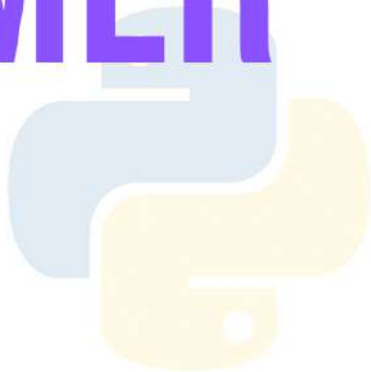
CODE with Thura

“ Hello World မှသည် လုပ်ငန်းခွင်သို့

ကိုယ်တိုင်လေ့လာ

PROGRAMMER



အခြေခံမှ စ၍ လက်တွေ့ လုပ်ငန်းခွင်အထိ၊
ကိုယ်တိုင် လေ့လာလိုသူများအတွက် ပြီးပြည့်စုံသော စာအုပ်။

CODE WITH THURA

ကိုယ်တိုင်လေ့လာ

Programmer

“Hello World မှသည် လုပ်ငန်းခွင်သို့

အခြေခံမှ စ၍ လက်တွေ့ လုပ်ငန်းခွင်အထိ၊

ကိုယ်တိုင် လေ့လာလိုသူများအတွက် ပြီးပြည့်စုံသော စာအုပ်။

Thura (Code with Thura)

ကိုယ်တိုင်လေ့လာ Programmer

“Hello World မှသည် လုပ်ငန်းခွင်သို့”

By: Thura (Code with Thura)

Year: 2025

စာရေးသူအကြောင်း

- ပညာရေးနယ်ပယ်နှင့် နည်းပညာလုပ်ငန်း နယ်ပယ်နှစ်ခုလုံးတွင် လုပ်ကိုင်နေသော IT အင်ဂျင်နီယာ တစ်ဦးဖြစ်သည်။
- နည်းပညာတက္ကသိုလ် (မန္တလေး) ၌ IT အင်ဂျင်နီယာဘာသာရပ်ကို အထူးပြုသင်ယူခဲ့ပြီး ၂၀၁၇ ခုနှစ် နောက်ပိုင်းတွင် Development နယ်ပယ်သို့ စတင်ဝင်ရောက်ခဲ့သည်။
- လုပ်ငန်းအတွေ့အကြုံ လေးနှစ်ကျော်ရှိခဲ့ပြီးနောက် 'Code with Thura' ကို စတင်တည်ထောင်ခဲ့သည်။
- လက်ရှိတွင် နည်းပညာနယ်ပယ်၌ Researcher တစ်ဦးအဖြစ် ပါဝင်ဆောင်ရွက်လျက်ရှိပြီး Development Project များနှင့် သင်ကြားရေးလုပ်ငန်းများကို လုပ်ကိုင်လျက်ရှိသည်။

ဆက်သွယ်လိုပါက thura.00011@gmail.com သို့လည်းကောင်း၊ 'Code with Thura' ၏ တရားဝင် ဝက်ဘ်ဆိုဒ် စာမျက်နှာဖြစ်သည့် https://www.codewiththura.com/contact သို့လည်းကောင်း ဝင်ရောက် ဆက်သွယ် မေးမြန်းနိုင်ပါသည်။

Contents

အခန်း (၁)

Programming မိတ်ဆက်

Programming ဆိုတာ ကွန်ပျူတာကို ခိုင်းစေဖို့အတွက် ညွှန်ကြားချက် (Instruction) တွေ ရေးသားရတဲ့ လုပ်ငန်းစဉ်ဖြစ်ပါတယ်။ ညွှန်ကြားချက်တွေဆိုတာ ကွန်ပျူတာကို လုပ်ဆောင်စေချင်တဲ့ အရာတွေကိုဆိုလိုတာပါ။ ဥပမာ၊ ဖိုင်တွေ အဖွင့်အပိတ်လုပ်တာ၊ အင်တာနက်ပေါ်ကနေ သတင်းအချက်အလက်တွေ ရယူတာ စသည်ဖြင့်ပါ။ အဆိုပါ ညွှန်ကြားချက် (Instruction) တွေကို ကုဒ်တွေ အသုံးပြုပြီး ရေးသားပေးရပါတယ်။ လွယ်လွယ်ပြောရရင် ကုဒ်တွေဆိုတာ ကွန်ပျူတာနဲ့ လူတွေကြား အပြန်အလှန်ဆက်သွယ်ပေးတဲ့ ဘာသာစကားတွေပါပဲ။ အဆိုပါ ကုဒ်တွေကို ပရိုဂရမ်မာတွေက မိမိနှစ်သက်ရာ Programming Language တွေအသုံးပြုပြီး ရေးသားကြတာဖြစ်ပါတယ်။

အရင်ကဆိုရင် Low-level Programming Language တွေကိုသာ အသုံးပြုရတဲ့အတွက် ပရိုဂရမ်တွေ ရေးသားရတာ မလွယ်ကူခဲ့ပါဘူး။ Low-level Programming Language ဆိုတာ ကွန်ပျူတာစနစ်တစ်ခုလုံးကို (သုည သို့မဟုတ် တစ်ကိုပဲ အခြေခံပြီး) တိုက်ရိုက် အနီးကပ် ထိန်းချုပ် ညွှန်ကြားရတာဖြစ်လို့ ခက်ခဲရှုပ်ထွေးပါတယ်။ အခုအချိန်မှာတော့ High-level Programming Langauge တွေ ပေါ်ထွက်လာခဲ့တဲ့အတွက် ပရိုဂရမ်တွေ ရေးသားရတာ လွယ်ကူလာခဲ့ပါပြီ။ High-Level Language တွေက အင်္ဂလိပ်စကားနဲ့ အနီးစပ်ဆုံးတူညီတဲ့အတွက် ဖတ်ရှုနားလည်ရ လွယ်ကူစေသလို ရေးသားရတာလည်း မြန်ဆန်စေပါတယ်။ နမူနာအနေနဲ့ Low-level Language သုံးပြီးရေးထားတဲ့ ပရိုဂရမ်တစ်ခုကို ဖော်ပြထားပါတယ်။

```
section .data
    msg db "Hello, World!", 0

section .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, 13
    int 0x80

    mov eax, 1
    xor ebx, ebx
```

```
    int 0x80
```

ရလဒ်တူညီတဲ့ ပရိုဂရမ်တစ်ခုကို High-level Language သုံးပြီးရေးတဲ့အခါ အခုလို ရရှိမှာဖြစ်ပါတယ်။

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

နမူနာနှစ်ခုကို ယှဉ်လေ့လာကြည့်လိုက်မယ်ဆိုရင် ဒီနေ့ခေတ် Programming Language တွေကို ရေးသား၊ လေ့လာရတာ အင်မတန် လွယ်ကူ ရိုးရှင်းတာကို တွေ့ရမှာဖြစ်ပါတယ်။ ဒီစာအုပ်ထဲမှာတော့ Python Programming Language ကို အသုံးပြုပြီး Programming သဘောတရားတွေကို အခြေခံကနေ တဆင့်ချင်းလေ့လာသွားမှာ ဖြစ်ပါတယ်။ Python ဟာ ရေးသား၊ ဖတ်ရှုရ လွယ်ကူပြီး လူသုံးများတဲ့ High-level Language တစ်ခုဖြစ်ပါတယ်။

Python မိတ်ဆက်

Python ကို ၁၉၉၁ ခုနှစ်မှာ Dutch ပရိုဂရမ်မာ **Guido van Rossum** ဆိုသူက ဖန်တီးခဲ့တာပါ။ သူက "Monty Python's Flying Circus" ဆိုတဲ့ ဟာသပြဇာတ်အစီအစဉ်ကို သဘောကျတဲ့အတွက်၊ အဲ့ဒီနာမည်ကို အသုံးပြုပြီး "Python" လို့ နာမည်ပေးခဲ့တာပါ။ ပရိုဂရမ်မာတွေဟာ ကုဒ်ရေးရတဲ့ အချိန်တွေထက် ရေးပြီးသားကုဒ်တွေကို ပြန်ဖတ်ရတဲ့ အချိန်တွေက ပိုများပါတယ်၊ ဒါကြောင့် Van Rossum က ဖတ်ရှုရလွယ်ကူတဲ့ Programming Language တစ်ခုအဖြစ် ဖန်တီးဖို့ ရည်ရွယ်ခဲ့တာ ဖြစ်ပါတယ်။ Python ဟာ ကမ္ဘာပေါ်မှာ လူကြိုက်အများဆုံး Programming Language တွေထဲက တစ်ခု ဖြစ်ပါတယ်။ Operating System မျိုးစုံပေါ်မှာ အလုပ်လုပ်နိုင်တဲ့အပြင် Web Server, Desktop Application, Mobile Application, Data Science, AI စတဲ့ နေရာတွေမှာ အသုံးပြုလျက်ရှိပါတယ်။ ဒါကြောင့် လက်ရှိ အိုင်တီနယ်ပယ်မှာ Python Programmer တွေအတွက် အခွင့်အလမ်းတွေ အများအပြား ရှိနေတာပဲ ဖြစ်ပါတယ်။

Python ထည့်သွင်းခြင်း

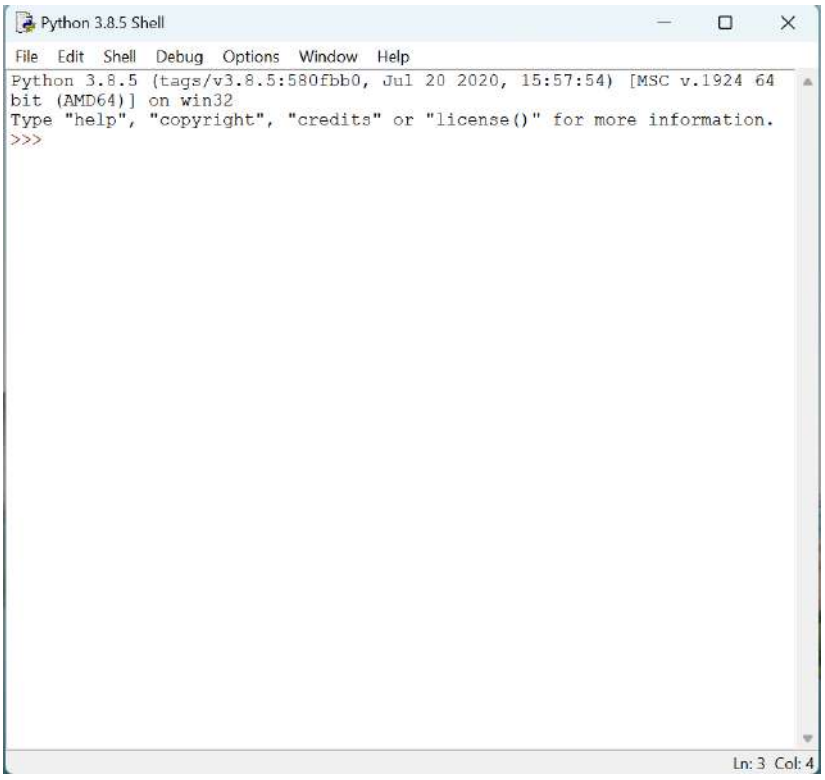
စာအုပ်ထဲမှာ ဖော်ပြထားတဲ့ နမူနာကုဒ်တွေကို စမ်းသပ်ရေးသားဖို့အတွက် Python ကို အသုံးပြုသွားမှာပါ။ Python ကို ထည့်သွင်းဖို့အတွက် http://python.org/downloads မှာ ဝင်ရောက် ရယူနိုင်ပါတယ်။ အကယ်၍ စာဖတ်သူက Ubuntu ကို အသုံးပြုသူဆိုရင်တော့ Python ကို ထပ်မံ ထည့်သွင်းဖို့ မလိုအပ်တော့ပါဘူး၊ Default အနေနဲ့ ပါဝင်နေပြီးသားဖြစ်ပါတယ်။

Python ကို 32-bit ကွန်ပျူတာတွေအတွက်ရော၊ 64-bit ကွန်ပျူတာတွေအတွက်ပါ ရယူနိုင်ပါတယ်။ မိမိ ကွန်ပျူတာနဲ့ သင့်တော်မယ့် အမျိုးအစားကို ရွေးချယ်ပြီး ထည့်သွင်းနိုင်ပါတယ်။

ထည့်သွင်းနည်း အပြည့်အစုံကို ဒီဗီဒီယိုမှာ ဝင်ရောက်ကြည့်ရှုနိုင်ပါတယ်။ ခက်ခဲရှုပ်ထွေးတာမျိုး မရှိလို့ စာအုပ်မှာ အကျယ်တဝင့် မဖော်ပြတော့ပါဘူး။ မိမိစက်ထဲက Version နဲ့ စာအုပ်ထဲက Version နှစ်ခု တူညီမှုရှိမရှိကိုတော့ သတိပြုဖို့ လိုအပ်ပါတယ်။

Python ကို ထည့်သွင်းလိုက်ရင် IDLE လို့ခေါ်တဲ့ ပရိုဂရမ်တစ်ခုကိုပါ တခါတည်း တွဲဖက် ထည့်သွင်းသွားမှာဖြစ်ပါတယ်။

ကွန်ပျူတာ Search ထဲမှာ IDLE လို့ ရှာ၊ Enter နှိပ်ပြီး ဖွင့်လိုက်ပါ။ အဲဒီအခါ ပုံ (၁.၁) အတိုင်း မြင်တွေ့ရမှာ ဖြစ်ပါတယ်။ ဒီပရိုဂရမ်လေးကို Interactive Shell လို့ခေါ်ပါတယ်။ အဲဒီထဲမှာ Python ကုဒ်တွေကို တိုက်ရိုက် ရေးချနိုင်ပြီး ရလဒ်တွေကို တခါတည်း ဖော်ပြပေးသွားမှာ ဖြစ်ပါတယ်။

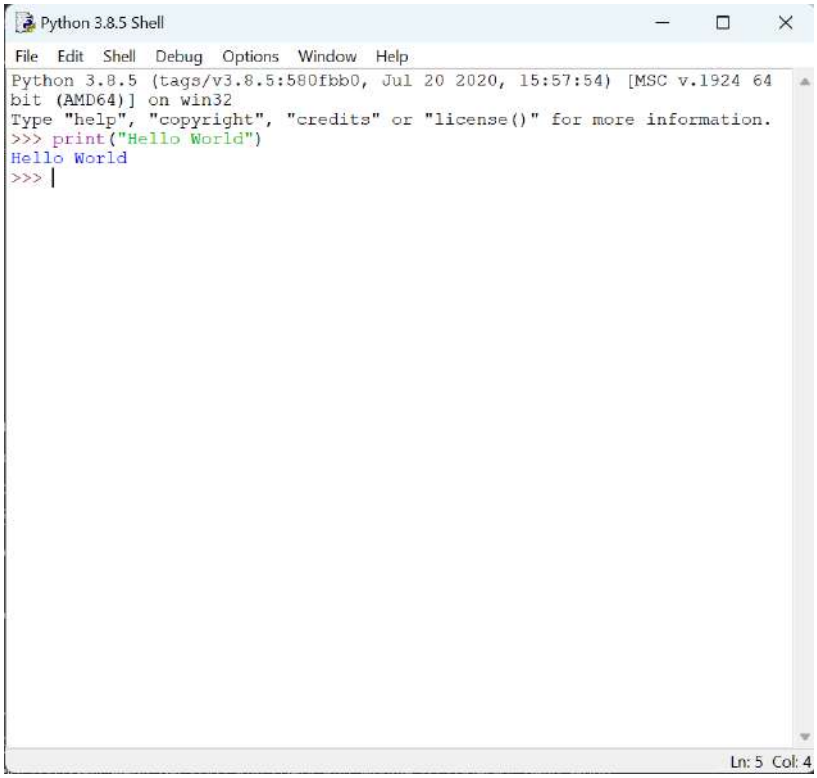


ပုံ (၁.၁) IDLE Interactive Shell

ဒီမျှားလေးနှစ်ခု >> ရဲ့ နောက်မှာ

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

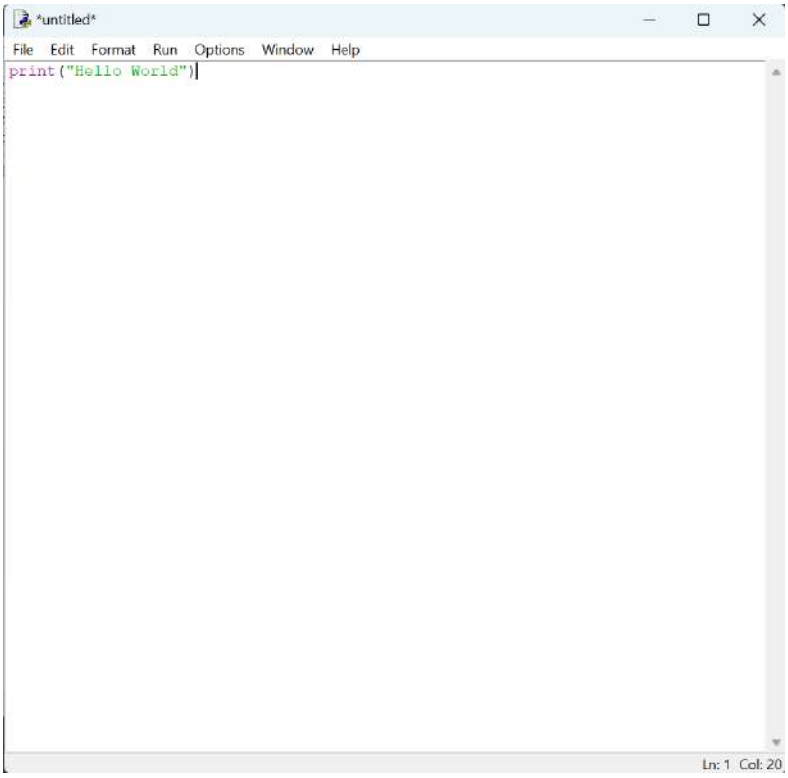
ကို ရိုက်ပြီး Enter နှိပ်လိုက်ပါ။ အခုလို ရလဒ်ကို မြင်တွေ့ရမှာဖြစ်ပါတယ်။



ပုံ (၁.၂) Hello World ပရိုဂရမ်ကို IDLE မှာ မြင်တွေ့ရခြင်း

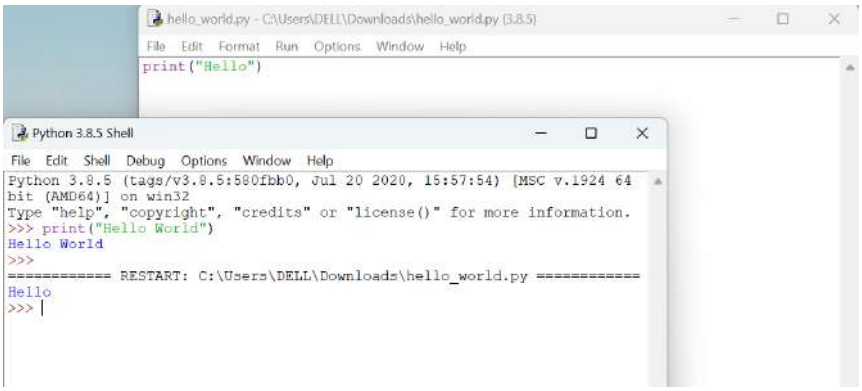
Interactive Shell တွေထဲမှာ ရှုပ်ထွေးတဲ့ကုဒ်တွေ၊ ပရိုဂရမ်တွေကို ရေးသားလို့ အဆင်မပြေပါဘူး။ ကုဒ်တစ်ကြောင်းစီတိုင်းအတွက် ရလဒ်တစ်ကြောင်းစီကို ထုတ်ပေးနေတဲ့အတွက် ရိုးရှင်းတဲ့ကုဒ်တွေ၊ စမ်းသပ်ကုဒ်တွေ အတွက်သာ ရေးသားဖို့ အဆင်ပြေပါတယ်။ ဒါကြောင့် IDLE ရဲ့ အပေါ်ဖက် ဘယ်ဘက်ဒေါင့် Menu Bar လေးထဲက File ကိုနှိပ်ပါ၊ အဲနောက်မှာ New File ကို နှိပ်လိုက်ပါ။ Text Editor တစ်ခုပေါ်လာမှာဖြစ်ပါတယ်။ ကုဒ်တွေကို Text Editor ထဲမှာ ရေးသားလို့ ရနိုင်မှာဖြစ်ပါတယ်။

Hello World ပရိုဂရမ်လေးကို Text Editor ထဲမှာ ရေးသားပါမယ်။ ပုံ (၁.၃) ကို ကြည့်ပါ။



ပုံ (၁.၃) Hello World ပရိုဂရမ်ကို IDLE Text Editor မှာ ရေးသားခြင်း

အဆိုပါ ပရိုဂရမ်ကို သိမ်းဆည်းဖို့အတွက် File ကိုနှိပ်ပါ၊ Save As ကိုထပ်နှိပ်ပါမယ်။ ဖိုင်နာမည်ကို “hello_world.py” လို့ပေးပြီး နှစ်သက်ရာ ဖိုဒါထဲမှာ သိမ်းဆည်းလိုက်ပါ။ (ဖိုင်နာမည်ကို “hello_world” အစား မိမိ နှစ်သက်ရာ အမည်ကို ပေးလို့ရပါတယ်။ Python ဖိုင်တွေကို .py Extension နဲ့ ဖန်တီးရပါတယ်။) ဖိုင်ကို သိမ်းပြီးသွားတဲ့အခါ Menu Bar ထဲက Run ကိုနှိပ်၊ Run Module ကို ထပ်နှိပ်ပြီး Hello World ပရိုဂရမ်လေးကို Run လိုက်ပါမယ်။ အဲဒီအခါ Interactive Shell ထဲမှာ “Hello World” ဆိုတဲ့ရလဒ်ကို အခုလို မြင်တွေ့ရမှာပဲဖြစ်ပါတယ်။



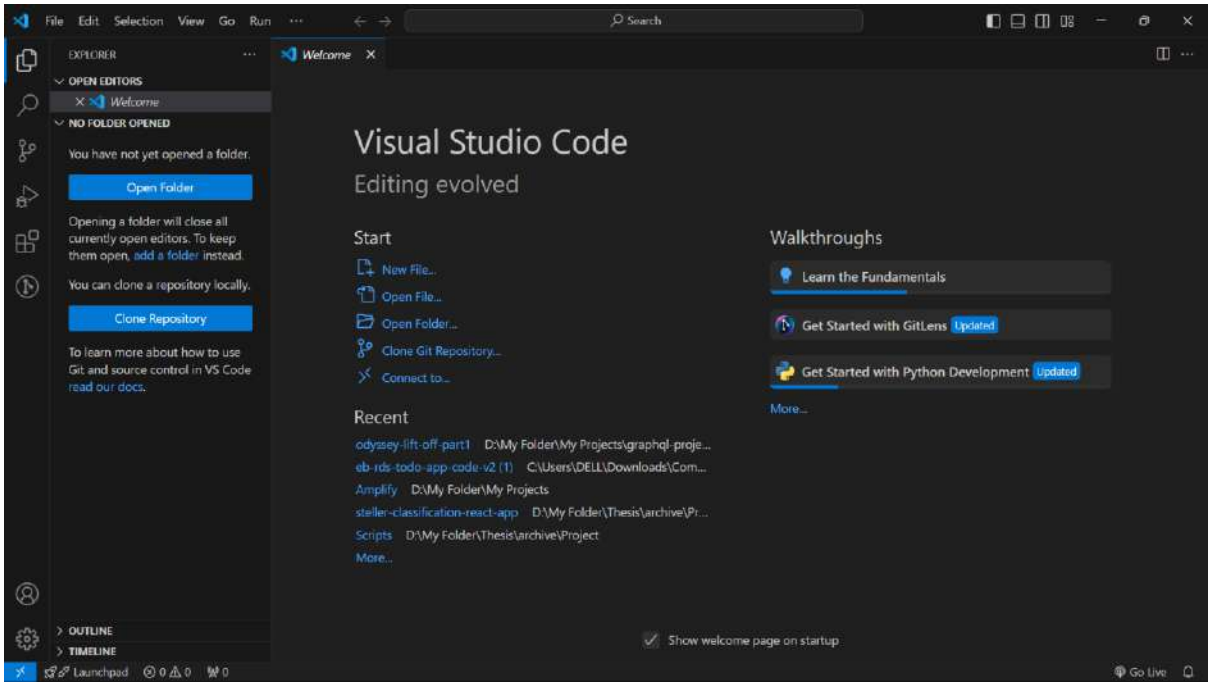
ပုံ (၁.၄) Hello World ပရိုဂရမ်ကို Text Editor နဲ့ Interactive Shell ထဲမှာ တွေ့မြင်ရခြင်း

VSCode ထည့်သွင်းခြင်း

ကုဒ်တွေကို ရေးသားနိုင်ဖို့အတွက် Text Editor တွေ လိုအပ်ပါတယ်။ Python ရဲ့ IDLE ကို အသုံးပြုလို့ရနေပေမယ့် ပိုပြီး အဆင်ပြေ၊ လွယ်ကူစေမယ့် Text Editor IDE (Integrated Development Enviroment) တွေ အများအပြား ရှိပါတယ်။ လက်ရှိ Community မှာ အသုံးအများဆုံးက VS Code နဲ့ PyCharm ဖြစ်ပါတယ်။ စာအုပ်ထဲမှာ VS Code ကို အသုံးပြုပြီး ရေးသားသွားမှာ ဖြစ်ပါတယ်။

VS Code ကို Microsoft က ဖန်တီးထားတာဖြစ်ပြီး Python အပြင် တခြားသော Programming Language ပေါင်းများစွာကို ရေးသားနိုင်ဖို့ Support လုပ်ပေးထားပါတယ်။ VS Code ကို ထည့်သွင်းဖို့အတွက် https://code.visualstudio.com/ မှာ ဝင်ရောက်ပြီး သက်ဆိုင်ရာ Operating System အတွက် Installer ကို ရယူနိုင်ပါတယ်။ ထည့်သွင်းနည်း အပြည့်အစုံကို ဒီဗီဒီယိုမှာ ဝင်ရောက်ကြည့်ရှုနိုင်ပါတယ်။

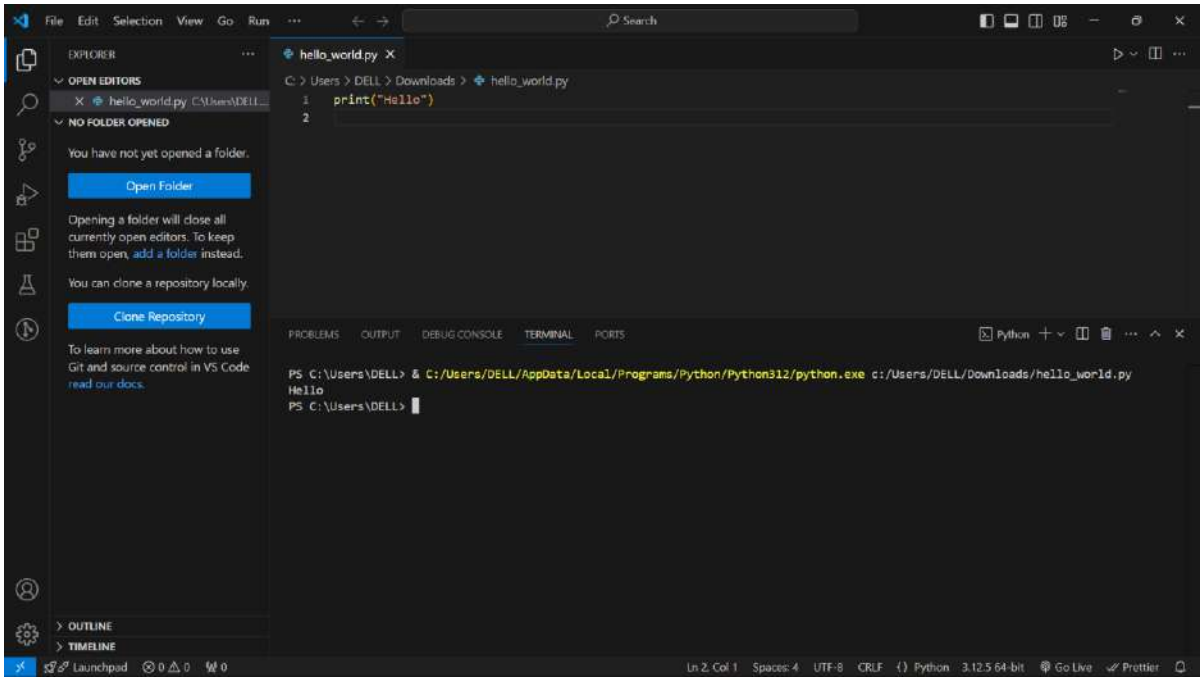
VS Code ကို ဖွင့်လိုက်တဲ့အခါ အခုလိုမျိုး မြင်တွေ့ရမှာပါ။



ပုံ (၁.၅) VS Code ကို စဖွင့်ခြင်း မြင်တွေ့ရပုံ

ရှေ့မှာ ရေးသားခဲ့တဲ့ Hello World ပရိုဂရမ်ကို VS Code မှာ Run ကြည့်ပါမယ်။ ပထမဆုံး VS Code ရဲ့ အပေါ်ဆုံး ဒေါင့်က File ကို နှိပ်ပါမယ်၊ Open File ကို ထပ်နှိပ်ပြီး “hello_python.py” ကို

ရွေးလိုက်ပါ။ ပရိုဂရမ်ကို Run ဖို့အတွက် VS Code ရဲ့ ညာဖက် အပေါ်ဆုံးဒေါင့်က ▶ Button လေးကို နှိပ်လိုက်ပါမယ်။ ရလာဒ်ကို အခုလိုမျိုး မြင်တွေ့ရမှာဖြစ်ပါတယ်။



ပုံ (၁.၆) Hello World ပရိုဂရမ်ကို VS Code မှာ Run လိုက်တဲ့အခါ မြင်တွေ့ရပုံ

ကုဒ်တွေကို စတင်ရေးသားဖို့အတွက် File -> New File -> Python File ကို ရွေးချယ်ပြီး Python ကုဒ်တွေကို ရေးသားလို့ရပါတယ်။ File -> Save As ကို နှိပ်ပြီး အဆိုပါ ဖိုင်ကို နှစ်သက်ရာ ဖိုဒါမှာ သိမ်းဆည်းလို့ ရနိုင်ပါတယ်။

နောက်လာမယ့်အခန်းတွေမှာ Python ကိုသုံးပြီး နမူနာကုဒ်တွေကို ရေးသားသွားမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် အခုဖော်ပြခဲ့တဲ့ နည်းလမ်းတွေကို အသုံးပြုပြီး နမူနာ Python ကုဒ်တွေကို ကိုယ်တိုင် Run ကြည့်နိုင်သလို၊ ကုဒ်အသစ်တွေ ရေးသားဖန်တီးနိုင်မယ်လို့ မျှော်လင့်ပါတယ်။

ဝေါဟာရများ

Code	ကွန်ပျူတာကို ခိုင်းစေနိုင်သည့် ညွှန်ကြားချက်များ။
Programming	ကွန်ပျူတာကို ခိုင်းစေနိုင်ရန် ညွှန်ကြားချက်များ ရေးသားသည့် လုပ်ငန်းစဉ်။
Programming Language	ကွန်ပျူတာကို ခိုင်းစေနိုင်ရန် ညွှန်ကြားချက်များ ရေးသားနိုင်သည့် ဘာသာစကား။
Low-level Language	ကွန်ပျူတာ၏ စက်ပိုင်းဆိုင်ရာစနစ်တစ်ခုလုံးအထိ တိုက်ရိုက် ညွှန်ကြားချက်များ ရေးသားနိုင်သည့် ဘာသာစကား။
High-level Language	အလွယ်တကူ ဖတ်ရှုနားလည်နိုင်ပြီး ညွှန်ကြားချက်များ ရေးသားနိုင်သည့် ဘာသာစကား။
Python	၁၉၉၁ ခုနှစ်တွင် Dutch ပရိုဂရမ်မာ Guido van Rossum ဆိုသူက ဖန်တီးခဲ့သော High-level Programming Language တစ်မျိုး။
Operating System	ကွန်ပျူတာ၏ စက်ပိုင်းဆိုင်ရာစနစ်တစ်ခုလုံးကို ထိန်းချုပ် လုပ်ကိုင်ပေးသည့် အထူးဆော့ဖ်ဝဲလ် တစ်မျိုး။
Ubuntu	Linux ကို အခြေခံထားသည့် Operating System အမျိုးအစားတစ်မျိုး။
IDLE	Python နှင့်အတူ တစ်ပါတည်း တွဲဖက်ပါဝင်သည့် IDE ဆော့ဖ်ဝဲလ် Tool တစ်ခု။
Interactive Shell	ကုဒ်တွေကို တိုက်ရိုက် ရေးချခိုင်းစေနိုင်သည့် Command Line Interface တစ်မျိုး။

IDE	(Integrated Development Enviroment) ဆော့ဖ်ဝဲလ် တစ်ခု ဖန်တီးဖို့အတွက် လိုအပ်သည့် Tool များကို တခါတည်း ထည့်သွင်း တည်ဆောက်ပေးထားသည့် ဆော့ဖ်ဝဲလ် အမျိုးအစား တစ်မျိုး။
VS Code, PyCharm	ဆော့ဖ်ဝဲလ် ရေးသားဖန်တီးရန် အသုံးပြုနိုင်သည့် IDE ဆော့ဖ်ဝဲလ်များ။

အခန်း (၂)

Programming စတင်ရေးသားခြင်း

ရှေ့ကအခန်းမှာ “Hello World” ဆိုတဲ့ အခြေခံပရိုဂရမ်လေးတစ်ခုကို ရေးသားခဲ့ပါတယ်။ ဒီအခန်းမှာတော့ Programming နဲ့ ပတ်သတ်တဲ့ အခြေခံသဘောတရားတွေကို နမူနာကုဒ်တွေသုံးပြီး လေ့လာသွားကြပါမယ်။

ပထမဆုံးအနေနဲ့ “Hello World” ပရိုဂရမ်ကို အနည်းငယ်ပြောင်းလဲပါမယ်။ “Hello World” ဆိုတဲ့စာကြောင်းလေးအစား “Hola” လို့ပြောင်းလဲပြီး ရေးသားပါမယ်။

hola_program.py

```
print("Hola!")
>> Hola!
```

ဒီပရိုဂရမ်ကို ရလာဒ် အကြိမ် (၁၀၀) ထုတ်ပေးနိုင်တဲ့ ပရိုဂရမ်တစ်ခုအဖြစ် ပြင်ဆင် ရေးသားပါမယ်။ အောက်မှာ ဖော်ပြထားတဲ့ကုဒ်လေးကို ကူးရေးပြီး Run ကြည့်ပါ။

hola_100_time.py

```
for i in range(100):
    print("Hola!")
```

“Hola!” ဆိုတဲ့စာကြောင်းလေးကို အကြိမ် ၁၀၀ ထုတ်ပေးသွားတာကို တွေ့ရမှာဖြစ်ပါတယ်။ အကြိမ် ၁၀၀ ထုတ်ခိုင်းဖို့အတွက် ကုဒ်ကြောင်းရေ အခု ၁၀၀ ရေးသားဖို့ မလိုအပ်ဘဲ၊ သက်ဆိုင်ရာကုဒ်တွေကို ရေးသားပေးယုံနဲ့ ရလာဒ်ကို လွယ်ကူလျှင်မြန်စွာ ရရှိနိုင်မှာ ဖြစ်ပါတယ်။ ဒါဟာ Programming တွေရဲ့ အာသာချက်ပဲဖြစ်ပါတယ်။

> နမူနာကုဒ်တွေမှာ ဖော်ပြထားတဲ့ ဒီမျှားသင်္ကေတ >> နောက်က စာတွေဟာ ပရိုဂရမ်က ထုတ်ပေးလိုက်တဲ့ ရလာဒ် (Ouput) တွေကို ဖော်ပြထားတာဖြစ်ပါတယ်။ စက်ထဲမှာ စမ်းသပ်တဲ့အခါ ထည့်သွင်းရေးသားဖို့ မလိုအပ်ပါဘူး။ နမူနာကုဒ်တွေအားလုံးကို တခြားသောစာတွေနဲ့ ကွဲပြားခြားနားစေဖို့အတွက်လဲ အခန်းတိုင်းမှာ Courier New ဖောင့်ကို အသုံးပြုပြီး ရေးသားပေးထားပါတယ်။

Comments - ကွန်မန့်များ

Comment တွေဆိုတာ ကုဒ်တွေကို ရှင်းလင်းထားတဲ့ ဖော်ပြချက်တွေဖြစ်ပါတယ်။ ကုဒ်ကို ဖန်တီးထားတဲ့ မူရင်းပရိုဂရမ်မာက တခြားသောပရိုဂရမ်မာတွေကို နားလည်နိုင်စေဖို့အတွက် အများနားလည်တဲ့ဘာသာစကားနဲ့ ရေးသားထားတဲ့ ရှင်းလင်းချက်တွေပါ။။ Comment တွေဟာ ပုံမှန်အားဖြင့် တစ်ကြောင်း သို့မဟုတ် စာပိုဒ်တစ်ခုအနေနဲ့ ရှိတတ်ပါတယ်။ အင်္ဂလိပ်ဘာသာကို အများဆုံး အသုံးပြုပြီး ရေးသားလေ့ရှိသလို တချို့သောအခြေအနေတွေမှာ ပရိုဂရမ်ပေါ် မူတည်ပြီး တခြားသော ဘာသာစကားတွေကိုလည်း အသုံးပြုလေ့ရှိပါတယ်။ Comment တွေကို တခြားကုဒ်တွေလိုမျိုး ပရိုဂရမ်က ထည့်သွင်း တွက်ချက်သွားမှာ မဟုတ်ပါဘူး။ Python မှာ Comment ကို Hash `#` သင်္ကေတအသုံးပြုပြီး ရေးသားရပါတယ်။

နမူနာ Comment ရေးသားပုံနည်းလမ်းတွေကို အောက်မှာ ဖော်ပြပေးထားပါတယ်။

single_line_comment.py

```
# This is a comment
print("The comment does not affect this code")

print("The comment does not affect this code")  # This is a comment
```

သတိထားရမှာက Comment တွေကို ရှုပ်ထွေးတဲ့ ကုဒ်တွေအတွက်သာ ရေးသားဖော်ပြသင့်ပါတယ်။ ရိုးရှင်းပြီး နားလည်လွယ်ကူတဲ့ ကုဒ်တွေအတွက် Comment တွေ ရေးသားဖော်ပြဖို့ မလိုအပ်ပါဘူး။ ကုဒ်တစ်ကြောင်းစီတိုင်းအတွက်လည်း Comment တွေ ရေးသားဖော်ပြဖို့ မလိုအပ်ပါဘူး။

အောက်မှာ ဖော်ပြထားတဲ့ကုဒ်က Comment ကို မလိုအပ်ဘဲ ရေးသားဖော်ပြထားတဲ့ နမူနာဖြစ်ပါတယ်။

```
print("Hello, World!")  # this line of code prints Hello, World
```

တကယ်တမ်း Comment ရေးသားဖော်ပြသင့်တဲ့ အခြေအနေကို အောက်ကကုဒ်မှာ နမူနာ ဖော်ပြထားပါတယ်။

complex_comment_example.py

```
y_vertex = a * x_vertex**2 + b * x_vertex + c
# the y-vertex of a parabola defined by the equation y = ax^2 + bx + c.
```

လောလောဆယ် အပေါ်ကဖော်ပြထားတဲ့ ကုဒ်ကို နားလည်ဖို့ မလိုအပ်သေးပါဘူး။ နမူနာကုဒ်ဟာ Parabola တစ်ခုရဲ့ Y-Vertex ကို တွက်ချက်တဲ့ ဖော်မြူလာကုဒ်တစ်ခုပါ။ ဒီလိုမျိုး ရှုပ်ထွေးတဲ့ အခြေအနေမှာသာ သက်ဆိုင်ရာကုဒ်ကို Comment ရေးသားပြီး ရှင်းလင်းဖော်ပြပေးဖို့ လိုအပ်တာဖြစ်ပါတယ်။

Lines - လိုင်းများ

ကုဒ်တွေကို သက်ဆိုင်ရာ Line နံပါတ်တွေ အသုံးပြုပြီး ဖော်ပြလေ့ရှိပါတယ်။ အောက်ကနမူနာ ကုဒ်မှာဆိုရင် ကုဒ် Line သုံးကြောင်းပါဝင်ပါတယ်။

```
#line 1
#line 2
#line 3
```

ပုံမှန်အားဖြင့် ကုဒ်တစ်ကြောင်းကို Line တစ်ကြောင်းအဖြစ် ဖော်ပြလေ့ရှိပါတယ်။ အကယ်၍ ကုဒ်က လိုင်းတစ်ကြောင်းထက် ပိုပြီး ရေးသားဖို့လိုအပ်လာတဲ့အခါ နောက်ထပ်တစ်ကြောင်းကို ထပ်ဆင်းပြီး ရေးသားလို့ရပါတယ်။ Python မှာ Line တွေက အရေးပါပါတယ်။ အောက်က နမူနာမှာ လိုင်းတစ်ကြောင်းထက်ပိုပြီး ရေးသားထားတဲ့ ကုဒ်ကို ဖော်ပြထားပါတယ်။

long_line_example.py

```
print("""This is a really really really really really really
         really really long line of code.""")
```

Keywords

Programming Language တိုင်းမှာ အထူးသီးသန့် အဓိပ္ပါယ်ရှိတဲ့ စကားလုံးတွေ အများအပြား ပါရှိပါတယ်။ Keyword တွေလို့ ခေါ်ပါတယ်။ ဥပမာ၊ `for` ဆိုတဲ့ Keyword က ကုဒ်ကို အကြိမ်ပေါင်းများစွာ ထပ်တလဲလဲ ဆောင်ရွက်စေနိုင်ကြောင်း ရှေ့မှာ ဖော်ပြခဲ့ပြီးပါပြီ။ အခြားသော Keyword တွေကိုလည်း သက်ဆိုင်ရာ အခန်းတိုင်းမှာ လိုအပ်သလို လေ့လာဖော်ပြသွားမှာ ဖြစ်ပါတယ်။

Spacing

နမူနာ အကြိမ် ၁၀၀ ပရင့်ထုတ်ပေးနိုင်တဲ့ ပရိုဂရမ်လေးကို သေသေချာချာ လေ့လာကြည့်ပါ။

indentation_example.py

```
for i in range(100):
    print("Hello World")
```

`print` ဆိုတဲ့စာကြောင်းဟာ Space လေးခုစာ အကွာအဝေးကနေ စတင်ရေးသားထားတာကို တွေ့ရမှာဖြစ်ပါတယ်။ ဒါဟာ Indentation ကို အသုံးပြုပြီး ရေးသားထားတာဖြစ်ပါတယ်။ Tab Key ကိုအသုံးပြုပြီး Indentation တစ်ခု ဖန်တီးနိုင်ပါတယ်။

Indentation တစ်ခုတိုင်းမှာ Space လေးခုစာ အကွာအဝေး ပါရှိပါတယ်။ Python မှာ တိကျမှန်ကန်တဲ့ Indentation (Space အကွာအဝေး) ကို အသုံးမပြုရင် ပရိုဂရမ်ဟာ အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ ဘာကြောင့် Indentation ကို အသုံးပြုရသလဲဆိုရင် Code Block တစ်ခုရဲ့ အစနဲ့ အဆုံးကို ဖော်ပြပေးဖို့ အသုံးပြုရတာပါ။ ဒီအကြောင်းကို နောက်ပိုင်းမှာ ပိုမိုရှင်းပြသွားပါမယ်။ တခြားသော Programming Language တွေမှာတော့ Python လိုမျိုး Indentation တွေအစား Curly Bracket လို့ခေါ်တဲ့ တွန့်ကွင်းတွေ `{}` ကိုသာ အသုံးပြုပြီး Code Block တွေကို ဆုံးဖြတ်ကြပါတယ်။

Indentation တွေဟာ Python ကုဒ်တွေကို ပိုမိုလွယ်ကူစွာ ဖတ်ရှုနိုင်စေတဲ့ အဓိက အချက်တစ်ခုလည်းဖြစ်ပါတယ်။

Data Type များ

ဒေတာတွေကို သွင်ပြင်လက္ခဏာပေါ်မူတည်ပြီး အမျိုးအစား အများအပြား ခွဲခြားနိုင်ပါတယ်။ အဲဒီလို မတူညီတဲ့ ဒေတာအမျိုးအစားတွေကို Data Type တွေလို့ ခေါ်ပါတယ်။

Python မှာဆိုရင် `123` လိုမျိုး၊ ဒါမှမဟုတ် `"Hello World"` လိုမျိုး စသည်ဖြင့် ရှိရှိသမျှ ဒေတာတန်ဖိုးတိုင်းကို Object အဖြစ်သတ်မှတ်ပါတယ်။ Object တွေအကြောင်းကို နောက်ပိုင်းမှာ အသေးစိတ်ဆက်ပြီး လေ့လာသွားပါမယ်။

လောလောဆယ် Python ရဲ့ Object ကို Identity, Data Type နဲ့ Value ဆိုတဲ့ Property ၃ ခု ပါဝင်တဲ့ အထူးသီးသန့် Data အမျိုးအစားတစ်ခုလို့ အကြမ်းဖျင်းမှတ်သားထားနိုင်ပါတယ်။

- Identity ဆိုတာက Memory ပေါ်မှာ သိမ်းဆည်းထားတဲ့နေရာပါ။

- Data Type က အဲဒီ Object လက်ခံထားတဲ့ ဒေတာအမျိုးအစားဖြစ်ပါတယ်။ မတူညီတဲ့ ဒေတာအမျိုးအစားတွေပေါ်မူတည်ပြီး Object ရဲ့ ဂုဏ်သတ္တိ (Property) တွေလည်း ကွဲပြားသွားရပါတယ်။
- Value ကတော့ အဲဒီ Object ရဲ့ တကယ်တမ်း ပိုင်ဆိုင်ထားတဲ့ ဒေတာတန်ဖိုးဖြစ်ပါတယ်။ ဥပမာ၊ 2 ဆိုတဲ့ ဒေတာကို လက်ခံထားတဲ့ Object တစ်ခုရဲ့ Value တန်ဖိုးက 2 ဖြစ်ပါမယ်။

ပြောရရင် “Hello World” လို့မြင်ရတဲ့ စာသားဟာ Python မှာဆိုရင် String လို့ ခေါ်တဲ့ Data Type ပါဝင်တဲ့ Object တစ်မျိုးပါ။ `Hello World` ဆိုတာက ဒီ String ရဲ့ Value တန်ဖိုးပါ။ Python မှာ String ကို `str` နဲ့ ရည်ညွှန်းပါတယ်။ String တွေကို စရေးမယ်ဆိုရင် စာသားရဲ့အစနဲ့ အဆုံးမှာ တူညီတဲ့ Single Quote `'` ဒါမှမဟုတ် Double Quote `"` တစ်ခုခုကို သုံးပြီး ရေးရပါတယ်။ String တစ်ခုမှာ Character တစ်လုံးနဲ့အထက် ပါဝင်နိုင်ပါတယ်။ Character ဆိုတာ Unicode Character Table မှာ ပါဝင်တဲ့ သင်္ကေတတစ်ခုခုကို ဆိုလိုတာပါ။ Unicode Character တွေအကြောင်း ပိုသိချင်ရင် http://unicode-table.com/en မှာ အပြည့်အစုံ ဝင်ရောက် ကြည့်ရှုနိုင်ပါတယ်။ ရှင်းရှင်းပြောရရင် String တွေက စာသား (Text) တွေကို ဖော်ပြဖို့အတွက် ဖြစ်ပါတယ်။

string_data_type_example.py

```
print("Hello World")
>> Hello World
```

စာသားတွေလိုပါပဲ၊ ကိန်းဂဏန်း (Number) တွေကလည်း Object တွေပါပဲ။ 2, 3, 4, 10 စတဲ့ ကိန်းပြည့်တွေရဲ့ Data Type က Integer ပါ။ Integer ကို Python မှာ `int` နဲ့ ရည်ညွှန်းပါတယ်။ String တွေနဲ့မတူတာက Integer တွေမှာ ပေါင်းခြင်း၊ နုတ်ခြင်း စတဲ့ ဂုဏ်သတ္တိ (Property) တွေ ရှိနေတာပါ။ ဒါကြောင့်ပဲ ကိန်းဂဏန်းဆိုင်ရာ တွက်ချက်မှုတွေ ပြုလုပ်လို့ရတာဖြစ်ပါတယ်။

ကိန်းပြည့်တွေကို Integer လို့ခေါ်ပေမယ့် ဒဿမကိန်းတွေကိုတော့ Float လို့ ခေါ်ပါတယ်။ 3.2, 15.8, 3.14 စတဲ့ ဒဿမကိန်းတွေအားလုံးရဲ့ Data Type က `float` ပါ။ Float တွေမှာလည်း သီးခြားဂုဏ်သတ္တိ (Property) တွေ ရှိပါတယ်၊ ယေဘုယျအားဖြင့် Integer တွေနဲ့ ဆင်တူပါတယ်။

integer_and_float.py

```
print(2.2 + 2.2)
>> 4.4
```

နောက်ထပ် Data Type တစ်မျိုးက Boolean ပါ။ သူ့မှာ True နဲ့ False ဆိုတဲ့ တန်ဖိုးနှစ်ခုသာရှိပါတယ်။ Python မှာ `bool` နဲ့ ဖော်ပြပါတယ်။

boolean_and_none_example.py

```
print(True)
>> True
print(False)
>> False
```

Python မှာ ထူးခြားတဲ့ Data Type တစ်မျိုးရှိပါတယ်။ သူကတော့ `None` ပါ။ သူ့မှာ တိကျတဲ့ တန်ဖိုးတစ်ခုမှ မပါဝင်ပါဘူး။

boolean_and_none_example.py

```
print(None)
>> None
```

Constant နှင့် Variable များ

Python ကို သုံးပြီး ပေါင်းခြင်း၊ နုတ်ခြင်း၊ စားခြင်း၊ မြှောက်ခြင်း၊ ထပ်ကိန်း စသည်ဖြင့် သခ်ၤာတွက်ချက်မှုအားလုံးကို လုပ်ဆောင်နိုင်ပါတယ်။

constant_and_variable_arithmetic_operations.py

```
print(2 + 2)
>> 4
```

```
print(4 - 2)
>> 2
```

```
print(4 * 4)
>> 16
```

```
print(4 / 2)
>> 2
```

```
print(4 ** 2)
>> 16
```

ဘယ်တော့မှ မပြောင်းလဲဘဲ ရှိနေမယ့် တန်ဖိုးတွေကို ကိန်းသေတန်ဖိုး (Constant) တွေ လို့ခေါ်ပါတယ်။ အပေါ်က နမူနာမှာ ဖော်ပြခဲ့တဲ့ ကိန်းဂဏန်းတွေက Constant တွေဖြစ်ပါတယ်။ ဥပမာအားဖြင့် 2 ဆိုတဲ့ ဂဏန်းဟာ ဘယ်နေရာမှာဖြစ်ဖြစ် အမြဲတမ်း 2 ရဲ့ တန်ဖိုးကိုပဲ ကိုယ်စားပြုနေရမှာပါ။

တချိန်ချိန်မှာ ပြောင်းလဲသွားနိုင်မယ့် တန်ဖိုးတွေရှိနေရင် Variable လို့ခေါ်ပါတယ်။ Variable တိုင်းမှာ ကိုယ်ပိုင်အမည်တစ်ခုရယ်၊ သူနဲ့ သက်ဆိုင်တဲ့ တန်ဖိုးတစ်ခုရယ် ရှိရပါတယ်။ Variable ရဲ့ အမည်ကို Character တစ်လုံး ဒါမှမဟုတ် တစ်လုံးထက်ပိုပြီး သတ်မှတ်နိုင်ပါတယ်။ Variable တစ်ခု ဖန်တီးဖို့အတွက် Assignment Operator လို့ခေါ်တဲ့ `=` သင်္ကေတလေးကို အသုံးပြုရပါတယ်။ နမူနာကို အောက်မှာဖော်ပြထားပါတယ်။

variable_declaration_and_reassignment.py

```
x = 20
print(x)
>> 20
```

အဆိုပါ Variable `x` လေးရဲ့တန်ဖိုးကို အခုလို ပြောင်းလဲနိုင်ပါတယ်။

variable_declaration_and_reassignment.py

```
x = 20
print(x)
>> 20

x = 200
print(x)
>> 200
```

သင်္ချာတွက်ချက်မှုတွေအတွက်လည်း Variable တွေကို လိုအပ်သလို အသုံးပြုနိုင်ပါတယ်။

variables_in_arithmetic_expression.py

```
x = 20
y = 10
z = x + y
print(z)
>> 30

a = x * y
print(a)
>> 200
```

ပရိုဂရမ်ရေးသားတဲ့အခါ Variable တန်ဖိုးတွေကို တိုးတာ (Increment)၊ လျော့တာ (Decrement) စတဲ့ လုပ်ဆောင်ချက်တွေကို အမြဲလိုလို လုပ်ဆောင်ရလေ့ရှိပါတယ်။ Python မှာတော့ ဒီလိုလုပ်ဖို့အတွက် လွယ်ကူတဲ့နည်းလမ်း ရှိပါတယ်။ Variable တစ်ခုရဲ့ တန်ဖိုးကို တိုးမယ်ဆိုရင် ညီမျှခြင်းလက္ခဏာ `=` ရဲ့ ညာဘက်မှာ လက်ရှိ Variable ရဲ့ တန်ဖိုးကို တိုးချင်တဲ့ တန်ဖိုးနဲ့ ပေါင်းပြီး၊ တန်ဖိုးအသစ် ပြန်သတ်မှတ်ပေးရုံပါပဲ။ လျော့ချင်ရင်လည်း အလားတူပဲ နုတ်ပေးရမှာပါ။ နမူနာကို အောက်မှာ ဖော်ပြထားပါတယ်။ သေသေချာချာ လေ့လာကြည့်ပါ။

variable_increment_and_decrement.py

```
x = 20
x = x + 40
print(x)
>> 60
```

Variable တန်ဖိုးကို လျော့ချဖို့အတွက်ဆိုရင်လည်း အလားတူစွာ ပြန်နုတ်ပေးရုံပါပဲ။

variable_increment_and_decrement.py

```
x = 20
x = x - 5
print(x)
>> 15
```

Variable တန်ဖိုးကို တိုးတာ၊ လျော့တာတွေအတွက် ပိုပြီး လွယ်ကူမြန်ဆန်တဲ့ ရေးထုံးနောက်တစ်ခုရှိပါသေးတယ်။ အခုလို ရေးသားတာပါ။ ရလာဒ်က အတူတူပါပဲ။

variable_increment_and_decrement.py

```
x = 20
x += 40
print(x)
>> 60
```

```
x = 20
x -= 5
print(x)
>> 15
```

Variable တွေကို ကိန်းဂဏန်းတွေအတွက်တင် မဟုတ်ပါဘူး၊ တခြားသော Data Type တွေအတွက်လည်း အသုံးပြုလို့ရပါတယ်။ အောက်မှာ ဖော်ပြထားပါတယ်။

variables_with_other_data_types.py

```
my_string = “hello world”
this_is_float = 4.10
is_male = True
```

Python မှာ Variable တွေရဲ့ အမည်တွေကို သတ်မှတ်တဲ့အခါ ဒီစည်မျဉ်းတွေကို မဖြစ်မနေလိုက်နာရပါမယ်။

- Variable အမည်တွေမှာ Space လုံးဝ မပါရပါ။ စကားလုံးနှစ်လုံးကြားမှာ Underscore ကို သုံးနိုင်ပါတယ်။ ဥပမာ၊ `my_name = “Code with Thura”`
- Variable အမည်တွေမှာ Letter, Number နဲ့ Underscore တွေသာ အသုံးပြုရပါမယ်။
- Variable အမည်တွေရဲ့ ပထမစာလုံးကို Number နဲ့ စ, မကြေညာရပါ။

- Python က ကြိုတင်သတ်မှတ်ထားပြီးသား Keyword အမည်တွေနဲ့ တူညီနေတဲ့ Variable အမည်တွေကို မပေးရပါ။ Keyword အမည်စာရင်း အပြည့်အစုံကို ဒီလင့်ခ်မှာ ဝင်ကြည့်နိုင် ပါတယ်။

Syntax များ

Syntax ဆိုတာ ဘာသာစကားတစ်ခုရဲ့ ဝါကျတည်ဆောက်ပုံကို ထိန်းချုပ်ပြဋ္ဌာန်းထားတဲ့ စည်းမျဉ်းစည်းကမ်းတွေ၊ အခြေခံမူတွေနဲ့ လုပ်ငန်းစဉ်တွေဖြစ်ပါတယ်။ တိတိကျကျပြောရရင် စကားလုံး အစီအစဉ် ဖွဲ့စည်းပုံဖြစ်ပါတယ်။ အင်္ဂလိပ်ဘာသာစကားမှာ Syntax တွေရှိသလို၊ မြန်မာဘာသာစကားမှာလည်း Syntax တွေရှိပါတယ်။ အလားတူစွာပဲ၊ Programming တိုင်းမှာလည်း ရှိပါတယ်။ Python မှာလည်း ရှိပါတယ်။ ဥပမာအားဖြင့် အင်္ဂလိပ်စာမှာ ဝါကျတစ်ခုကို အစက် (.) နဲ့ အဆုံးသတ်ရသလို၊ မြန်မာစာမှာလည်း ဝါကျတစ်ခုကို ပုဒ်မ (။) နဲ့ အဆုံးသတ်ရပါတယ်။ အဲဒီလိုပါပဲ Python မှာဆိုရင်လည်း စနစ်တကျ လိုက်နာရမယ့် စည်းမျဉ်းသတ်မှတ်ချက်တွေ ရှိပါတယ်။

နမူနာတစ်ခုကို ပြောရရင်၊ Python မှာ String တွေကို အမြဲတမ်း Quotation Marks (အဖွင့်အပိတ်ကွင်း) တွေနဲ့ ရေးသားရပါတယ်။

```
print("Hello World!")
```

ဒီကုဒ်ဟာ Python ရဲ့ မှန်ကန်တဲ့ ရေးထုံးကို အသုံးပြုထားတာပါ။ အကယ်၍ ကျွန်တော်တို့ဟာ Quotation Mark ကို အဖွင့်မှာပဲ သုံးခဲ့ပြီး၊ အပိတ်အတွက်မသုံးခဲ့ဘူးဆိုရင် Python ရဲ့ Syntax ကို ချိုးဖောက်ရာရောက်ပြီး ကျွန်တော်တို့ရဲ့ Code ဟာ အလုပ်လုပ်မှာ မဟုတ်တော့ပါဘူး။ Python ကို အသုံးပြုတဲ့အခါ သူ့ရဲ့ သတ်မှတ်ထားတဲ့ Syntax တွေကို မချိုးဖောက်ရပါဘူး။ အဲဒီလိုမျိုး ချိုးဖောက်မိတဲ့အခါ ဘာတွေဖြစ်လာမလဲဆိုတာ ဆက်လေ့လာသွားပါမယ်။

Comparison Operator များ

နောက်ထပ် Operator အမျိုးအစားကတော့ Comparison Operator တွေပါ။ Comparison Operator တွေကို Arithmetic Operator တွေလိုပဲ Expression တွေမှာ အသုံးပြုရပါတယ်။ ဒါပေမယ့် မတူညီတဲ့အချက်က Comparison Operator ကို အသုံးပြုထားတဲ့ Expression တွေဟာ `True` သို့မဟုတ် `False` ဆိုတဲ့ တန်ဖိုးနှစ်ခုကိုသာ တွက်ထုတ်ပေးမှာဖြစ်ပါတယ်။

Operator	Meaning	Example	Result
>	Greater Than	`5 > 3`	`True`
<	Less Than	`5 < 3`	`False`
>=	Greater than or equal	`5 >= 5`	`True`
<=	Less than or equal	`5 <= 5`	`False`
==	Equal	`5 == 5`	`True`
!=	Not equal	`5 != 3`	`True`

အကယ်၍ > Operator ကိုသုံးထားတဲ့ Expression တစ်ခုမှာ၊ ဘယ်ဘက်အခြမ်းက ကိန်းက ညာဘက်အခြမ်းက ကိန်းထက် ကြီးနေခဲ့ရင် `True` ကို ထုတ်ပေးမှာဖြစ်ပြီး၊ မ,ကြီးခဲ့ရင်တော့ `False` ကို ထုတ်ပေးမှာဖြစ်ပါတယ်။

comparison_operators.py

```
23 > 3
>> True

2 > 3
>> False
```

အကယ်၍ < Operator ကိုသုံးထားတဲ့ Expression တစ်ခုမှာဆိုရင်တော့၊ ဘယ်ဘက်အခြမ်းက ကိန်းက ညာဘက်အခြမ်းက ကိန်းထက် ငယ်နေခဲ့ရင် `True` ကို ထုတ်ပေးမှာဖြစ်ပြီး၊ မငယ်ခဲ့ရင်တော့ `False` ကို ထုတ်ပေးမှာဖြစ်ပါတယ်။ နမူနာကို အောက်မှာ ဖော်ပြထားပါတယ်။

```
4 < 43
>> True

27 < 8
>> False
```

ဝေါဟာရများ

Comment	ကုဒ်ရေးသားသူ နှင့် အခြားသူများ နားလည်လွယ်ကူစေရန် သို့မဟုတ် ပရိုဂရမ်က လျစ်လျူရှုစေရန်၊ ကုဒ်ကို သင်္ကေတ တစ်ခုခုဖြင့် မှတ်သား ဖော်ပြခြင်း။
Keyword	Programming တိုင်းတွင် ပါဝင်သည့် ထူးခြားသော၊ သီးခြားအဓိပ္ပာယ်ရှိသော စကားလုံး အမည်နာမများ။
Indentation	ကုဒ်လိုင်းများကို ဘယ်ဘက်အစွန်းကနေ နေရာချထားသည့် Space လေးခုစာ အကွာအဝေး ပမာဏ။
Code Block	အတူတကွ ပေါင်းစပ်ပြီး သီးခြားလုပ်ဆောင်ချက်တစ်ခုကို ပြီးမြောက်စေသော ကုဒ်အစုအဝေးတစ်ခု။ Python မှာ Indentation သုံးပြီး Code Block များကို ခွဲခြား သတ်မှတ်သည်။
Constant	မည်သည့်အခါမျှ ပြောင်းလဲခြင်းမရှိသော တန်ဖိုးတစ်ခု။
Variable	ပြောင်းလဲနိုင်သော တန်ဖိုးတစ်ခုခုကို သိမ်းဆည်ထား နိုင်သည့် အမည်တစ်ခု။
Assignment Operator	တန်ဖိုးသတ်မှတ်ရာတွင် အသုံးပြုသော `=` (ညီမျှခြင်း) သင်္ကေတ။
Increment	Variable တစ်ခု၏ တန်ဖိုးကို တိုးမြှင့်ခြင်း။
Decrement	Variable တစ်ခု၏ တန်ဖိုးကို လျှော့ချခြင်း။
Data Type	ဒေတာများ၏ အမျိုးအစား။

အခန်း (၃)

Function များ

> *“Functions should do one thing. They should do it well. They should do it only.”*
> *— Robert C. Martin (ရောဘတ် စီ. မာတင်)*

Function ဆိုတာ လုပ်ငန်းစဉ်တစ်ခု သို့မဟုတ် အလုပ်တစ်ခုခုကို လုပ်ဆောင်ဖို့ ရည်ရွယ်ပြီး သီးခြားရေးသားထားတဲ့ ကုဒ်အစုအဝေးတစ်ခုပါ။ Function တွေဟာ User ဆီက ဒေတာ အချက်အလက် Input တွေကို လက်ခံရယူနိုင်ပြီး သက်ဆိုင်ရာ ညွှန်ကြားချက် (Instruction) တွေကို လုပ်ဆောင်ပါတယ်။ လိုအပ်မယ်ဆိုရင် ရလာဒ် (Output) ကိုလဲ ပြန်ပေးနိုင်ပါတယ်။

Function ကို အလုပ်လုပ်စေဖို့အတွက် Function Calling လုပ်ဖို့ (Function ကို ခေါ်ယူသုံးစွဲဖို့) မဖြစ်မနေ လိုအပ်ပါတယ်။ Function Calling လုပ်တယ်ဆိုတာက Function ကို စတင်အလုပ်လုပ်စေဖို့အတွက် လိုအပ်တဲ့ Input တွေကို ထည့်သွင်း ခေါ်ယူလိုက်တာကို ဆိုလိုတာပါ။ ဒါမှသာ Function က သူ့ရဲ့ Instruction တွေအတိုင်း လုပ်ဆောင်ပြီး သက်ဆိုင်ရာ ရလာဒ် Output ကို တွက်ပေးနိုင်မှာပါ။

Function တွေဟာ ကျောင်းမှာသင်ခဲ့ရတဲ့ အက္ခရာသင်္ချာ (Algebra) ဘာသာရပ်က Function တွေနဲ့ ဆင်တူပါတယ်။ ဥပမာအနေနဲ့ ဒီညီမျှခြင်းကို ကြည့်ပါ။

```
# ဒါက algebra ညီမျှခြင်းပါ။ Python Code မဟုတ်ပါ
f(x) = x * 2
```

ဒီသင်္ချာညီမျှခြင်းမှာ - ဘယ်ဘက်ခြမ်းက `f(x)` ဆိုတာ `f` လို့ခေါ်တဲ့ Function တစ်ခုကို သတ်မှတ်ထားတာပါ။ ဒီ Function က `(x)` ဆိုတဲ့ Parameter တစ်ခုကို လက်ခံပါတယ်။ Parameter ဆိုတာ Function ကို ခေါ်တဲ့အခါ ထည့်သွင်းပေးရမယ့် Input တန်ဖိုးတွေပါ။ Function တစ်ခုမှာ Parameter တစ်ခုထက် ပိုပါနိုင်သလို လုံးဝမပါတာမျိုးလဲ ဖြစ်နိုင်ပါတယ်။ ညာဘက်ခြမ်းက `x * 2` ကတော့ Function ရဲ့ အဓိပ္ပာယ်ဖွင့်ဆိုချက်ပါ။ တနည်းအားဖြင့် Function ကို ခေါ်တဲ့အခါ အလုပ်လုပ်မယ့် ညွှန်ကြားချက် (Instruction) တွေပါ။ ထည့်သွင်းပေးလိုက်တဲ့ Parameter `x` တန်ဖိုးနဲ့ တွက်ချက်မှုတွေ ပြုလုပ်ပြီး ရလာဒ် Output ကို ရရှိတာမျိုးပါ။ ဒီ Function မှာဆိုရင် `x` တန်ဖိုးကို 2 နဲ့ မြှောက်ပြီး ရလာတဲ့ Output ကို တွက်ပေးနိုင်ပါတယ်။

နောက်ထပ် Parameter တစ်မျိုးကတော့ Optional Parameter တွေပါ။ ဒါတွေကတော့ Function ကို ခေါ်ယူသုံးစွဲတဲ့အခါ အခြေအနေပေါ်မူတည်ပြီး ထည့်သွင်းပေးစရာမလိုပါဘူး။ တကယ်လို့ မထည့်သွင်းခဲ့ဘူးဆိုရင်လည်း Function က ကြိုတင်သတ်မှတ်ထားတဲ့ Default တန်ဖိုးကို အလိုအလျောက် အသုံးပြုသွားမှာ ဖြစ်ပါတယ်။

Optional Parameter တွေကို အခုလိုပုံစံနဲ့ သတ်မှတ်နိုင်ပါတယ်။

```
[function_name]([parameter_name]=[parameter_value])
```

Optional Parameter ပါဝင်တဲ့ Function ကို ခေါ်ယူသုံးစွဲတဲ့ ဥပမာကို ကြည့်ရအောင်ပါ။

function_parameter.py

```python
def f(x=10):
    if x == 10:
        print("x is ten")
    else:
        print("x is not ten")

default = f()
pass_in = f(2)

# x is ten
# x is not ten
```

ပထမဆုံးအနေနဲ့ `f()` ဆိုပြီး Parameter တန်ဖိုး ဘာမှမထည့်ဘဲ Function ကို ခေါ်ပါတယ်။ `x` ကို Optional Parameter အဖြစ် သတ်မှတ်ထားတဲ့အတွက် တကူးတက ထည့်သွင်းဖို့ မလိုတော့ပါဘူး။ ဒီအခြေအနေမှာ `x` က Default တန်ဖိုးဖြစ်တဲ့ 10 ကို ရရှိပါတယ်။ ဒါကြောင့် `x` က 10 နဲ့ ညီသွားတဲ့အတွက် `"x is ten"` လို့ ထွက်လာတာပါ။ နောက်ထပ်အနေနဲ့ Function ကို `f(2)` ဆိုပြီး ထပ်ခေါ်ပါတယ်။ ဒီတစ်ခါမှာတော့ 2 ကို Parameter အဖြစ် ထည့်သွင်းပေးလိုက်တဲ့အတွက် 10 ဆိုတဲ့ Default တန်ဖိုးကို လျစ်လျူရှုပြီး `x` တန်ဖိုးက 2 ဖြစ်သွားပါတယ်။ ဒါကြောင့် `"x is not ten"` လို့ ထွက်လာတာဖြစ်ပါတယ်။

Required နဲ့ Optional Parameter တွေကို အတူတူ တွဲလျက် အသုံးပြုလို့လဲ ရပါတယ်။ ဒါပေမယ့် သတိထားရမယ့် စည်းမျဉ်းတစ်ခု ရှိပါတယ်။ ဒါကတော့ Required Parameter တွေကို အရင်ရေးပြီးမှ Optional Parameter တွေကို နောက်က ရေးရတာပါ။ နမူနာကို ကြည့်ပါ။

```python
def required_optional(x, y=10):
    return x + y
```

This is a sample preview. Get the full version of this ebook to access all chapters

User က 10 နဲ့ 0 ကို ထည့်လိုက်တဲ့အခါ၊ `try` အပိုင်းက `print(a / b)` ဆိုတဲ့ ကုဒ်မှာ `ZeroDivisionError` ဖြစ်ပေါ်လာပါတယ်။ အဲဒီလို Exception တတ်လာတဲ့အခါ `except` အပိုင်းက `print("b cannot be zero. Try again.")` ဆိုတဲ့ မက်ဆေ့ကို ဖော်ပြသွားမှာပါ။ အကယ်၍ User က 0 ကမဟုတ်ဘဲ တခြားဂဏန်းတစ်ခုခု ထည့်မယ်ဆိုရင်တော့ `try` Block ထဲက ကုဒ်က ပုံမှန်အတိုင်း အလုပ်လုပ်သွားမှာဖြစ်ပါတယ်။ `except` Block ကတော့ အလုပ်လုပ်သွားမှာ မဟုတ်ပါဘူး။ ဒါပေမယ့် 0 ထည့်လိုက်တာနဲ့ တပြိုင်နက် Except Block ထဲက ကုဒ်က အလုပ်လုပ်ပြီး User ကို သတိပေးစာ ပြသပေးလာမှာ ဖြစ်ပါတယ်။ ဒီနည်းလမ်းက Program ကို Error ကြောင့် ရပ်မသွားစေဘဲ၊ ဖြစ်လာတဲ့ ပြဿနာကို စနစ်တကျ ကိုင်တွယ်ပြီး User ကိုလည်း နားလည်လွယ်တဲ့ Message တွေ ပြန်ပေးနိုင်စေပါတယ်။

Docstrings

Python မှာ Docstring ဆိုတာ Function ဒါမှမဟုတ် Method တစ်ခုရဲ့ ထိပ်ဆုံးမှာ ရေးထားတဲ့ မှတ်ချက်တွေပါ။ ဒီ Docstring တွေက အဲဒီ Function ဒါမှမဟုတ် Method က ဘာလုပ်တယ်၊ ဘယ်လို Parameter တွေ လက်ခံပြီး ဘာတွေ ပြန်ပေးတယ်ဆိုတာကို ရှင်းပြပေးပါတယ်။ အောက်က ဥပမာကို ကြည့်ပါ။

docstring_example.py

```python
def add(x, y):
    """
    Returns x + y.
    :param x: int first integer to be added.
    :param y: int second integer to be added.
    :return: int sum of x and y.
    """
    return x + y
```

ဒီ `add` ဆိုတဲ့ Function ရဲ့ Docstring ကို သေချာကြည့်မယ်ဆိုရင် ပထမဆုံးစာကြောင်းမှာ `Returns x + y.` လို့ ရေးထားပါတယ်။ ဒါက ဒီ Function ဘာလုပ်တယ်ဆိုတာကို ရှင်းရှင်းလင်းလင်း ဖော်ပြထားတာဖြစ်ပါတယ်။ ကျန်တဲ့စာကြောင်းတွေမှာတော့ `:param x: int first integer to be added.` လိုမျိုး Function ရဲ့ Parameter တွေ၊ သူတို့ရဲ့ Type တွေနဲ့ `:return : int sum of x and y.` လိုမျိုး Function ကနေ ဘာပြန်ပေးမယ်ဆိုတာကို ဖော်ပြထားပါတယ်။

Docstring တွေက ကုဒ်တွေ ပိုပြီး မြန်မြန်ဆန်ဆန် ရေးသားနိုင်အောင် ကူညီပေးပါတယ်။ ကိုယ်ရေးခဲ့တဲ့ ကုဒ်က ဘာလုပ်တယ်ဆိုတာ မေ့သွားခဲ့ရင် Docstring ကို ဖတ်လိုက်ရုံနဲ့ အလွယ်တကူ

This is a sample preview. Get the full version of this ebook to access all chapters

အခန်း (၇)

Module များ

ကုဒ်လိုင်းပေါင်း ထောင်ချီ၊ သောင်းချီပါဝင်တဲ့ ပရောဂျက်ကြီးတစ်ခုအတွက် ကုဒ်ရေးနေတယ်လို့ မြင်ယောင်ကြည့်ပါ။ အဲဒီကုဒ်တွေအားလုံးကို ဖိုင်တစ်ခုတည်းမှာပဲ စုပြုံရေးထားမယ်ဆိုရင် ကိုယ့်ပရောဂျက်ကို ပြန်လည်စီမံခန့်ခွဲဖို့နဲ့ ပြင်ဆင်ဖို့ အလွန်ခက်ခဲပါလိမ့်မယ်။ ကုဒ်ထဲမှာ Error တစ်ခုခု ဖြစ်တိုင်း ပြဿနာရဲ့ဇာစ်မြစ်ကိုရှာဖို့ ကုဒ်လိုင်းသောင်းချီရှိတဲ့ ဖိုင်ကြီးထဲမှာ အပေါ်အောက် လိုက်ရှာနေရမှာပါ။ ဒီအခြေအနေဟာ အင်မတန် အလုပ်မဖြစ်တဲ့ပုံစံပါ။

ဒါကြောင့် ဒီပြဿနာကို ဖြေရှင်းဖို့အတွက် Programmer တွေက ပရောဂျက်ရဲ့ကုဒ်တွေကို သေးငယ်ပြီး စီမံခန့်ခွဲရလွယ်ကူတဲ့ အပိုင်းလေးတွေအဖြစ် ခွဲခြမ်းလိုက်ကြပါတယ်။ ဒီလိုခွဲခြမ်းလိုက်တဲ့ အပိုင်းတစ်ခုစီကို Module တွေထဲမှာ သိမ်းဆည်းရေးသားပါတယ်။ Module ဆိုတာ အခြေခံအားဖြင့် Python ဖိုင် (.py file) တစ်ခုကို ခေါ်တဲ့ နောက်ထပ်အမည်တစ်ခုလို့ မှတ်နိုင်ပါတယ်။

Module တစ်ခုထဲကကုဒ်ကို နောက်ဖိုင်တစ်ခုကနေ ပြန်လည်အသုံးပြုနိုင်ပါတယ်။ ဒါ့အပြင် Python ကို Install လုပ်လိုက်ကတည်းက အသင့်ခေါ်သုံးနိုင်တဲ့ မူလပါရှိပြီးသား (Built-In) Module တွေလည်း အများကြီးရှိပါတယ်။ အဲဒီ Module တွေမှာ ကျွန်တော်တို့ရဲ့ ပရိုဂရမ်အတွက် အလွယ်တကူ ခေါ်ယူသုံးစွဲနိုင်တဲ့ အသုံးဝင်တဲ့ Functionality တွေ အများအပြား ပါဝင်ပါတယ်။

မူလပါရှိပြီးသား Built-in Module များကို အသုံးပြုခြင်း

Module တစ်ခုကို ခေါ်ယူ အသုံးမပြုခင်မှာ၊ အဲဒီ Module ကို ကိုယ့်ရဲ့ ပရိုဂရမ်ထဲ အရင်ဆုံး Import လုပ်ရပါမယ်။ Import လုပ်လိုက်ပြီဆိုရင် အဲဒီ Module ထဲမှာပါတဲ့ ကုဒ်တွေ၊ Function တွေကို ကိုယ့်ပရိုဂရမ်မှာ စတင်အသုံးပြုလို့ ရသွားပါပြီ။

Module တစ်ခုကို Import လုပ်ဖို့

```
import [module_name]
```

ဆိုတဲ့ Syntax ရေးထုံးကို သုံးပါတယ်။ `import` က Keyword ဖြစ်ပြီး သူ့နောက်မှာ ကိုယ်အသုံးပြုချင်တဲ့ Module ရဲ့ နာမည်ကို ထည့်ပေးရပါတယ်။

This is a sample preview. Get the full version of this ebook to access all chapters

အခန်း (၁၁)

The Four Pillars of Object-Oriented Programming

> *“Good design adds value faster than it adds cost.”*
>
> *— Thomas C. Gale*

Object-Oriented Programming (OOP) မှာ အဓိက ဝိသေသန (၄) မျိုး ရှိပြီး၊ **The Four Pillars of Object-Oriented Programming** လို့ ခေါ်လေ့ရှိပါတယ်။ အဲဒီ လက္ခဏာတွေကတော့ **Inheritance**, **Polymorphism**, **Abstraction** နဲ့ **Encapsulation** တို့ပဲ ဖြစ်ပါတယ်။ Programming Language တစ်ခုကို OOP Language လို့ သတ်မှတ်ဖို့အတွက် ဒီဝိသေသန (၄) မျိုးလုံး ပါဝင်နေရပါမယ်။ ဥပမာ၊ Python, Java, နဲ့ Ruby တို့ဟာ OOP Language တွေ ဖြစ်ပါတယ်။ Programming Language အားလုံးကတော့ OOP ကို ထောက်ပံ့ မပေးပါဘူး။ ဥပမာအားဖြင့် Haskell ဟာ OOP ကို မထောက်ပံ့တဲ့ Functional Programming Language တစ်ခုဖြစ်ပါတယ်။ ဒီအခန်းမှာ OOP ရဲ့ အဓိက လက္ခဏာ တစ်ခုချင်းစီကို အသေးစိတ်လေ့လာသွားပါမယ်။

Inheritance

Inheritance ကို တိုက်ရိုက်သာသာပြန်ရရင် အမွေဆက်ခံခြင်းလို့ ဆိုပါတယ်။ Programming မှာလဲ Inheritance က မျိုးရိုးဗီဇဆိုင်ရာ အမွေဆက်ခံခြင်းနဲ့ လုံးဝ ဆင်တူပါတယ်။ မျိုးရိုးဗီဇအရ သင့်မိဘတွေဆီက တူညီတဲ့ မျက်လုံးအရောင်ကို လက်ခံရရှိထားသလိုမျိုး အလားတူပါပဲ၊ Class တစ်ခုကို ဖန်တီးတဲ့အခါမှာ အခြား Class တစ်ခုဆီကနေ အမွေဆက်ခံနိုင်ပါတယ်။ အဲဒီအခြား Class ကိုတော့ Parent Class လို့ ခေါ်ပါတယ်။ အဲဒီလို Inheritance ပြုလုပ်လိုက်တဲ့အခါ Class အသစ်ဟာ Parent Class ရဲ့ Variable တွေ နဲ့ Method တွေကို လက်ခံရရှိလာမှာ ဖြစ်ပါတယ်။ ဒီသဘောကို ပိုပြီးနားလည်သွားအောင် နမူနာတစ်ခုကို လေ့လာကြည့်ပါမယ်။ ဒီနမူနာမှာ လူကြီးတစ်ယောက်၊ ကလေးတစ်ယောက်ကို Inheritance သုံးပြီး ပုံဖော်ကြည့်ပါမယ်။ လူကြီးကို ကိုယ်စားပြုတဲ့ Class တစ်ခုကို အခုလို သတ်မှတ်ပါမယ်။

```
class Adult():
    def __init__(self, name, height, weight, eye_color):
        self.name = name
        self.height = height
        self.weight = weight
        self.eye_color = eye_color
```

This is a sample preview. Get the full version of this ebook to access all chapters

အခန်း (၁၂)

More Object-oriented Programming

> *“Commenting your code is like cleaning your bathroom –*
> *you never want to do it, but it really does create*
> *a more pleasant experience for you and your guests.”*
> *— Ryan Campbell*

ဒီအခန်းမှာ Object-Oriented Programming နဲ့ပတ်သက်တဲ့ နောက်ထပ်အယူအဆတွေကို ဆက်လက်ဖော်ပြသွားမှာပါ။ ဒီဥပမာတွေထဲက တချို့က Python အတွက်ပဲ သီးသန့်ဖြစ်ပေမယ့် အများစုကတော့ Object-Oriented Programming ကို ပံ့ပိုးပေးတဲ့ တခြားသော Programming Language တွေမှာလည်း အသုံးပြုလို့ရပါတယ်။

Variable တွေ ဘယ်လိုအလုပ်လုပ်လဲ

ဒီအပိုင်းမှာ Variable တွေအကြောင်း ပိုပြီးလေ့လာသွားပါမယ်။ Variable တစ်ခုဟာ Object တစ်ခုကို "Point" လုပ်ပါတယ်။

```
number = 100
```

အပေါ်ကကုဒ်မှာ `number` ဟာ Value 100 ရှိတဲ့ Integer Object တစ်ခုကို Point လုပ်ပါတယ်။

```
number = 101
```

ဒီကုဒ်မှာတော့ `number` ကို Value အသစ်တစ်ခု သတ်မှတ်ပေးလိုက်တဲ့အတွက်၊ နောက်ထပ် Value 101 ရှိတဲ့ Integer Object အသစ် တစ်ခုကို Point လုပ်သွားပါတယ်။ အရင်က Value 100 ရှိတဲ့ Integer Object အဟောင်းကိုတော့ ဆက်မသုံးတော့တဲ့အတွက် ပယ်ဖျက်လိုက်မှာ ဖြစ်ပါတယ်။

Variable နှစ်ခုဟာ တူညီတဲ့ Object တစ်ခုတည်းကို Point လုပ်နိုင်ပါတယ်။

```
x = 100
y = x
```

This is a sample preview. Get the full version of this ebook to access all chapters

အခန်း (၁၅)

Version Control

> *“Every programmer is an author.”*
>
> *— Sercan Leylek*

Software ရေးတယ်ဆိုတာ အဖွဲ့လိုက်လုပ်ဆောင်ရတဲ့ လုပ်ငန်းတစ်ခုပါ။ ပရောဂျက်တစ်ခုမှာ Team Member အားလုံးက Codebase တစ်ခုတည်းကို အတူတူ ပြင်ဆင်ရေးသားကြရမှာပါ။ အဲဒီလို ပြင်ဆင်ပြောင်းလဲလိုက်တာတွေကိုလည်း တစ်ယောက်နဲ့ တစ်ယောက် အချိန်နဲ့တပြေးညီ သိရှိနေဖို့ လိုအပ်ပါတယ်။ ဆိုပါစို့၊ အဲဒီ ကုဒ်အပြောင်းအလဲတွေကို အီးမေးလ်နဲ့ အပြန်အလှန် ပို့ပေးပြီး ကိုယ့်ဘာသာကိုယ် ပေါင်းစပ်ရေးသားရမယ် ဆိုရင် အင်မတန် အလုပ်ရှုပ်ပြီး အဆင်မပြေဖြစ်လာမှာပါ။ အဲဒီအပြင် လူနှစ်ယောက်က ပရောဂျက်ရဲ့ ကုဒ်နေရာတစ်ခုတည်းကို တစ်ချိန်တည်းမှာ ပြင်လိုက်မိရင် ဘယ်သူ့ရဲ့ အပြောင်းအလဲကို အတည်ပြုရမလဲဆိုတဲ့ ပြဿနာလည်း ရှိလာနိုင်ပါတယ်။

ဒီလိုပြဿနာတွေကို ဖြေရှင်းပေးနိုင်တာကတော့ Version Control System (VCS) ဖြစ်ပါတယ်။ Version Control System ဆိုတာက Software Project တွေကို အတူတကွ ပူးပေါင်းလုပ်ဆောင်ကြတဲ့အခါ အလွယ်တကူ ပေါင်းစပ်ရေးသားနိုင်စေဖို့ တည်ဆောက်ထားတဲ့ Software တစ်မျိုး ဖြစ်ပါတယ်။

Git နဲ့ **SVN** တို့လို နာမည်ကြီးတဲ့ Version Control System တွေ အများကြီးရှိပါတယ်။ ဒီ System တွေဟာ များသောအားဖြင့် Github တို့လို ကုဒ်တွေကို Cloud ပေါ်မှာ သိမ်းဆည်းပေးထားနိုင်တဲ့ Platform တွေနဲ့ တွဲဖက်ပြီး အသုံးပြုရပါတယ်။ ဒီအခန်းမှာတော့ Version Control System တစ်ခုဖြစ်တဲ့ Git နဲ့၊ Cloud ပေါ်မှာ ကုဒ်တွေသိမ်းဆည်းပေးနိုင်တဲ့ Github အသုံးပြုနည်းကို လေ့လာသွားမှာ ဖြစ်ပါတယ်။

အခန်း (၁၆)

Network Programming

ဒီအခန်းမှာတော့ Network ပေါ်ကနေ ကွန်ပျူတာတွေ တစ်ခုနဲ့တစ်ခု ဘယ်လို ဆက်သွယ်ကြတယ်ဆိုတာကို လေ့လာသွားပါမယ်။ Network ဆိုတာကတော့ မက်ဆေ့ချ်တွေ ဖလှယ်နိုင်ဖို့အတွက် Software နဲ့ Hardware တွေ အသုံးပြုပြီး ချိတ်ဆက်ထားတဲ့ ကွန်ပျူတာ ကွန်ရက်တစ်ခုကို ဆိုလိုတာပါ။ အင်တာနက်ဟာလည်း Network တစ်ခုဖြစ်ပါတယ်။

ဒီအခန်းမှာ Client-Server Model နဲ့ TCP/IP Protocol လို့ခေါ်တဲ့ အင်တာနက်ရဲ့ အခြေခံသဘောတရားတွေကို အရင်ဆုံး ဆွေးနွေးပါမယ်။ အဲဒီနောက် Client နဲ့ Server နှစ်ခုလုံးကို တည်ဆောက်ပြီး လက်တွေ့ကျကျ လေ့လာသွားပါမယ်။

Client-Server ပုံစံ

အင်တာနက်ဟာ Client-Server Model ကို အသုံးပြုပြီး ဆက်သွယ်ပါတယ်။ ဒီပုံစံမှာ Server က Request တွေကို အမြဲ Listen လုပ်နေရပါတယ်၊ ဆိုလိုတာက Client (Browser) ကနေ ပေးပို့လာမယ့် Request တွေကို အမြဲ စောင့်ဆိုင်းနေရတာပါ။ Google Server ကို နမူနာအဖြစ် ကြည့်လို့ရပါတယ်။

Client တွေကတော့ Webpage တစ်ခုကို ပြသပေးဖို့အတွက် လိုအပ်တဲ့ Resource ဒေတာ တွေကို Server ဆီ တောင်းခံပါတယ်။ Server က အဲဒီလိုအပ်တဲ့ Resource တွေကို Client (Browser) ဆီ ပြန်ပို့ပေးရပါတယ်။

ဒီ Request တွေကို HTTP (Hypertext Transfer Protocol) ကို အသုံးပြုပြီး ပေးပို့ကြပါတယ်။ ဒီလင့်ကနေ HTTP အကြောင်း ဆောင်းပါး အပြည့်အစုံကို ဝင်ရောက် ဖတ်ရှုနိုင်ပါတယ်။ HTTP (Hypertext Transfer Protocol) ဆိုတာ ဘာလဲ။ ဒီနေရာမှာ Resource ဆိုတာကတော့ HTML, Javascript, CSS ဖိုင်တွေအပြင် Image ၊ Video စသည်ဖြင့် Website တစ်ခုကို ပြသဖို့အတွက် Browser မှာ လိုအပ်တဲ့အရာတွေကို ဆိုလိုပါတယ်။

အခန်း (၁၇)

Flask မိတ်ဆက်

ဒီအခန်းမှာတော့ Python ရဲ့ လူသုံးများတဲ့ Web Framework တစ်ခုဖြစ်တဲ့ Flask အကြောင်းကို လေ့လာသွားကြပါမယ်။ Flask ကို အသုံးပြုပြီး Web Applications တွေကို ဘယ်လိုမျိုး ပေါ့ပေါ့ပါးပါး ဖန်တီးနိုင်လဲဆိုတာကို နမူနာကုဒ်တွေနဲ့တကွ လေ့လာသွားမှာ ဖြစ်ပါတယ်။

Flask ဆိုတာ ဘာလဲ

Flask ဆိုတာ Python ကို အသုံးပြုပြီး Web Applications တွေ ရေးသားနိုင်ဖို့ ကူညီထောက်ပံ့ပေးတဲ့ Micro Web Framework တစ်ခု ဖြစ်ပါတယ်။ "Micro" လို့ ဆိုရတဲ့ အကြောင်းအရင်းကတော့ Web Framework တွေမှာပါလေ့ရှိတဲ့ Database Access Layer ဒါမှမဟုတ် Form Validation လိုမျိုး အရာတွေကို တခါတည်း ထည့်သွင်းမထားတဲ့အတွက်ပါ။ ဒါပေမဲ့ ပရောဂျက်ပေါ်မူတည်ပြီး လိုအပ်သလို Functionality တွေကို အလွယ်တကူ ထပ်ထည့်သွားနိုင်ပါတယ်။

Flask က အသုံးပြုရတာ ရိုးရှင်းပြီး၊ သင်ယူရတာလည်း လွယ်ကူပါတယ်။ သူ့ရဲ့ Minimalist ဒီဇိုင်းက Developer တွေကို လိုအပ်သလို ပြုပြင်ဖန်တီးနိုင်စေပါတယ်။ ဒါ့အပြင် သူ့ရဲ့ Extensibility ကြောင့်လည်း လိုအပ်တဲ့ အရာတွေကို ထပ်ပြီး ထည့်လာနိုင်စေပါတယ်။

Flask ကို Install လုပ်ခြင်း

Flask နဲ့ စတင်အလုပ်လုပ်ဖို့အတွက် ပထမဆုံး Virtual Environment တစ်ခု ဖန်တီးပြီး Flask ကို Install လုပ်ရပါမယ်။ Virtual Environment ဆိုတာက Project တစ်ခုချင်းစီအတွက် သီးခြား Enviroment တစ်ခု ဖန်တီးပေးတာဖြစ်ပြီး၊ Library တွေ မရောထွေးအောင် ကာကွယ်ပေးပါတယ်။

Virtual Environment တစ်ခုဖန်တီးဖို့ Terminal မှာ အခုလို ရိုက်ပါ။

```
python -m venv myvenv
```

This is a sample preview. Get the full version of this ebook to access all chapters

အခန်း (၁၈)

Database

Database ဆိုတာ ဒေတာ အချက်အလက်တွေကို မပျောက်မပျက် အမြဲသိမ်းဆည်းထားနိုင်ဖို့ အသုံးပြုတဲ့ ပရိုဂရမ်တွေ ဖြစ်ပါတယ်။ Database ထဲက ဒေတာတွေက Persist ဖြစ်ပါတယ်။ ဆိုလိုတာက ဒေတာတွေဟာ သူတို့ကို ဖန်တီးပြီးသွားတဲ့နောက်ပိုင်း မပျောက်မပျက် ဆက်ရှိနေသေးတာပါ။

အခုချိန်ထိ ကျွန်တော်တို့ နမူနာ ရေးသားခဲ့တဲ့ ပရိုဂရမ်အများစုဟာ ဒေတာတွေကို အမြဲသိမ်းဆည်းစရာမလိုဘဲ အလုပ်လုပ်နေတဲ့ ပရိုဂရမ်တွေပါ။ အကယ်၍ User ရဲ့ အချက်အလက်တွေ တောင်းခံတဲ့ ပရိုဂရမ်တစ်ခုက တစ်ခါ Run တိုင်း User ဆီက ထပ်ခါထပ်ခါ တောင်းနေလို့ အဆင်မပြေပါဘူး။ ပထမဆုံး တစ်ခါတောင်းပြီး Database ထဲမှာ ထည့် သိမ်းဆည်းထားလိုက်ရင် အဆင်ပြေပါပြီ။

Database တွေမှာ အဓိကလုပ်ဆောင်ချက် နှစ်ခုရှိပါတယ်။ Read နဲ့ Write ပါ။ Write လုပ်တယ်ဆိုတာ ဒေတာ အချက်အလက်တွေကို သိမ်းဆည်းဖို့ ညွှန်းကြားတာပါ။ Read လုပ်တာကတော့ သိမ်းဆည်းထားတဲ့ ဒေတာအချက်အလက်တွေကို ပြန်ထုတ်ယူတာပါ။

ဒေတာ အချက်အလက်တွေ လည်ပတ်နေတဲ့ ကမ္ဘာကြီးမှာ Database တွေက အင်မတန် အရေးပါ ပါတယ်။ ဒီအခန်းမှာ Database တွေအကြောင်းကို စတင် မိတ်ဆက်ပေးသွားပါမယ်။

NoSQL နဲ့ SQL

Relational Database ကို **Edgar F. Codd** က ၁၉၇၀ မှာ စတင်အဆိုပြုခဲ့ပါတယ်။ Relational Database တွေဟာ ဒေတာအချက်အလက်တွေကို Excel Sheet လိုမျိုး Row တွေ၊ Column တွေနဲ့ သိမ်းဆည်းပါတယ်။ လွယ်လွယ်ပြောရရင် Table Format နဲ့ သိမ်းဆည်းတာပါ။ အဲဒီလို သိမ်းဆည်းဖို့ **SQL လို့ခေါ်တဲ့ Structured Query Language** ကို အသုံးပြုပါတယ်။ အလားတူပါပဲ ပြန်လည်ထုတ်ယူဖို့အတွက်လည်း SQL ကို သုံးပါတယ်။ လူသုံးများတဲ့ Relational Database တွေထဲမှာ PostgreSQL, MySQL နဲ့ SQLite တို့ ပါဝင်တယ်။

ရှိရှိသမျှ Row တွေ အားလုံးကို ရွေးထုတ်ယူပေးနိုင်တဲ့ General SQL Syntax Pattern က အခုလိုပါ။

```
SELECT *
FROM table_name;
```

Column တန်ဖိုးနဲ့ စစ်ထုတ်ထားတဲ့ Row တွေကို ရယူပေးနိုင်တဲ့ General SQL Syntax Pattern က အခုလိုပါ။

```
SELECT *
FROM table_name
WHERE column_name = 'value1';
```

OR

SQL Query မှာ `OR` Keyword ကို ထည့်သွင်းပြီး Condition တစ်ခု သို့မဟုတ် တခြား တစ်ခုကို ပြေလည်တဲ့ Row တွေကို ဆွဲထုတ်ယူနိုင်ပါတယ်။ `OR` Keyword ကို WHERE Clause မှာ ထည့်သွင်း အသုံးပြုရပါတယ်။ နမူနာ အနေနဲ့ ကျွန်တော်တို့ရဲ့ `bdfl` Table ထဲကနေ 'Ruby on Rails' ဒါမှမဟုတ် 'Wordpress' ပရောဂျက်တွေ ပါဝင်တဲ့ ဒေတာ Row တွေကိုပဲ ရွေးထုတ်ယူချင်တယ် ဆိုရင်တော့ အခုလို အသုံးပြုလို့ ရပါတယ်။

```
SELECT * FROM bdfl WHERE project = "Linux" OR project="Ruby on Rails";
```

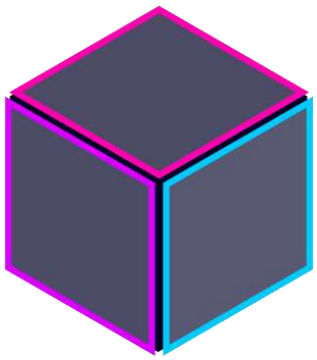
Column တန်ဖိုးတစ်ခု သို့မဟုတ် အခြားတစ်ခုနဲ့ စစ်ထုတ်ထားတဲ့ Row တွေကို ရယူပေးနိုင်တဲ့ General SQL Syntax Pattern က အခုလိုပါ။

```
SELECT *
FROM table_name
WHERE column_name = 'value1'
   OR column_name = 'value2';
```

LIKE

`LIKE` က Text စာသားတွေကို စစ်ဆေးဖို့ သုံးပါတယ်။ `LIKE` Keyword ကို WHERE Clause မှာ ထည့်သွင်း အသုံးပြုရပါတယ်။ ပထမဆုံး အောက်က Query ကို စမ်းကြည့်ပါ။

This is a sample preview. Get the full version of this ebook to access all chapters



CODE
with Thura