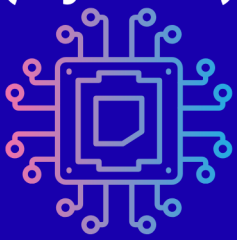
2026 Edition

The Future Coder Series

Computer Science စာအုပ်

Vol 4: Practical Programming (Python)



ရန်နိုင်လင်း (Software Engineer)

MBA, PG Dip (CS), BSc(Physics)

Chapter 1: Getting Started

Chapter 2: Variables & Data Types

Chapter 3: Control Structures

3.1 Creating a File (ဖိုင်အသစ် ပြင်ဆင်ခြင်း)

3.2 Comparison Operators (နှိုင်းယှဉ်ခြင်း သင်္ကေတများ)

3.3 Selection (If-Else) - ဆုံးဖြတ်ချက် ချခြင်း

3.4 Elif (ရွေးချယ်စရာ အများကြီးရှိရင်)

3.5 Loops (ထပ်ခါပြုလုပ်ခြင်း)

3.6 Lab Exercise: Number Guessing Game (ဂဏန်းမှန်းတဲ့ဂိမ်း)

Chapter 4: Data Structures

4.1 Creating a Workspace

4.2 Python Lists (စာရင်းများ)

4.3 Indexing (နေရာသတ်မှတ်ချက်) - အရေးကြီးသည်

4.5 Dictionaries (အဘိဓာန် ပုံစံ)

4.6 List of Dictionaries (အဆင့်မြင့် ဖွဲ့စည်းပုံ)

Chapter 5: GUI Programming (Tkinter)

5.1 What is GUI? (GUI ဆိုတာ ဘာလဲ?)

7.3 Loading Data Function (ဇယားကွက်တွင် ပြသခြင်း ရှင်းလင်းချက်)

7.4 Selecting & Editing (ရွေးချယ်ခြင်းနှင့် ပြင်ဆင်ခြင်း)

7.5 Deleting Function (ဖျက်ခြင်း ရှင်းလင်းချက်)

7.6 PDF Generation (PDF ထုတ်ခြင်း ရှင်းလင်းချက်)

Chapter 1: Getting Started

ဒီအခန်းမှာ ကွန်ပျူတာကို ပရိုဂရမ်ရေးဖို့ အဆင်သင့်ဖြစ်အောင် ပြင်ဆင်တာနဲ့ ပထမဆုံး Python ဖိုင်တစ်ခု တည်ဆောက်ပုံကို လေ့လာရပါမယ်။

1.1 Setting Up (ကွန်ပျူတာ ပြင်ဆင်ခြင်း)

ပရိုဂရမ် မရေးခင် မဖြစ်မနေ လိုအပ်တဲ့ ဆော့ဖ်ဝဲ (၂) ခုကို အရင်ဆုံး ထည့်သွင်းရပါမယ်။

1. **Python Interpreter (Language Translator):**
- ရေးလိုက်တဲ့ python code တွေကို computer နားလည်းအောင် ဘာသာပြန်ပေးမဲ့ program။
- python.org ဝက်ဘ်ဆိုက်ကိုသွားပါ။ **Download Python** ကိုနှိပ်ပြီး Install လုပ်ပါ။
- သတိပြုရန်: Install လုပ်တဲ့အခါ **"Add Python to PATH"** ဆိုတဲ့ အကွက်လေးကို အမှန်ခြစ်ခဲ့ဖို့ မမေ့ပါနဲ့။ (ဒါမှ ကွန်ပျူတာရဲ့ ဘယ်နေရာကမဆို Python ကို ခေါ်သုံးလို့ရမှာပါ)။
2. **VS Code (Code Editor):**
- ကုဒ်တွေကို အရောင်လေးတွေနဲ့ သေသေချာချာ ရေးဖို့ အတွက် Code Editor လိုပါတယ်။ Microsoft ကထုတ်တဲ့ **Visual Studio Code** က အကောင်းဆုံးပါပဲ။ (ကြိုက်တာသုံးနိုင်ပါတယ်။)
- code.visualstudio.com ကနေ Download ဆွဲပြီး Install လုပ်ပါ။

1.2 Creating Your Workspace (အလုပ်ခွင် တည်ဆောက်ခြင်း)

ကုဒ်တွေကို ဒီတိုင်း လျှောက်သိမ်းရင် ပျောက်ကုန်မှာစိုး လို့ Folder စနစ်တကျ ဆောက်ကြမယ်။

1. Desktop ပေါ်မှာ Right-click နှိပ်ပါ။
2. New > Folder ကို ရွေးပါ။
3. Folder နာမည်ကို **"Python_Lessons"** လို့ ပေးလိုက်ပါ။
4. VS Code ကို ဖွင့်ပါ။
5. File Menu ထဲက Open Folder ကိုနှိပ်ပြီး ခုနက ဆောက်ခဲ့တဲ့ **"Python_Lessons"** folder ကို ရွေးလိုက်ပါ။

1.3 Creating a Python File (.py ဖိုင် တည်ဆောက်ခြင်း)

ဒါက အရေးကြီးဆုံး အဆင့်ပါ။ Word ဖိုင်ဆိုရင် .docx၊ သီချင်းဖိုင်ဆိုရင် .mp3 နဲ့ ဆုံးသလိုပဲ၊ Python ဖိုင်တွေက **.py** နဲ့ ဆုံးရပါမယ်။

1. VS Code ရဲ့ ဘယ်ဘက်ထောင့်က New File icon လေးကို နှိပ်ပါ။ (သို့မဟုတ်) File > New File ကို နှိပ်ပါ။
2. ဖိုင်နာမည်ပေးတဲ့အခါ **hello.py** လို့ ပေးလိုက်ပါ။

- hello = ဖိုင်နာမည်
- .py = Python ဖိုင်ဖြစ်ကြောင်း ကွန်ပျူတာကို ပြောပြခြင်း (Extension)။

3. Enter ခေါက်လိုက်ရင် Python ဖိုင်လေး ပွင့်လာပါပြီ။

မှတ်ချက်: အကယ်၍ .py မပါဘဲ hello လို့ပဲ ပေးလိုက်ရင် ကွန်ပျူတာက ဒါကို ရိုးရိုး Text file လို့ပဲ ထင်သွားပြီး ကုဒ်တွေ အလုပ်မလုပ်တော့ပါဘူး။

1.4 Writing Your First Code (ကုဒ်စရေးခြင်း)

ပွင့်နေတဲ့ hello.py ဖိုင်ထဲမှာ အောက်ပါအတိုင်း စာရိုက်ထည့်ပါ။

```
print("Hello, Myanmar!")
```

- print = ထုတ်ပြပါ (စာလုံးအသေးနဲ့ ရေးရမယ်)။
- (...) = ကွင်းစကွင်းပိတ် ထဲမှာ ရေးရမယ်။
- "..." = စာသားဖြစ်တဲ့အတွက် မျက်တောင်အဖွင့်အပိတ် (Quotes) ထဲမှာ ထည့်ရေးရမယ်။

Saving (သိမ်းဆည်းခြင်း): စာရိုက်ပြီးရင် ကီးဘုတ်က Ctrl နဲ့ S ကို တွဲနှိပ်ပြီး Save လိုက်ပါ။ (ဖိုင်နာမည်ဘေးက အစက်ဝိုင်းလေး ပျောက်သွားရင် Save ပြီးပါပြီ)။

1.5 Running the Program (ပရိုဂရမ် မောင်းနှင်ခြင်း)

ရေးထားတဲ့ ကုဒ် အလုပ်လုပ်/မလုပ် စမ်းသပ်ကြမယ်။

နည်းလမ်း (၁) – Play Button: VS Code ရဲ့ ညာဘက်အပေါ်ထောင့်မှာရှိတဲ့ **တြိဂံပုံ Play Button (▶)** လေးကို နှိပ်လိုက်ပါ။

နည်းလမ်း (၂) – Terminal: အောက်က Terminal (အမည်းရောင် အကွက်) မှာ အောက်ပါအတိုင်း ရိုက်ထည့်ပြီး Enter ခေါက်ပါ။

```
python hello.py
```

ရလဒ် (Result): Terminal ထဲမှာ အောက်ပါအတိုင်း ပေါ်လာရပါမယ် -

```
Hello, Myanmar!
```

ဂုဏ်ယူပါတယ်။ သင်ဟာ ကွန်ပျူတာကို သင့်စကားနားထောင်
အောင် အောင်မြင်စွာ ခိုင်းစေနိုင်ခဲ့ပါပြီ။

1.6 Common Mistakes (အဖြစ်များသော အမှားများ)

ကျောင်းသားအသစ်တွေ ဖြစ်တတ်တဲ့ အမှားတွေကို သတိပြုပါ -

- ❌ print "Hello" (ကွင်းစကွင်းပိတ် မပါရင် မှားပါတယ် - Python 3 မှာ မရပါ)။
- ❌ Print("Hello") (P အကြီးနဲ့ ရေးရင် မှားပါတယ်၊ Python က အကြီး/အသေး ခွဲပါတယ် Case Sensitive)။
- ❌ hello (ဖိုင်နာမည်မှာ .py မပါရင် Run လို့ မရပါ)။

Chapter 2: Variables & Data Types

ဒီအခန်းမှာ ကွန်ပျူတာရဲ့ မှတ်ဉာဏ် (Memory) ကို စတင်အသုံးပြုပါမယ်။

2.1 Creating a New File (ဖိုင်အသစ် တည်ဆောက်ခြင်း)

Chapter 1 တုန်းက hello.py ဆောက်ခဲ့သလိုပဲ၊ ဒီအခန်းအတွက် ဖိုင်သစ်တစ်ခု ဆောက်ကြမယ်။

1. **VS Code** ကို ဖွင့်ပါ။
2. File > New File ကို နှိပ်ပါ။
3. ဖိုင်နာမည်ကို **variables.py** လို့ ပေးလိုက်ပါ။
- .py ပါမှ Python ဖိုင်မှန်း ကွန်ပျူတာက သိမှာနော်။
4. Desktop ပေါ်က Python_Lessons folder ထဲမှာပဲ Save လိုက်ပါ။

2.2 Variables (ကိန်းရှင်များ) ဆိုတာ ဘာလဲ?

Variable ဆိုတာ Data တွေကို ခေတ္တသိမ်းဆည်းထားတဲ့ **"ပုံး" (Box)** လေးတွေနဲ့ တူပါတယ်။ ပုံးတစ်ပုံးချင်းစီမှာ နာမည် (Label) တစ်ခုစီ ရှိပါတယ်။ variables.py ဖိုင်ထဲမှာ အောက်ပါအတိုင်း ရိုက်ထည့်လိုက်ပါ -

```
# နာမည် (Variable Name) = တန်ဖိုး (Value)
name = "Mg Mg"
age = 16

print(name)
```

```
print(age)
```

ရှင်းလင်းချက်:

- name = "Mg Mg" -> "Mg Mg" ဆိုတဲ့ စာသားကို name ဆိုတဲ့ ပုံးထဲ ထည့်လိုက်တယ်။
- age = 16 -> 16 ဆိုတဲ့ ဂဏန်းကို age ဆိုတဲ့ ပုံးထဲ ထည့်လိုက်တယ်။
- print(name) -> name ပုံးထဲက အရာကို ထုတ်ပြပါ (ဒီတော့ "Mg Mg" ထွက်လာမယ်)။

Run ကြည့်မယ်: Terminal မှာ အောက်ပါအတိုင်း ရိုက်ထည့်ပြီး Enter ခေါက်ပါ။

```
python variables.py
```

(Result: Mg Mg နဲ့ 16 ပေါ်လာပါလိမ့်မယ်)

2.3 Data Types (အချက်အလက် အမျိုးအစားများ)

ကွန်ပျူတာက "စာ" နဲ့ "ဂဏန်း" ကို မတူညီတဲ့ ပုံစံတွေနဲ့ သိမ်းဆည်းပါတယ် ။ အဓိက (၄) မျိုးကို သိထားရပါမယ်။

သင့်ရဲ့ variables.py ဖိုင်ထဲမှာရှိတဲ့ ကုဒ်အဟောင်းတွေကို ဖျက်ပြီး ဒါတွေ ကူးထည့်ပါ -

```
# 1. String (စာသား)
student_name = "Hla Hla"
# စာသားဆိုရင် "..." (Quotes) ထဲမှာ အမြဲထည့်ရမယ်
```

```
# 2. Integer (ကိန်းပြည့်)
student_age = 18
# ဂဏာန်းဆိုရင် Quotes ထည့်စရာ မလိုဘူး

# 3. Float (ဒသမကိန်း)
student_height = 5.6
# အစက်ပါတဲ့ ဂဏာန်းတွေပါ

# 4. Boolean (အမှန်/အမှား)
is_student = True
# True (မှန်) သို့မဟုတ် False (မှား) နှစ်မျိုးပဲ ရှိတယ်

# အားလုံးကို ထုတ်ကြည့်မယ်
print(student_name)
print(student_height)
print(is_student)
```

Save လုပ်ပြီး Run ကြည့်ပါ: (Ctrl+S နှိပ်၊ Terminal မှာ python variables.py ရိုက်)။

2.4 User Input (အသုံးပြုသူထံမှ လက်ခံခြင်း)

အခု ကျွန်တော်တို့က Data တွေကို ကုဒ်ထဲမှာ အသေထည့်ထားတာ (Hard Code) ဖြစ်နေတယ်။ အခု ဆော့ဖ်ဝဲအစစ်တွေလိုမျိုး အသုံးပြုသူ (User) ဆီကနေ တောင်းကြည့်မယ်။

input() ဆိုတဲ့ Command ကို သုံးရပါမယ်။ ဖိုင်ထဲမှာ အားလုံးဖျက်ပြီး ဒါကို ရိုက်ထည့်ပါ -

```
print("--- မိတ်ဆက် ပရိုဂရမ် ---")

# User ကို မေးခွန်းမေးပြီး ပြန်ဖြေတာကို name ဆိုတဲ့ Variable ထဲ ထည့်မယ်
name = input("မင်းနာမည် ဘယ်သူလဲ: ")

# နောက်တစ်ခု ထပ်မေးမယ်
color = input("ဘယ်အရောင် ကြိုက်လဲ: ")

# ပြန်ထုတ်ပြမယ်
print("မင်္ဂလာပါ " + name)
print("မင်းက " + color + " ရောင်ကို ကြိုက်တာပဲ!")
```

စမ်းသပ်ပုံ:

1. Terminal မှာ Run လိုက်ပါ။
2. မင်းနာမည် ဘယ်သူလဲ: လို့ ပေါ်လာရင် Terminal ထဲမှာတင် ကိုယ့်နာမည်ရိုက်ပြီး Enter ခေါက်ပါ။
3. အရောင်မေးရင် ရိုက်ထည့်ပြီး Enter ခေါက်ပါ။
4. ကွန်ပျူတာက ပြန်ပြောပါလိမ့်မယ်။

2.5 Type Conversion (အမျိုးအစား ပြောင်းလဲခြင်း) - အရေးကြီးသည်

input() နဲ့ လက်ခံလိုက်တဲ့ Data မှန်သမျှကို ကွန်ပျူတာက **စာသား (String)** အနေနဲ့ပဲ မှတ်ပါတယ်။ ဒါကြောင့် ဂဏန်းသချိၤာ တွက်ချင်ရင် **ဂဏန်း (Integer/Float)** ဖြစ်အောင် ပြောင်းပေးရပါမယ် ။

မှားယွင်းသော ဥပမာ (မလိုက်လုပ်ရ):

```
age = input("အသက် ဘယ်လောက်လဲ: ") # ဥပမာ 16 ရိုက်တယ်
next_year = age + 1              # Error တက်မယ်!
# ဘာလို့လဲဆိုတော့ "16" (စာသား) ကို 1 (ဂဏန်း) နဲ့ ပေါင်းလို့မရဘူး
```

မှန်ကန်သော နည်းလမ်း (Type Casting): ဖိုင်အသစ်တစ်ခု calculator.py ဆိုပြီး ဆောက်လိုက်ပါ။ ပြီးရင် ဒါကို ရိုက်ထည့်ပါ -

```
# အသက်တွက်တဲ့ ပရိုဂရမ်

# 1. မွေးသက္ကရာဇ် တောင်းမယ်
birth_year = input("မွေးသက္ကရာဇ် ရိုက်ထည့်ပါ (ဥပမာ- 2005): ")

# 2. စာသားကို ဂဏန်းပြောင်းမယ် (String to Integer)
birth_year_num = int(birth_year)

# 3. အသက်တွက်မယ် (လက်ရှိနှစ် - မွေးနှစ်)
current_year = 2025
age = current_year - birth_year_num
```

```
# 4. အဖြေထုတ်မယ်
# age က ဂဏန်းဖြစ်နေလို့ စာနဲ့တွဲချင်ရင် str() နဲ့ ပြန်ပြောင်းရတယ်
print("မင်းရဲ့ အသက်က " + str(age) + " နှစ် ဖြစ်ပါတယ်။")
```

ရှင်းလင်းချက်:

- int("2005") -> 2005 (ဂဏန်း ဖြစ်သွားတယ်)။
- str(20) -> "20" (စာသား ပြန်ဖြစ်သွားတယ်၊ print ထဲမှာ စာကြောင်းတွေနဲ့ ဆက်ဖို့အတွက်ပါ)။

2.6 Lab Exercise (လက်တွေ့လေ့ကျင့်ခန်း)

အခု သင်ခဲ့တာတွေကို ပေါင်းပြီး **"ဈေးဝယ် ဘေလ်တွက်စက်" (Simple Bill Calculator)** လေး ရေးကြည့်ရအောင်။

1. bill.py ဆိုတဲ့ ဖိုင်အသစ် ဆောက်ပါ။
2. အောက်ပါ အချက်အလက်တွေကို User ဆီက တောင်းပါ -
   - ပစ္စည်းအမည် (Item Name)
   - ဈေးနှုန်း (Price)
   - အရေအတွက် (Quantity)
3. ကျသင့်ငွေ စုစုပေါင်း (Total) ကို တွက်ချက်ပါ။
4. Screen ပေါ်မှာ လှလှပပ ပြန်ထုတ်ပြပါ။

နမူနာ အဖြေကုဒ်:

```
print("--- City Mart Bill Calculator ---")

item = input("ပစ္စည်းအမည်: ")
price = input("ဈေးနှုန်း (ကျပ်): ")
```

```
qty = input("အရေအတွက်: ")

# တွက်ချက်ဖို့ ဂဏန်းပြောင်းမယ် (Price က ဒသမဖြစ်နိုင်လို့ float သုံးမယ်)
total = float(price) * int(qty)

print("--------------------------")
print("ဝယ်ယူခဲ့သည့် ပစ္စည်း: " + item)
print("ကျသင့်ငွေ စုစုပေါင်း: " + str(total) + " ကျပ်")
print("--------------------------")
```

Chapter 3: Control Structures

3.1 Creating a File (ဖိုင်အသစ် ပြင်ဆင်ခြင်း)

ဒီအခန်းအတွက် ကုဒ်တွေရေးဖို့ ဖိုင်အသစ်တစ်ခု အရင်ဆောက်ကြမယ်။

1. **VS Code** မှာ File > New File ကိုနှိပ်ပါ။
2. ဖိုင်နာမည်ကို **logic.py** လို့ ပေးပြီး Save လိုက်ပါ။

3.2 Comparison Operators (နှိုင်းယှဉ်ခြင်း သင်္ကေတများ)

ဆုံးဖြတ်ချက် မချခင်မှာ၊ ကွန်ပျူတာ ဘယ်လို နှိုင်းယှဉ်သလဲ ဆိုတာ အရင်သိရပါမယ်။

- == : တူညီသည် (Equal to) သတိပြုရန် = တစ်ခုတည်းဆိုရင် တန်ဖိုးထည့်တာပါ၊ နှိုင်းယှဉ်ရင် နှစ်ခုသုံးရတယ်။

- != : မတူညီပါ (Not equal to)
- > : ထက်ကြီးသည် (Greater than)
- < : ထက်ငယ်သည် (Less than)
- >= : ကြီးသည် (သို့) တူသည်
- <= : ငယ်သည် (သို့) တူသည်

3.3 Selection (If-Else) - ဆုံးဖြတ်ချက် ချခြင်း

ကွန်ပျူတာကို လမ်းကြောင်းရွေးခိုင်းတာ ဖြစ်ပါတယ်။
Indentation (အရေးကြီးဆုံး အချက်): Python မှာ If အောက်က အလုပ်လုပ်မယ့် စာကြောင်းတွေကို **Space ၄ ချက် (Tab တစ်ချက်)** ခွာပြီး ရေးရပါတယ်။ ဒါမှ ဒီ If နဲ့ သက်ဆိုင်မှန်း ကွန်ပျူတာက သိမှာပါ။

logic.py ထဲမှာ အောက်ပါကုဒ်ကို ရိုက်ထည့်ပြီး Run ကြည့်ပါ -

```
# Password စစ်ဆေးခြင်း ပရိုဂရမ်

password = input("စကားဝှက် ရိုက်ထည့်ပါ: ")

if password == "1234":
    print("မှန်ကန်ပါတယ် (Access Granted)")
    print("Welcome to the System!")
else:
    print("မှားယွင်းနေပါတယ် (Access Denied)")
    print("ပြန်လည် ကြိုးစားကြည့်ပါ")
```

```
print("--- ပြီးဆုံး ---")
```

လေ့လာရန်:

- if နောက်မှာ : (Colon) ပါတာ သတိပြုပါ။
- print တွေကို ဘေးကပ်မရေးဘဲ အထဲကို **ရွှေ့ရေး (Indent)** ထား တာ တွေ့လား? အဲ့ဒါက "ဒီအခြေအနေမှန်မှ လုပ်မယ့်အလုပ်" လို့ ပြော တာပါ။
- နောက်ဆုံး print("--- ပြီးဆုံး ---") ကတော့ ဘေးကပ်ရေးထားတဲ့ အတွက် If နဲ့ မဆိုင်ဘဲ အမြဲတမ်း ပေါ်နေမှာ ဖြစ်ပါတယ်။

3.4 Elif (ရွေးချယ်စရာ အများကြီးရှိရင်)

လမ်းကြောင်းက ၂ ခုထက်မက (အောင်/ရှုံး သာမက ဂုဏ်ထူးပါ ပါလာ ရင်) elif (Else If) ကို သုံးရပါတယ်။

```
# အမှတ်စာရင်း စစ်ဆေးခြင်း

score = int(input("အမှတ်ထည့်ပါ (0-100): "))

if score >= 80:
    print("ဂုဏ်ထူးထွက်သည် (Distinction)")
elif score >= 40:
    print("စာမေးပွဲ အောင်သည် (Passed)")
else:
    print("စာမေးပွဲ ကျရှုံးသည် (Failed)")
```

- score >= 80 ကို အရင်စစ်မယ်။ မှန်ရင် "Distinction" ပြပြီး ပြီးသွား မယ်။

- မမှန်ဘူးဆိုမှ နောက်တစ်ဆင့် 40 ကျော်လား ဆက်စစ်မယ်။

3.5 Loops (ထပ်ခါပြုလုပ်ခြင်း)

အလုပ်တစ်ခုကို အကြိမ်ကြိမ် လုပ်ခိုင်းချင်ရင် ကုဒ်တွေ အများကြီး ကူးရေးစရာ မလိုပါဘူး။ Loop သုံးလိုက်ရင် ရပါပြီ။

ဖိုင်အသစ် **loops.py** ဆောက်ပြီး စမ်းကြည့်ရအောင်။

(A) For Loop (ရေတွက်ပြီး လုပ်ခြင်း)

သတ်မှတ်ထားတဲ့ အကြိမ်အရေအတွက်အတိုင်း လုပ်ဆောင်ပါတယ်။

```
# ၁ ကနေ ၁၀ အထိ ဂဏန်းထုတ်ပြမယ်
# range(1, 11) ဆိုတာ ၁ ကနေ ၁၀ အထိ (၁၁ မပါ) လို့ ဆိုလိုပါတယ်

for i in range(1, 11):
    print("နံပါတ်: " + str(i))
```

(B) While Loop (အခြေအနေပေးပြီး လုပ်ခြင်း)

အခြေအနေတစ်ခု မှန်ကန်နေသမျှ ကာလပတ်လုံး လုပ်နေမှာပါ။

```
# ဂဏန်း ၁ ရိုက်ထည့်မှ ရပ်မယ့် ပရိုဂရမ်

user_input = ""

while user_input != "1":
    user_input = input("ရပ်ချင်ရင် 1 ကို နှိပ်ပါ: ")
    print("မင်းရိုက်တာက: " + user_input)

print("ပရိုဂရမ် ရပ်သွားပါပြီ။")
```

3.6 Lab Exercise: Number Guessing Game (ဂဏန်းမှန်းတဲ့ဂိမ်း)

Chapter 3 တစ်ခုလုံးရဲ့ အနှစ်ချုပ်အနေနဲ့ ဂိမ်းလေးတစ်ခု ရေးကြည့်မယ်။ ကွန်ပျူတာက ဂဏန်းတစ်ခု (၁ပမာ ၇) ကို ပုက်ထားမယ်။ မင်းက မှန်အောင် ခန့်မှန်းရမယ်။

1. game.py ဆိုတဲ့ ဖိုင်အသစ် ဆောက်ပါ။
2. အောက်ပါကုဒ်ကို ရိုက်ထည့်ပါ -

```
import random # ကွန်ပျူတာကို ဂဏန်းအသစ်တွေ ထုတ်တတ်အောင် သင်ပေးတာ

print("--- ဂဏန်းမှန်း ဂိမ်း (Number Guessing Game) ---")

# ၁ ကနေ ၁၀ ကြား ဂဏန်းတစ်ခုကို ကွန်ပျူတာကို ရွေးခိုင်းမယ်
secret_number = random.randint(1, 10)
guess = 0 # ကစားသမားရဲ့ အဖြေကို သိမ်းဖို့ (ယာယီ ၀ ထားလိုက်မယ်)

# အဖြေမမှန်မချင်း မေးနေမယ် (While Loop)
while guess != secret_number:
    # 1. User ဆီက အဖြေတောင်းမယ်
    user_input = input("၁ မှ ၁၀ ကြား ဂဏန်းတစ်ခု မှန်းပါ: ")
    guess = int(user_input) # ဂဏန်းပြောင်းမယ်

    # 2. စစ်ဆေးမယ် (If-Else)
```

```
    if guess == secret_number:
        print("မှန်ပါတယ်! ဂုဏ်ယူပါတယ်!")
    elif guess > secret_number:
        print("များနေတယ်.. လျှော့ပါ")
    else:
        print("နည်းနေတယ်.. တိုးပါ")

print("--- ဂိမ်းပြီးဆုံး ---")
```

စမ်းသပ်ပုံ:

- Run လိုက်ပါ။ ကွန်ပျူတာက ဂဏန်းတစ်ခု စဉ်းစားထားပါလိမ့်မယ်။
- မင်းက 5 ရိုက်ကြည့်။ သူက "နည်းနေတယ်" လို့ ပြောရင် 6, 7, 8 ဆက် စမ်း။
- "မှန်ပါတယ်" လို့ ပြောတဲ့အထိ ဂိမ်းက ဆက်သွားနေပါလိမ့်မယ်။

Chapter 4: Data Structures

(အချက်အလက် ဖွဲ့စည်းပုံများ – Lists & Dictionaries)

4.1 Creating a Workspace

ဒီအခန်းအတွက် ဖိုင်အသစ်တစ်ခု အရင်ဆောက်ကြမယ်။

1. **VS Code** မှာ File > New File ကိုနှိပ်ပါ။
2. ဖိုင်နာမည်ကို **data_structures.py** လို့ ပေးပြီး Save လိုက်ပါ။

4.2 Python Lists (စာရင်းများ)

Variable တစ်ခုထဲမှာ အချက်အလက်တွေ အများကြီးကို တန်းစီပြီး သိမ်းထားချင်ရင် **List** ကို သုံးပါတယ်။ List တွေကို လေးထောင့်ကွင်း **[ ] (Square Brackets)** နဲ့ ရေးရပါတယ်။

data_structures.py ထဲမှာ အောက်ပါကုဒ်ကို ရိုက်ထည့်ပြီး Run ကြည့်ပါ -

```
# စျေးဝယ်စာရင်း (List)
shopping_list = ["Apple", "Bread", "Milk", "Eggs"]

print(shopping_list)
```

ရှင်းလင်းချက်:

- shopping_list ဆိုတဲ့ Variable တစ်ခုထဲမှာ ပစ္စည်း ၄ မျိုးလုံး ရောက်နေပါပြီ။

4.3 Indexing (နေရာသတ်မှတ်ချက်) - အရေးကြီးသည်

List ထဲက ပစ္စည်းတွေကို တစ်ခုချင်းစီ ပြန်ယူချင်ရင် သူတို့ရဲ့ **"နေရာနံပါတ်" (Index)** ကို သုံးရပါတယ်။ Python (နှင့် ကွန်ပျူတာအများစု) မှာ **နံပါတ်စဉ်ကို 0 ကနေ စရေပါတယ်**။

- Apple = 0
- Bread = 1
- Milk = 2
- Eggs = 3

ကုဒ်ဆက်ရေးကြည့်မယ် -

```
# ပထမဆုံး ပစ္စည်းကို ထုတ်မယ် (Index 0)
print("ပထမဆုံး ပစ္စည်း: " + shopping_list[0])

# တတိယ ပစ္စည်းကို ထုတ်မယ် (Index 2)
print("တတိယ ပစ္စည်း: " + shopping_list[2])
```

4.4 Modifying Lists (စာရင်းကို ပြုပြင်ခြင်း)

List ဆိုတာ အသေမဟုတ်ပါဘူး။ ထပ်ထည့်လို့ရတယ်၊ ဖျက်လို့ရပါတယ်။

```
# 1. အသစ်ထပ်ထည့်ခြင်း (.append)
shopping_list.append("Coffee")
print("ထပ်ထည့်ပြီးနောက်:", shopping_list)

# 2. ပြင်ဆင်ခြင်း (နေရာနံပါတ်ကို ခေါ်ပြီး ပြင်တာ)
shopping_list[1] = "Cake" # Bread နေရာမှာ Cake အစားထိုးလိုက်မယ်
print("ပြင်ဆင်ပြီးနောက်:", shopping_list)

# 3. ဖျက်ပစ်ခြင်း (.remove)
shopping_list.remove("Eggs")
print("ဖျက်ပြီးနောက်:", shopping_list)
```

4.5 Dictionaries (အဘိဓာန် ပုံစံ)

List က 0, 1, 2 နဲ့ မှတ်ရတာ ရှုပ်တယ်ဆိုရင်၊ ကိုယ်ပိုင်ခေါင်းစဉ် (Key) လေးတွေ နဲ့ မှတ်လို့ရတဲ့ **Dictionary** ကို သုံးနိုင်ပါတယ်။ Dictionary တွေကို တွန့်ကွင်း **{ } (Curly Braces)** နဲ့ ရေးရပါတယ်။

dictionaries.py ဆိုတဲ့ ဖိုင်သစ်တစ်ခု ဆောက်ပြီး စမ်းကြည့်ရအောင်။

```
# ကျောင်းသား အချက်အလက်
student = {
    "name": "Mg Mg",
    "age": 16,
    "major": "Computer Science",
    "passed": True
}

# Data ပြန်ယူချင်ရင် Key (ခေါင်းစဉ်) ကို သုံးရတယ်
print("ကျောင်းသားအမည်: " + student["name"])
print("မေဂျာ: " + student["major"])
```

List နဲ့ Dictionary ကွာခြားချက်:

- **List []:** အစဉ်လိုက် တန်းစီနေတာ လိုချင်ရင်သုံး (ဥပမာ - ဈေးဝယ်စာရင်း၊ သီချင်း Playlist)။
- **Dictionary {}:** ခေါင်းစဉ်နဲ့ တန်ဖိုး တွဲသိမ်းချင်ရင်သုံး (ဥပမာ - မှတ်ပုံတင် အချက်အလက်၊ ဖုန်းစာအုပ်)။

4.6 List of Dictionaries (အဆင့်မြင့် ဖွဲ့စည်းပုံ)

လက်တွေ့လုပ်ငန်းခွင် (Database) တွေမှာ ဒီနှစ်ခုကို တွဲသုံးလေ့ရှိပါတယ်။ "Dictionary တွေကို စုထားတဲ့ List" ပုံစံပါ။

```
# ကျောင်းသားများ စာရင်း (List အကြီးကြီးထဲမှာ Dictionary အသေးလေးတွေ ရှိမယ်)
all_students = [
    {"name": "Mg Mg", "age": 16},
    {"name": "Hla Hla", "age": 17},
    {"name": "Aung Aung", "age": 16}
]

# For Loop နဲ့ ပတ်ပြီး ထုတ်ကြည့်မယ်
print("--- ကျောင်းသားစာရင်း ---")

for s in all_students:
    # s ဆိုတာ တစ်လှည့်စီဝင်လာမယ့် Dictionary လေးတွေပါ
    print(s["name"] + " (အသက်: " + str(s["age"]) + ")")
```

4.7 Lab Exercise: To-Do List App (လုပ်စရာ စာရင်း မှတ်တမ်း)

ဒီအခန်းရဲ့ Project အနေနဲ့ **"To-Do List"** ဆော့ဖ်ဝဲလေး ရေးကြည့်မယ်။ User က လုပ်စရာတွေ ထည့်မယ်၊ ကြည့်မယ်၊ ဖျက်မယ်။

1. todo.py ဖိုင်အသစ် ဆောက်ပါ။
2. အောက်ပါကုဒ်ကို ရေးပါ -

```
todos = [] # စာရင်းသိမ်းဖို့ List အလွတ်တစ်ခု

print("--- To-Do List App ---")

while True:
    print("\n1. စာရင်းကြည့်မယ်")
    print("2. အသစ်ထည့်မယ်")
    print("3. ဖျက်မယ်")
    print("4. ထွက်မယ်")

    choice = input("နံပါတ် ရွေးပါ: ")

    if choice == "1":
        print("\n--- လုပ်စရာများ ---")
        # List ထဲမှာ ဘာမှမရှိရင်
        if len(todos) == 0:
            print("(ဘာမှ မရှိသေးပါ)")
        else:
            # ရှိတာတွေ အကုန်ထုတ်ပြမယ်
            for task in todos:
                print("- " + task)

    elif choice == "2":
        new_task = input("ဘာလုပ်မလဲ ရေးပါ: ")
        todos.append(new_task) # List ထဲထည့်မယ်
```

```
    print("ထည့်သွင်းပြီးပါပြီ!")

elif choice == "3":
    task_to_remove = input("ဖျက်ချင်တာ ရေးပါ: ")
    if task_to_remove in todos:
        todos.remove(task_to_remove) # List ထဲကဖျက်မယ်
        print("ဖျက်ပြီးပါပြီ!")
    else:
        print("အဲ့ဒီအရာ စာရင်းထဲမှာ မရှိပါ!")

elif choice == "4":
    print("တာ့တာ!")
    break # Loop ကို ရပ်မယ်

else:
    print("နံပါတ် ၁ ကနေ ၄ ပဲ ရွေးရမှာနော်!")
```

စမ်းသပ်ပုံ:

- Run လိုက်ပါ။
- 2 နှိပ်ပြီး "Homework လုပ်ရန်" ထည့်ပါ။
- 2 ထပ်နှိပ်ပြီး "Game ဆော့ရန်" ထည့်ပါ။
- 1 နှိပ်ပြီး ပြန်ကြည့်ပါ။
- 3 နှိပ်ပြီး "Homework လုပ်ရန်" ကို ဖျက်လိုက်ပါ။

Chapter 5: GUI Programming (Tkinter)

(မြင်ကွင်းစနစ် ဖန်တီးခြင်း)

5.1 What is GUI? (GUI ဆိုတာ ဘာလဲ?)

- **CLI (Command Line Interface):** ရှေ့အခန်းတွေမှာ လုပ်ခဲ့သလို စာ ရိုက်ပြီး ခိုင်းတာပါ။ (Hacker တွေ သုံးသလိုမျိုး)။
- **GUI (Graphical User Interface):** ယနေ့ ခေတ် Facebook, Microsoft Word တို့လို Window တွေ၊ Mouse နဲ့ နှိပ်လို့ရတဲ့ ခလုတ်တွေ ပါဝင်တဲ့ စနစ်ပါ။

Python မှာ GUI ရေးဖို့အတွက် **Tkinter** (Tk Interface) ဆိုတဲ့ Library ပါပြီး သား ဖြစ်ပါတယ်။ ဘာမှ ထပ် Install လုပ်စရာ မလိုပါဘူး။

5.2 Creating Your First Window)

1. VS Code မှာ **gui_app.py** ဆိုတဲ့ ဖိုင်အသစ်တစ်ခု ဆောက်ပါ။
2. အောက်ပါ ကုဒ်ကို ရိုက်ထည့်ပြီး Run ကြည့်ပါ -

```
import tkinter as tk

# 1. Main Window တည်ဆောက်ခြင်း
```

```
window = tk.Tk()

# 2. ခေါင်းစဉ် နှင့် အရွယ်အစား သတ်မှတ်ခြင်း
window.title("ကျွန်ုပ်၏ ပထမဆုံး GUI")
window.geometry("400x300")  # အကျယ် x အမြင့်

# 3. Window ကို မပိတ်မချင်း ပြထားရန် (Loop)
window.mainloop()
```

ရှင်းလင်းချက်:

- tk.Tk(): ဒါက Window အကြီး (Container) ကို စဆောက်လိုက်တာပါ။
- window.mainloop(): ဒါမပါရင် Window က ဖျတ်ခနဲ ပွင့်ပြီး ချက်ချင်း ပြန်ပိတ်သွားပါလိမ့်မယ်။ သူက "Close (X)" ခလုတ် မနှိပ်မချင်း Window ကို ဆက်ပြထားပေးပါတယ်။

5.3 Adding Widgets (အစိတ်အပိုင်းများ ထည့်သွင်းခြင်း)

Window အလွတ်ကြီး ရပြီဆိုတော့ သူ့ထဲမှာ စာတွေ၊ ခလုတ်တွေ ထည့်ကြမယ်။ အဲ့ဒါတွေကို **Widgets** လို့ ခေါ်ပါတယ်။

အသုံးအများဆုံး Widgets (၃) ခုကတော့ -

1. **Label:** စာသား ဖော်ပြဖို့ (Display Text)။
2. **Entry:** စာရိုက်ထည့်ဖို့ (Input Box)။
3. **Button:** နှိပ်ဖို့ (Clickable Button)။

Widgets ထည့်သွင်းပုံ (Pack System): Python မှာ ပစ္စည်းတွေကို Window ပေါ်တင်ဖို့ pack() ဆိုတဲ့ Command သုံးရပါတယ်။ သူက ပစ္စည်းတွေကို အပေါ်ကနေ အောက် တစ်ခုပြီးတစ်ခု စီပေးသွားပါတယ်။
ကုဒ်ကို ဒီလို ပြင်ရေးကြည့်ရအောင် -

```
import tkinter as tk

window = tk.Tk()
window.title("Widget စမ်းသပ်ခြင်း")
window.geometry("300x200")

# 1. Label (ခေါင်းစဉ်စာသား)
label = tk.Label(window, text="မင်္ဂလာပါ Python", font=("Arial", 14))
label.pack(pady=10) # pady ဆိုတာ အပေါ်အောက် နေရာချန်တာ (Padding Y)

# 2. Entry (စာရိုက်ကွက်)
entry = tk.Entry(window, width=30)
entry.pack(pady=5)

# 3. Button (ခလုတ်)
btn = tk.Button(window, text="နှိပ်ပါ", bg="blue", fg="white")
btn.pack(pady=20)

window.mainloop()
```

5.4 Making Buttons Work (ခလုတ်နှိပ်လျှင် အလုပ်လုပ်စေခြင်း)

အခု ခလုတ်ကို နှိပ်လို့ရပြီ။ ဒါပေမဲ့ နှိပ်ရင် ဘာမှ မဖြစ်သေးဘူး။ နှိပ်လိုက်တာနဲ့ တုံ့ပြန်မှုရအောင် **Function** တစ်ခုနဲ့ ချိတ်ဆက်ပေးရပါမယ်။

- command= ဆိုတဲ့ နေရာမှာ Function နာမည် ထည့်ပေးရပါတယ်။

ပရောဂျက် – Greeting App (နှုတ်ဆက်စက်): နာမည်ရိုက်ထည့်ပြီး ခလုတ်နှိပ်လိုက်ရင် "မင်္ဂလာပါ [နာမည်]" လို့ ပြောင်းသွားမယ့် ဆော့ဖ်ဝဲ ရေးမယ်။

```
import tkinter as tk

# --- Function ပိုင်း (အလုပ်လုပ်မည့်နေရာ) ---
def greet_user():
    # 1. Entry ထဲက စာကို ယူမယ် (.get)
    user_name = entry.get()

    # 2. Label စာသားကို လှမ်းပြင်မယ် (.config)
    result_label.config(text="မင်္ဂလာပါ " + user_name + " ရေ!")

# --- GUI ပိုင်း (အမြင်ပိုင်း) ---
window = tk.Tk()
window.title("Greeting App")
window.geometry("300x250")

# ခေါင်းစဉ်
title_label = tk.Label(window, text="သင့်နာမည် ရိုက်ထည့်ပါ:")
```

```
title_label.pack(pady=10)

# စာရိုက်ကွက်
entry = tk.Entry(window)
entry.pack(pady=5)

# ခလုတ် (command=greet_user နဲ့ ချိတ်လိုက်ပြီ)
btn = tk.Button(window, text="နှိပ်ပါ", command=greet_user)
btn.pack(pady=10)

# ရလဒ်ပြမည့်နေရာ (စစချင်း ဘာမှမရေးထားဘူး)
result_label = tk.Label(window, text="", fg="green", font=("Arial", 12, "
bold"))
result_label.pack(pady=10)

window.mainloop()
```

စမ်းသပ်ပုံ:

1. Run လိုက်ပါ။
2. Entry မှာ "Mg Mg" လို့ ရိုက်ပါ။
3. "နှိပ်ပါ" ခလုတ်ကို နှိပ်ပါ။
4. အောက်ဆုံးမှာ "မင်္ဂလာပါ Mg Mg ရေ!" ဆိုပြီး စိမ်းစိမ်းလေး ပေါ်လာပါ လိမ့်မယ်။

5.5 Lab Exercise: Simple Calculator (ဂဏန်းပေါင်းစက်)

ဒီအခန်းရဲ့ လက်တွေ့လေ့ကျင့်ခန်းအနေနဲ့ ဂဏန်း ၂ လုံးပေါင်းပေးတဲ့ ဆော့ဖ်ဝဲလေး ရေးကြည့်မယ်။

1. my_calculator.py ဖိုင်အသစ် ဆောက်ပါ။
2. အောက်ပါ လိုအပ်ချက်တွေ ပါအောင် ရေးပါ -

- Entry အကွက် (၂) ကွက် (ဂဏန်း ၂ လုံးစာ)။
- Button တစ်ခု (စာသားက "ပေါင်းမည်")။
- Button နှိပ်ရင် အဖြေပြမယ့် Label တစ်ခု။

အဖြေကုဒ် (Solution Code):

```
import tkinter as tk

def calculate_sum():
    # Entry ထဲက စာကိုယူပြီး ဂဏန်းပြောင်း (int)
    num1 = int(entry1.get())
    num2 = int(entry2.get())

    total = num1 + num2

    # အဖြေပြန်ပြ (str ပြန်ပြောင်းရမယ်)
    ans_label.config(text="အဖြေ: " + str(total))

window = tk.Tk()
window.title("Simple Adder")
window.geometry("300x300")

tk.Label(window, text="ပထမ ဂဏန်း:").pack(pady=5)
```

```
entry1 = tk.Entry(window)
entry1.pack()

tk.Label(window, text="ဒုတိယ ဂဏန်း:").pack(pady=5)
entry2 = tk.Entry(window)
entry2.pack()

btn = tk.Button(window, text="ပေါင်းမည် (+)", command=calculate_sum, bg="orange")
btn.pack(pady=20)

ans_label = tk.Label(window, text="အဖြေ: 0", font=("Arial", 16))
ans_label.pack()

window.mainloop()
```

Chapter 6: Database Connectivity (GUI နှင့် Database ချိတ်ဆက်ခြင်း)

6.1 Concept (သဘောတရား)

အရင်အခန်းတွေတုန်းက Data တွေကို Variable ထဲမှာပဲ သိမ်းခဲ့လို့ ပရိုဂရမ်ပိတ်လိုက်ရင် ပျောက်သွားပါတယ်။ အခုတော့ **"ထာဝရ သိမ်းဆည်းခြင်း"** (Persistent Storage) ကို လုပ်ဆောင်ပါမယ်။

- **Frontend (အရှေ့ပိုင်း):** Tkinter နဲ့ဆောက်ထားတဲ့ Form (User မြင်ရမယ့်အပိုင်း)။
- **Backend (အနောက်ပိုင်း):** SQLite Database (Data သိမ်းမယ့် အပိုင်း)။

6.2 Preparing Workspace (ဖိုင်အသစ် တည်ဆောက်ခြင်း)

1. **VS Code** မှာ File အသစ်တစ်ခု ယူပါ။
2. ဖိုင်နာမည်ကို **student_db.py** လို့ ပေးပြီး Save လိုက်ပါ။

6.3 Building the Backend (Database ပိုင်း တည်ဆောက်ခြင်း)

ပထမဆုံး Database နဲ့ Table ကို အရင်တည်ဆောက်ရပါမယ်။ ဒါကို သီးသန့် Function တစ်ခုအနေနဲ့ ရေးကြမယ်။
student_db.py ထဲမှာ အောက်ပါကုဒ်ကို ရိုက်ထည့်ပါ -

```
import sqlite3

# Database နှင့် Table တည်ဆောက်မည့် Function
```

```
def create_table():
    # 1. Database ချိတ်ဆက်ခြင်း (File မရှိရင် အသစ်ဆောက်မယ်)
    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()

    # 2. Table ဆောက်မယ် (ID, Name, Phone ပါဝင်မယ်)
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS students (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            phone TEXT
        )
    ''')

    # 3. သိမ်းဆည်းပြီး ပိတ်မယ်
    conn.commit()
    conn.close()

# ပရိုဂရမ် စစချင်း Table ဆောက်ခိုင်းလိုက်မယ်
create_table()
```

(မှတ်ချက်: ဒါကို Run လိုက်ရင် school.db ဆိုတဲ့ ဖိုင်လေးတစ်ခု Folder ထဲမှာ ပေါ်လာပါလိမ့်မယ်)

6.4 Connecting GUI to Database (ဖောင်မှတဆင့် Data ထည့်ခြင်း)

အခု GUI Form ဆောက်ပြီး၊ "Save" ခလုတ်နှိပ်ရင် Database ထဲထည့်မယ့် ကုဒ်ကို ဆက်ရေးပါမယ်။ ကုဒ်အပောင်းအောက်မှာပဲ ဆက်ရေးပါ။

```
import tkinter as tk
from tkinter import messagebox # သတိပေးစာ ပြဖို့အတွက်

# --- Function ပိုင်း (Save ခလုတ်နှိပ်ရင် လုပ်မည့်အလုပ်) ---
def save_data():
    # 1. Entry အကွက်တွေထဲက စာကို ယူမယ်
    user_name = entry_name.get()
    user_phone = entry_phone.get()

    # 2. စာမရိုက်ရသေးရင် သတိပေးမယ်
    if user_name == "" or user_phone == "":
        messagebox.showwarning("သတိပေးချက်", "ကျေးဇူးပြု၍ အချက်အလက် ပြည့်စုံအောင် ဖြည့်ပါ")
        return # ဒီနားမှာ ရပ်လိုက်မယ်

    # 3. Database ထဲ လှမ်းထည့်မယ်
    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()

    # SQL Command (Data ထည့်ခြင်း)
```

```
    cursor.execute("INSERT INTO students (name, phone)
VALUES (?, ?)", (user_name, user_phone))

    conn.commit()
    conn.close()

    # 4. အောင်မြင်ကြောင်း ပြမယ်၊ အကွက်တွေကို ရှင်းမယ်
    messagebox.showinfo("အောင်မြင်သည်", "စာရင်းသွင်းပြီးပါပြီ")
    entry_name.delete(0, tk.END)
    entry_phone.delete(0, tk.END)

# --- GUI ပိုင်း ---
window = tk.Tk()
window.title("ကျောင်းသားမှတ်ပုံတင်")
window.geometry("350x250")

# နာမည် စာရိုက်ကွက်
label1 = tk.Label(window, text="ကျောင်းသား အမည်:")
label1.pack(pady=5)
entry_name = tk.Entry(window)
entry_name.pack(pady=5)

# ဖုန်းနံပါတ် စာရိုက်ကွက်
label2 = tk.Label(window, text="ဖုန်းနံပါတ်:")
label2.pack(pady=5)
```

```
entry_phone = tk.Entry(window)
entry_phone.pack(pady=5)

# Save ခလုတ်
btn_save = tk.Button(window, text="သိမ်းဆည်းမည် (Save)", bg="green", fg="white", command=save_data)
btn_save.pack(pady=20)

window.mainloop()
```

စမ်းသပ်ပုံ:

1. Run လိုက်ပါ။
2. နာမည်နဲ့ ဖုန်းနံပါတ် ဖြည့်ပါ။
3. "Save" နှိပ်ပါ။
4. "စာရင်းသွင်းပြီးပါပြီ" ဆိုတဲ့ Message ပေါ်လာရင် အောင်မြင်ပါပြီ။ (Data က school.db ထဲ ရောက်သွားပါပြီ)။

6.5 Retrieving Data (Data ပြန်လည် ကြည့်ရှုခြင်း)

ထည့်လိုက်တဲ့ Data တွေကို ပြန်မြင်ရမှ စိတ်ချရမှာပါ။ ဒါကြောင့် "Show Data" ခလုတ်တစ်ခု ထပ်ထည့်ပြီး၊ နှိပ်လိုက်ရင် Terminal (အမည်းကွက်) မှာ စာရင်းထုတ်ပြအောင် လုပ်ကြမယ်။
အပေါ်က save_data function အောက်မှာ ဒီ show_data function ကို ထပ်ထည့်ပါ။

```
def show_data():
    conn = sqlite3.connect('school.db')
```

```
cursor = conn.cursor()

# Data အကုန်ဆွဲထုတ်မယ်
cursor.execute("SELECT * FROM students")
records = cursor.fetchall()

print("--- ကျောင်းသား စာရင်း ---")
for row in records:
    print("ID:", row[0], "| Name:", row[1], "| Phone:", row[2])

conn.close()
```

ပြီးရင် အောက်ဆုံးက GUI ပိုင်းမှာ ခလုတ်တစ်ခု ထပ်ထည့်ပေးပါ။

```
Python
# Show ခလုတ် (Save ခလုတ် အောက်နားမှာ ထည့်ပါ)
btn_show = tk.Button(window, text="စာရင်းကြည့်မည်",
command=show_data)
btn_show.pack(pady=5)
```

ပြန်စမ်းသပ်ပုံ:

1. Run ပါ။ Data အသစ် ၂ ယောက်လောက် ထပ်ထည့်ပါ။
2. "စာရင်းကြည့်မည်" ခလုတ်ကို နှိပ်ပါ။
3. VS Code အောက်ခြေက Terminal မှာ ကျောင်းသားစာရင်းတွေ တန်းစီပြီး ထွက်လာတာ တွေ့ရပါလိမ့်မယ်။

6.6 Lab Exercise: Product Inventory (ကုန်ပစ္စည်း စာရင်း သွင်း စနစ်)

ဒီအခန်းအတွက် လက်တွေ့ လေ့ကျင့်ခန်းပါ။ inventory.py ဖိုင်အသစ် ဆောက် ပြီး အောက်ပါအတိုင်း ဖန်တီးပါ။

1. **Database:** shop.db နာမည်နဲ့ ဆောက်ပါ။ products ဇယား မှာ name နဲ့ price ပါပါစေ။
2. **GUI:**

- "ပစ္စည်းအမည်" နဲ့ "ဈေးနှုန်း" ထည့်စရာ Textbox ၂ ခု ပါရမယ်။
- "Add Product" ခလုတ်နှိပ်ရင် Database ထဲရောက်ရမယ်။
- "View All" ခလုတ်နှိပ်ရင် Terminal မှာ စာရင်းပြရမယ်။

Chapter 7: Final Project - Complete Student Management System

(နိဂုံးချုပ် ပရောဂျက် – CRUD နှင့် PDF Report ပါဝင်သော စနစ်)

ကြိုတင်ပြင်ဆင်ရန်: PDF ထုတ်ဖို့အတွက် Python မှာ reportlab ဆို တဲ့ Library လိုအပ်ပါတယ်။ Terminal မှာ အောက်ပါအတိုင်း ရိုက် ပြီး Install လုပ်ပေးပါ။

```
pip install reportlab
```

ဒီဗားရှင်းမှာ လုပ်ဆောင်ချက် (၆) မျိုး ပါဝင်လာပါမယ် -

1. **Add:** အသစ်ထည့်ခြင်း။
2. **View:** ဇယားဖြင့် ကြည့်ခြင်း။
3. **Edit:** မှားနေတာကို ပြန်ပြင်ခြင်း။
4. **Delete:** မလိုချင်တာကို ဖျက်ခြင်း။
5. **Clear:** စာရိုက်ကွက်များကို ရှင်းလင်းခြင်း။
6. **PDF Export:** စာရင်းကို PDF ဖိုင်အဖြစ် ထုတ်ယူခြင်း။

7.1 Full Source Code (ကုဒ်အပြည့်အစုံ)

final_system.py ဆိုတဲ့ ဖိုင်အသစ်တစ်ခု ဆောက်ပြီး အောက်ပါကုဒ်ကို ရေးပါ။

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sqlite3
# PDF ထုတ်ရန် Library
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

# --- 1. Database ပိုင်း (Backend) ---
def init_db():
    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
    cursor.execute('''
```

```
        CREATE TABLE IF NOT EXISTS students (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            phone TEXT,
            major TEXT
        )
    ''')
    conn.commit()
    conn.close()

# --- 2. လုပ်ဆောင်ချက်များ (Functions) ---

def add_student():
    name = entry_name.get()
    phone = entry_phone.get()
    major = combo_major.get()

    if name == "" or phone == "":
        messagebox.showwarning("သတိပေးချက်", "အမည်နှင့် ဖုန်း
နံပါတ် ဖြည့်ပါ")
        return

    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO students (name, phone, major)
```

```
VALUES (?, ?, ?)",
                (name, phone, major))
    conn.commit()
    conn.close()

    messagebox.showinfo("Success", "သိမ်းဆည်းပြီးပါပြီ!")
    clear_form()
    load_data()

def delete_student():
    # ဖယ်ထဲမှာ ရွေးထားတဲ့အကြောင်း (Selected Row) ကို ရှာမယ်
    selected_item = tree.selection()

    if not selected_item:
        messagebox.showwarning("သတိပေးချက်", "ဖျက်လို
သော စာကြောင်းကို ရွေးပါ")
        return

    # အတည်ပြုချက် တောင်းမယ်
    confirm = messagebox.askyesno("အတည်ပြုရန်", "တကယ် ဖျက်မှာ
လား?")
    if confirm:
        # ရွေးထားတဲ့ Data ရဲ့ ID ကို ယူမယ်
        row_id = tree.item(selected_item)['values'][0]
```

```
        conn = sqlite3.connect('school.db')
        cursor = conn.cursor()
        cursor.execute("DELETE FROM students WHERE id
= ?", (row_id,))
        conn.commit()
        conn.close()

        messagebox.showinfo("Deleted", "ဖျက်ပြီးပါပြီ")
        clear_form()
        load_data()

def select_student(event):
    # ဒယားက အကြောင်းကို နှိပ်လိုက်ရင် Form ထဲမှာ စာတွေ ပြန်ပေါ်လာ
အောင် လုပ်တာ
    selected_item = tree.selection()
    if selected_item:
        row = tree.item(selected_item)['values']

        # Form ထဲကို Data ပြန်ထည့်မယ်
        entry_id.config(state='normal') # ID ကွက်ကို ဖွင့်မယ်
        entry_id.delete(0, tk.END)
        entry_id.insert(0, row[0])
        entry_id.config(state='disabled') # ပြန်ပိတ်မယ် (ID မပြင်စေချင်လို့)

        entry_name.delete(0, tk.END)
```

```
        entry_name.insert(0, row[1])

        entry_phone.delete(0, tk.END)
        entry_phone.insert(0, row[2])

        combo_major.set(row[3])

def update_student():
    # ID မရှိရင် (မရွေးထားရင်) ပြင်လို့မရဘူး
    student_id = entry_id.get()

    if student_id == "":
        messagebox.showwarning("သတိပေးချက်", "ပြင်ဆင်လို
သော စာကြောင်းကို ဇယားမှ ရွေးပါ")
        return

    name = entry_name.get()
    phone = entry_phone.get()
    major = combo_major.get()

    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
    cursor.execute("""
        UPDATE students
        SET name = ?, phone = ?, major = ?
```

```
        WHERE id = ?
    """, (name, phone, major, student_id))
    conn.commit()
    conn.close()

    messagebox.showinfo("Updated", "ပြင်ဆင်ပြီးပါပြီ")
    clear_form()
    load_data()

def export_pdf():
    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM students")
    rows = cursor.fetchall()
    conn.close()

    filename = "Student_List.pdf"
    c = canvas.Canvas(filename, pagesize=letter)

    # PDF ခေါင်းစဉ်
    c.setFont("Helvetica-Bold", 16)
    c.drawString(200, 750, "Student Management Report")

    # ခေါင်းစဉ်တန်း
    c.setFont("Helvetica-Bold", 12)
```

```
    c.drawString(50, 720, "ID")
    c.drawString(100, 720, "Name")
    c.drawString(300, 720, "Phone")
    c.drawString(450, 720, "Major")

    # Data များ ထည့်ခြင်း
    y_position = 700
    c.setFont("Helvetica", 12)

    for row in rows:
        c.drawString(50, y_position, str(row[0]))
        c.drawString(100, y_position, row[1])
        c.drawString(300, y_position, row[2])
        c.drawString(450, y_position, row[3])
        y_position -= 20 # တစ်ကြောင်းပြီးရင် အောက်ဆင်းမယ်

    c.save()
    messagebox.showinfo("PDF Generated", f"{filename} ထုတ်ပြီးပါပြီ!")

def clear_form():
    entry_id.config(state='normal')
    entry_id.delete(0, tk.END)
    entry_id.config(state='disabled')
    entry_name.delete(0, tk.END)
    entry_phone.delete(0, tk.END)
```

```
    combo_major.current(0)

def load_data():
    for row in tree.get_children():
        tree.delete(row)

    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM students")
    rows = cursor.fetchall()

    for row in rows:
        tree.insert("", tk.END, values=row)
    conn.close()

# --- 3. GUI ဝိုင်း (Frontend Design) ---
init_db()

window = tk.Tk()
window.title("Student Management System (Full Version)")
window.geometry("800x600")

# ခေါင်းစဉ်
tk.Label(window, text="ကျောင်းသား စီမံခန့်ခွဲ
မှု စနစ်", font=("Arial", 20, "bold"), fg="#2c3e50").pack(pady=10)
```

```
# Input Frame
frame_input = tk.Frame(window, bg="#ecf0f1", bd=2, relief="groove")
frame_input.pack(pady=10, padx=20, fill="x")

# ID (Hidden mainly, but shown for edit logic)
tk.Label(frame_input, text="ID:", bg="#ecf0f1").grid(row=0, column=0,
padx=10, pady=10)
entry_id = tk.Entry(frame_input, width=10, state='disabled') #
ID ကို လက်နဲ့ပြင်ခွင့်မပေးဘူး
entry_id.grid(row=0, column=1, padx=10)

tk.Label(frame_input, text="အမည်:", bg="#ecf0f1").grid(row=0, column
=2, padx=10)
entry_name = tk.Entry(frame_input, width=20)
entry_name.grid(row=0, column=3, padx=10)

tk.Label(frame_input, text="ဖုန်း:", bg="#ecf0f1").grid(row=1, column=0
, padx=10, pady=10)
entry_phone = tk.Entry(frame_input, width=20)
entry_phone.grid(row=1, column=1, padx=10)

tk.Label(frame_input, text="ဌ
ာ:", bg="#ecf0f1").grid(row=1, column=2, padx=10)
combo_major = ttk.Combobox(frame_input, values=["CS", "Engineerin
```

```
g", "Business", "Other"], width=17)
combo_major.current(0)
combo_major.grid(row=1, column=3, padx=10)

# Buttons Frame
frame_btn = tk.Frame(window)
frame_btn.pack(pady=10)

btn_add = tk.Button(frame_btn, text="Add
Student", bg="#27ae60", fg="white", width=12, command=add_studen
t)
btn_add.grid(row=0, column=0, padx=5)

btn_update = tk.Button(frame_btn, text="Update", bg="#f39c12", fg="
white", width=12, command=update_student)
btn_update.grid(row=0, column=1, padx=5)

btn_delete = tk.Button(frame_btn, text="Delete", bg="#c0392b", fg="w
hite", width=12, command=delete_student)
btn_delete.grid(row=0, column=2, padx=5)

btn_clear = tk.Button(frame_btn, text="Clear
Form", bg="#7f8c8d", fg="white", width=12, command=clear_form)
btn_clear.grid(row=0, column=3, padx=5)
```

```
btn_pdf = tk.Button(frame_btn, text="Export
PDF", bg="#2980b9", fg="white", width=15, command=export_pdf)
btn_pdf.grid(row=0, column=4, padx=20)

# Table View
columns = ("ID", "Name", "Phone", "Major")
tree = ttk.Treeview(window, columns=columns, show="headings", heig
ht=10)

tree.heading("ID", text="စဉ်")
tree.heading("Name", text="အမည်")
tree.heading("Phone", text="ဖုန်းနံပါတ်")
tree.heading("Major", text="ဓမ္မာ")

tree.column("ID", width=50, anchor="center")
tree.column("Name", width=250)
tree.column("Phone", width=150)
tree.column("Major", width=150)

tree.pack(pady=10, padx=20, fill="both", expand=True)

# ဇယားထဲက တစ်ခုခုကို နှိပ်ရင် select_student function အလုပ်လုပ်မယ်
tree.bind('<<TreeviewSelect>>', select_student)
```

```
load_data()
window.mainloop()
```

7.2 Adding Data Function (အချက်အလက် ထည့်သွင်းခြင်း ရှင်းလင်းချက်)

add_student() function ဟာ Form ထဲက အချက်အလက်တွေကို Database ထဲ ပို့ဆောင်ပေးတဲ့ တံတား ဖြစ်ပါတယ်။

```
def add_student():
    # 1. GUI ကနေ Data ယူခြင်း
    name = entry_name.get()
    phone = entry_phone.get()
    major = combo_major.get()

    # 2. စစ်ဆေးခြင်း (Validation)
    if name == "" or phone == "":
        messagebox.showwarning("သတိပေးချက်", "အမည်နှင့် ဖုန်းနံပါတ် ဖြည့်ပါ")
        return

    # 3. Database ထဲ ထည့်ခြင်း
    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
```

```
    # SQL Injection ကာကွယ်ရန် (?) များကို အသုံးပြုသည်
    cursor.execute("INSERT INTO students (name, phone, major)
VALUES (?, ?, ?)",
                   (name, phone, major))
    conn.commit()
    conn.close()

    # 4. ပြီးဆုံးကြောင်း ပြသခြင်း
    messagebox.showinfo("Success", "သိမ်းဆည်းပြီးပါပြီ!")
    clear_form() # ဖောင်ကို ရှင်းမယ်
    load_data()  # ဇယားကို Refresh လုပ်မယ်
```

- **entry.get()**: Textbox ထဲမှာ User ရိုက်ထားတဲ့ စာကို Python ထဲ ဆွဲယူတာပါ။
- **VALUES (?, ?, ?)**: Data တွေကို SQL ကုဒ်ထဲ တိုက်ရိုက်မရေးဘဲ၊ ? နေရာမှာ အစားထိုးထည့်တာက Hacker ရန်က ကာကွယ်ဖို့ (Security) အတွက် အရေးကြီးပါတယ်။
- **load_data()**: Data အသစ်ထည့်လိုက်ရင် ဇယားကွက်မှာ ချက်ချင်း ပေါ်မလာပါဘူး။ Database ကို ပြန်ဖတ်ပြီး ဇယားကွက်ကို အသစ်ပြန်ဆွဲပေးဖို့ ဒီ Function ကို ပြန်ခေါ်ရတာပါ။

7.3 Loading Data Function (ဇယားကွက်တွင် ပြသခြင်း ရှင်းလင်းချက်)

load_data() function က Database ထဲက Data တွေကို Treeview (ဇယားကွက်) ပေါ် တင်ပေးတာပါ။

```
def load_data():
    # 1. ဇယားရှင်းလင်းခြင်း
    for row in tree.get_children():
        tree.delete(row)

    # 2. Data ဆွဲထုတ်ခြင်း
    conn = sqlite3.connect('school.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM students")
    rows = cursor.fetchall()

    # 3. ဇယားကွက်ထဲ ထည့်ခြင်း
    for row in rows:
        # row ဆိုတာ (1, "Mg Mg", "09..", "CS") ဆိုတဲ့ Tuple လေးပါ
        tree.insert("", tk.END, values=row)
    conn.close()
```

- **tree.get_children()**: ဇယားထဲမှာ လက်ရှိရှိနေတဲ့ စာကြောင်းတွေ အကုန်လုံးကို ယူလိုက်တာပါ။

- **tree.delete(row)**: Data အသစ်မထည့်ခင် အဟောင်းတွေကို အကုန် ဖျက်ပစ်တာပါ။ မဖျက်ရင် "Mg Mg" က နှစ်ခါ၊ သုံးခါ ထပ်ပြီး ပေါ်နေပါ လိမ့်မယ်။
- **tk.END**: Data အသစ်ကို ဇယားရဲ့ **အောက်ဆုံး** မှာ သွားဆက်မယ် လို့ ဆိုလိုတာပါ။

7.4 Selecting & Editing (ရွေးချယ်ခြင်းနှင့် ပြင်ဆင်ခြင်း)

Data ပြင်ဖို့အတွက် အရင်ဆုံး ဇယားထဲက လူကို နှိပ်လိုက်ရင် Form ထဲမှာ စာ တွေ ပြန်ပေါ်လာအောင် လုပ်ရပါတယ်။

Select လုပ်ခြင်း (select_student):

```
def select_student(event):
    selected_item = tree.selection() # Mouse နဲ့ နှိပ်ထားတဲ့ အကြောင်း
ကို ယူမယ်
    if selected_item:
        row = tree.item(selected_item)['values']

        # Form အကွက်တွေထဲကို Data ပြန်ထည့်ပေးမယ်
        entry_name.delete(0, tk.END)
        entry_name.insert(0, row[1]) # Name
        # ... other ...
```

Update လုပ်ခြင်း (update_student):

```
cursor.execute("""
    UPDATE students
    SET name = ?, phone = ?, major = ?
    WHERE id = ?
""", (name, phone, major, student_id))
```

WHERE id = ?: ဒါ အရေးကြီးဆုံးပါ။ နာမည်တူတဲ့ "Mg Mg" တွေ အများကြီး ရှိနိုင်ပေမယ့် "ID နံပါတ်" က တစ်ယောက်တည်း ရှိပါတယ်။ ဒါကြောင့် ပြင်ဆင်တဲ့ အခါ ID ကို ကိုင်ပြီး ပြင်မှ လူမှန်မှာ ဖြစ်ပါတယ်။

7.5 Deleting Function (ဖျက်ခြင်း ရှင်းလင်းချက်)

```
def delete_student():
    # 1. Confirm လုပ်ခြင်း
    confirm = messagebox.askyesno("အတည်ပြုရန်", "တကယ် ဖျက်မှာ
လား?")

    if confirm:
        # 2. Database မှ ဖျက်ခြင်း
        cursor.execute("DELETE FROM students WHERE
id = ?", (row_id,))
        # ...
```

messagebox.askyesno: မှားပြီး ဖျက်မိတာမျိုး မဖြစ်ရအောင် Yes/No ဘောက်စ်လေး တက်လာပြီး မေးခိုင်းတာပါ။ User က Yes နှိပ်မှ အလုပ်ဆက်လုပ်ပါမယ်။

7.6 PDF Generation (PDF ထုတ်ခြင်း ရှုင်းလင်းချက်)

PDF ထုတ်ဖို့ reportlab ကို သုံးထားပါတယ်။ သူ့ရဲ့ သဘောတရားက စာရွက်ဖြူပေါ်မှာ ပုံဆွဲသလိုပါပဲ။

```
c = canvas.Canvas(filename, pagesize=letter)
    y_position = 700 # စာရေးမည့် ဒေါင်လိုက် နေရာ (အပေါ်ကနေ စမယ်)

    for row in rows:
        c.drawString(50, y_position, str(row[0]))  # ID
        c.drawString(100, y_position, row[1])      # Name

        y_position -= 20 # တစ်ကြောင်းရေးပြီးရင် 20 units အောက်ဆင်းမယ်
```

- **Coordinates (x, y):** (0, 0) က စာရွက်ရဲ့ ဘယ်ဘက်အောက်ခြေထောင့် ဖြစ်ပါတယ်။ y တန်ဖိုး များလေလေ စာရွက်ရဲ့ အပေါ်ပိုင်းကို ရောက်လေလေ ဖြစ်ပါတယ်။
- **Looping:** Data တစ်ယောက်ချင်းစီအတွက် y_position ကို လျှော့လျှော့ပေးသွားတော့ စာရင်းက အောက်ကို တန်းစီပြီး ကျလာတာ ဖြစ်ပါတယ်။

*"ဤစာအုပ်များသည် တက္ကသိုလ်များတွင် သင်ကြားပို့ချနေသော Computer Science သင်ရိုးညွှန်းတမ်း (Curriculum) များကို အခြေခံထားပါသည်။ သို့သော် ယနေ့ခေတ် နည်းပညာများနှင့် ကိုက်ညီစေရန်အတွက် Python ဘာသာစကား၊ ခေတ်မီ Hardware နည်းပညာများနှင့် လက်တွေ့လုပ်ငန်းခွင်သုံး ဥပမာများကို ဖြည့်စွက် ပြုပြင်ရေးသားထားခြင်း ဖြစ်ပါသည်။

မူရင်းသင်ရိုးကို ပြုစုခဲ့ကြသော ဆရာ/ဆရာမ များအားလုံးကို လေးစားစွာဖြင့် Credit ပေးပါသည်။"*

Yan Naing Lin
Software Engineer