိုက်မယ်။ပြီးရင် Open Command Window Here ကိုနှိပ်ပါ။



ဒါဆိုရင် Dictionary Folder ထဲမှာ command windows ဖွင့်နိုင်ပြီပေါ့ဗျာ။အဲဒီထဲမှာ ရိုက်ရမယ့် command က pyinstaller yourpython.py ဖြစ်ပါတယ်။yourpython.py နေရာမှာ ကျွန်တော်တို့ .exe ပြောင်းလိုတဲ့ py ဖိုင်နာမည်ကိုထည့်ရပါ့မယ်။

```
pyinstaller Dictionary.py

D:\PythonCode\PythonProgram\Projects\Word_Dictionary>pyinstaller Dictionary.py
1362 INFO: PyInstaller: 3.3
1363 INFO: Python: 3.6.2
1363 INFO: Platform: Windows-10-10.0.10240-SP0
1371 INFO: wrote D:\PythonCode\PythonProgram\Projects\Word_Dictionary\Dictionary.spec
1372 INFO: UPX is not available.
1375 INFO: Extending PYTHONPATH with paths
['D:\\PythonCode\\PythonProgram\\Projects\\Word_Dictionary',
 'D:\\PythonCode\\PythonProgram\\Projects\\Word_Dictionary']
1376 INFO: checking Analysis
1376 INFO: Building Analysis because out00-Analysis.toc is non existent
1377 INFO: Initializing module dependency graph...
1387 INFO: Initializing module graph hooks...
1405 INFO: Analyzing base_library.zip ...
12187 INFO: running Analysis out00-Analysis.toc
12270 INFO: Adding Microsoft.Windows.Common-Controls to dependent assemblies of final executable
```

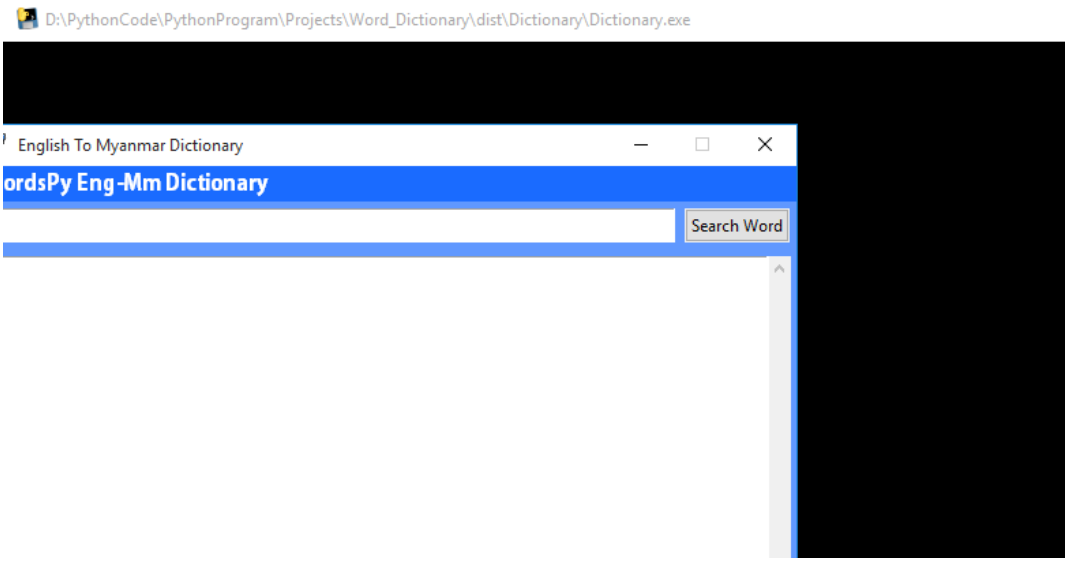
ဒီလိုရိုက်လိုက်ရင် executable file အဖြစ်ပြောင်းလဲနေတာကိုတွေ့ရမှာပါ။ပြီးတော့ ကျွန်တော်တို့ ရဲ့ py folder ထဲမှာ အောက်ပါအတိုင်း ဖိုင်နဲ့ဖိုဒါတချို့ပေါ်လာမှာပါ။

Name	Date modified	Type	Size
__pycache__	11/21/2017 9:15 AM	File folder	
build	11/21/2017 9:15 AM	File folder	
dist	11/21/2017 9:15 AM	File folder	
Dictionary	11/17/2017 7:42 AM	JetBrains PyChar...	3 KB
Dictionary.spec	11/21/2017 9:15 AM	SPEC File	1 KB
words_db	11/17/2017 6:29 AM	Data Base File	8 KB

အဲဒီထဲကမှ dist ထဲမှာရှိတဲ့ Application ကို Run ရပါမယ်။

_ssl	11/21/2017 9:15 AM	Python Extension ...	2,006 KB
_tkinter	11/21/2017 9:15 AM	Python Extension ...	61 KB
base_library	11/21/2017 9:15 AM	WinRAR ZIP archive	723 KB
Dictionary	11/21/2017 9:15 AM	Application	1,487 KB
Dictionary.exe.manifest	11/21/2017 9:15 AM	MANIFEST File	2 KB
pyexpat	11/21/2017 9:15 AM	Python Extension ...	185 KB
python36.dll	11/21/2017 9:15 AM	Application extens...	3,477 KB
select	11/21/2017 9:15 AM	Python Extension ...	20 KB

အထက်ကပုံပါအတိုင်း Dictionary ဆိုတဲ့ Application ကို Run လိုက်တဲ့အခါမှာ Application ပွင့်လာမှာ ဖြစ်ပါတယ်။ဒါပေမယ့်နောက်မှာ Console ကြီးပါနေတာကိုလဲတွေ့ရမှာပါ။



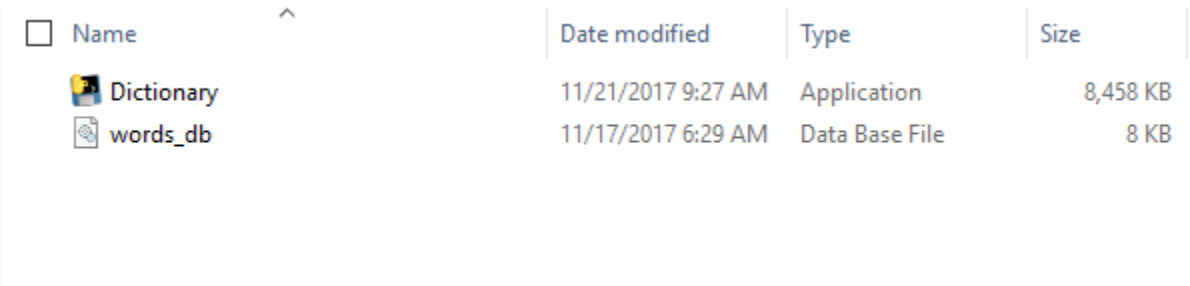
နောက်ပြီး သူ့အလိုလျောက်ပေးထားတဲ့ Application Icon ကြီးကိုလဲတွေ့ရပါတယ်။ဒီလို Console ဖျောက်တာနဲ့ကိုယ်ပိုင် Icon ပေးဖို့ PyInstaller မှာလုပ်နိုင်ပါတယ်။နောက်ပြီး ဒီ dicts folder ထဲမှာ ဖိုင်တွေ အများကြီးဖြစ်နေတာမဟုတ်ဘဲ single file အဖြစ် .exe ထုတ်လို့လဲရပါတယ်။

Single File Executable

Pyinstaller ကို Keyword တစ်ခုသုံးပြီး Single File Executable ပြုလုပ်နိုင်ပါတယ်။

pyinstaller -F yourpythonfile.py

အထက်က Command ကိုရိုက်ရမှာဖြစ်ပါတယ်။

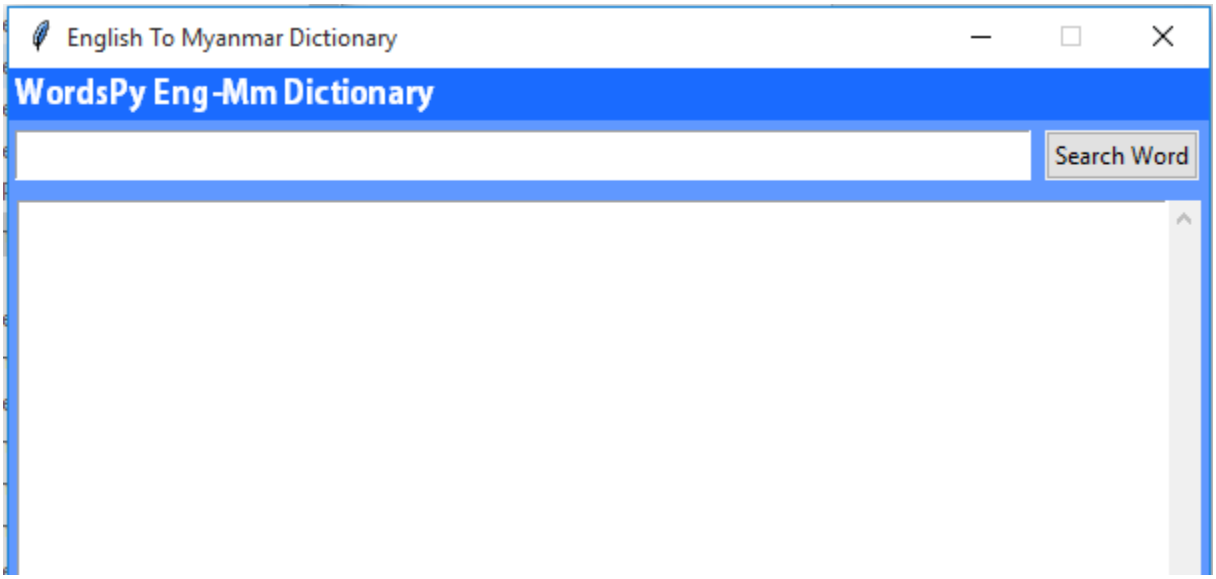


ဒါဆိုရင် dists folder ထဲမှာ အရင်လို ဖိုင်တွေအများကြီး .executable လုပ်ထားတာ မတွေ့ရဘဲ single file အနေနဲ့ဘဲတွေ့ရမှာပါ။ဒါဆိုရင်နောက်က console ကြီးဖျောက်ကြအုန်းစို့.....

No Console With Python Executable File

နောက်က console ကြီးဖျောက်ဖို့အတွက်လဲ ကျွန်တော်တို့ Keyword တစ်ခုကို အသုံးပြုတဲ့ command တစ်ခုကိုသုံးမှာပါ။အသုံးပြုရမယ့် command ကတော့

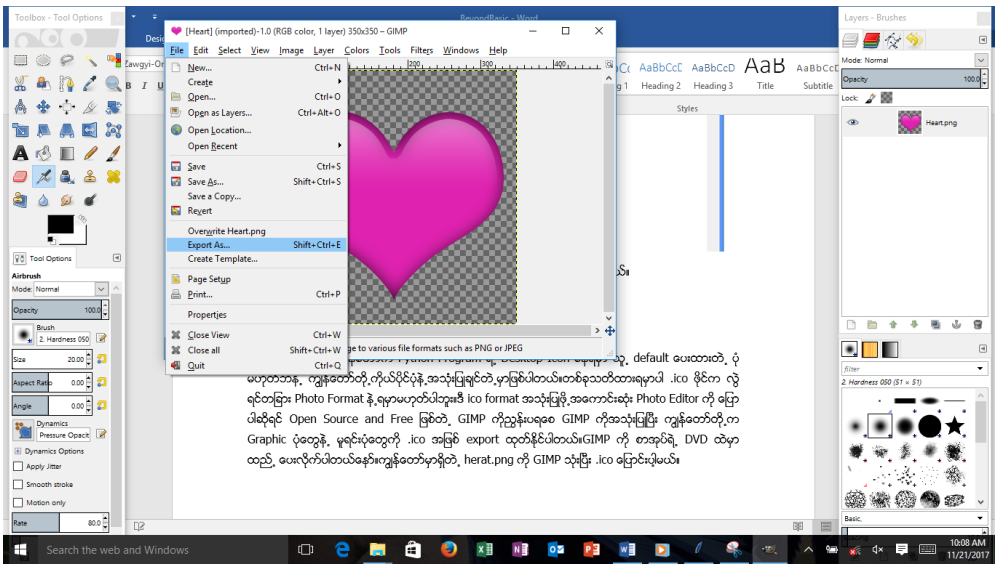
Pyinstaller –w yourpythonfile.py ဖြစ်ပါတယ်။



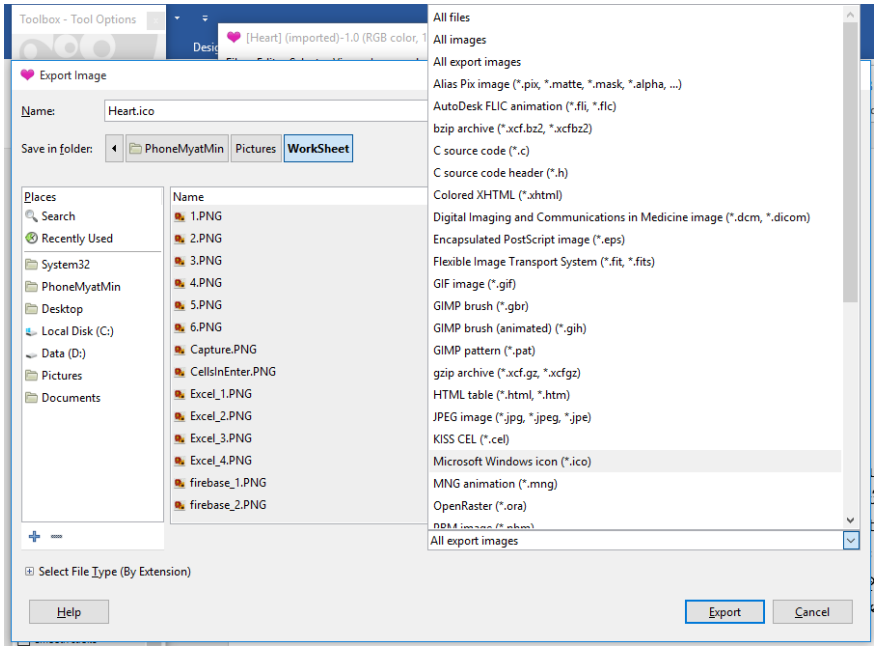
ဒါဆိုရင် ကျွန်တော်တို့ရဲ့ Python Program က console မပါဘဲတွေ့ရမှာဖြစ်ပါတယ်။

Set Icon In Python Executable Program

ဒီမှာတော့ ကျွန်တော်က Python Program ရဲ့ Desktop Icon နေရမှာ သူ့ default ပေးထားတဲ့ ပုံမဟုတ်ဘဲနဲ့ ကျွန်တော်တို့ကိုယ်ပိုင်ပုံနဲ့အသုံးပြုချင်တဲ့မှာဖြစ်ပါတယ်။တစ်ခုသတိထားရမှာပါ .ico ဖိုင်က လွဲရင်တခြား Photo Format နဲ့ရမှာမဟုတ်ပါဘူး။ဒီ ico format အသုံးပြုဖို့အကောင်းဆုံး Photo Editor ကို ပြောပါဆိုရင် Open Source and Free ဖြစ်တဲ့ GIMP ကိုညွှန်းပရစေ GIMP ကိုအသုံးပြုပြီး ကျွန်တော်တို့က Graphic ပုံတွေနဲ့ မူရင်းပုံတွေကို .ico အဖြစ် export ထုတ်နိုင်ပါတယ်။GIMP ကို စာအုပ်ရဲ့ DVD ထဲမှာ ထည့် ပေးလိုက်ပါတယ်နော်။ကျွန်တော်မှာရှိတဲ့ herat.png ကို GIMP သုံးပြီး .ico ပြောင်းပါ့မယ်။



File ထဲက Export As... ကိုနှိပ်ပါ။ပြီးရင် နာမည်နောက်က .png မှာ .ico လို့ပြောင်းပြီး image format မှာလဲ .ico ကိုရွေးပေးပါ။



ပြီးရင် ကျွန်တော်တို့ရဲ့ Program Folder ထဲကို .ico ပုံလေးကိုကူးထည့်လိုက်ပါ။သုံးရမယ့် command ကတော့ pyinstaller –i [ico file] [yourpythonfile.py] ဖြစ်ပါတယ်။ -i ကိုအသုံးပြုရပြီးတော့ ico file နေရာမှာ ကျွန်တော်တို့ပြုလုပ်ထားတဲ့ .ico ဖိုင်ရဲ့နာမည်ကိုထည့်ရမှာဖြစ်ပြီးတော့ အဲဒီနောက်မှာမှ ပြောင်းချင်တဲ့ .py ဖိုင်နာမည်ကိုထည့်ရမှာပါ။

Command တွေအားလုံးကိုစုပေါင်းလို့ရပါတယ်။

pyinstaller -w -F -i [ico file] [yourpythonfile.py]

ဖြစ်ပါတယ်။

ဒါဆိုရင် single file application လေးရမှာဖြစ်ပြီးတော့ console လည်းမပါသလို ကိုယ့်စိတ်ကြိုက် icon နဲ့ လည်းရမှာပါ။သတိပြုရမှာတစ်ချက်က icon ပြောင်းလိုက်လိုက်ခြင်း Application File က ပုံမပြောင်းပါဘူး restart ချမှပုံပြောင်းတတ်ပါတယ်။

ဒါဆိုရင်တော့ ကျွန်တော်တို့ Application တစ်ခုလုံးကိုတည်ဆောက်နိုင်ပြီဖြစ်ပါတယ်။ဒီ Beyond Basic Python Programming စာအုပ်လေးကိုလဲဒီမှာဘဲ အဆုံးသတ်မှာဖြစ်ပါတယ်။နောက်စာအုပ်တွေလဲကြိုးစားပြီး ရေးသားသွားမှာဖြစ်သလို အွန်လိုင်းကလည်း Free Course တွေကိုလည်း အွန်လိုင်းကနေ တင်ပေးမှာ ဖြစ်ပါတယ်။မိတ်ဆွေများအားလုံးကျန်းမာချမ်းသာပြီး လိုရာဆန္ဒပြည့်ဝလို့ သင်သမျှပညာရပါစေ။

ဘုန်းမြတ်မင်း