



Programming for beginner

မြန်မာလို ရေးသားထားတဲ့ programming ကို စတင်လေ့လာတဲ့ စာအုပ်ကို မတွေ့ ဖြစ်တာနဲ့ ဒီ စာအုပ်ကို ရေးမယ်လို့ ဆုံးဖြတ်ဖြစ်တာပါ။ ဒီ စာအုပ်ဟာ programming ဆိုတာ ဘာမှန်း မသိသေးတဲ့ သူများ အတွက် ရည်ရွယ်ပါတယ်။ ဒီ စာအုပ်ဖတ်ပြီးရင် program တွေ ရေးလို့ ရမလား ဆိုတော့ ရတယ်လည်း ဆိုလို့ ရသလို မရဘူးလည်း ဆိုလို့ ရပါတယ်။ ဒီ စာအုပ်ဟာ အခြေခံ ဖြစ်တဲ့ အတွက် အခြေခံ သဘောတရားကို အဓိက ထား ရေးသားထားပါတယ်။ Programming အတွက် python language ကို အသုံးပြုပြီး ရေးသားထားပါတယ်။

Saturngod

Contents

- Introduction
- 1. Chapter 1
 - 1.1. Programming
 - 1.2. Programming Language
 - 1.3. Sequential
 - 1.4. Variable
 - 1.5. Operators
 - 1.6. Problem Solving
 - 1.7. Installing Python 3
- 2. Chapter 2
 - 2.1. Pseudo Code
 - 2.2. Flowchart
 - 2.3. Hello World
 - 2.4. What is your name ?
 - 2.5. SUM
 - 2.6. Condition
 - 2.7. Calculator
 - 2.8. Looping
 - 2.9. Array
 - 2.10. Function
 - 2.11. Exercise Answers
- 3. Chapter 3
 - 3.1. Overview
 - 3.2. Classes
 - 3.3. Inheritance
- 4. Chapter 3
 - 4.1. Basic Data Structures
 - 4.2. Stack
 - 4.3. Stack Abstract Data Type
 - 4.4. Implementing A Stack

Introduction

မင်္ဂလာပါ။

မြန်မာလို ရေးသားထားတဲ့ programming ကို စတင်လေ့လာတဲ့ စာအုပ်ကို မတွေ့ ဖြစ်တာနဲ့ ဒီ စာအုပ်ကို ရေးမယ်လို့ ဆုံးဖြတ်ဖြစ်တာပါ။ ဒီ စာအုပ်ဟာ programming ဆိုတာ ဘာမှန်း မသိသေးတဲ့ သူများ အတွက် ရည်ရွယ်ပါတယ်။ ဒီ စာအုပ်ဖတ်ပြီးရင် program တွေ ရေးလို့ ရမလား ဆိုတော့ ရတယ်လည်း ဆိုလို့ ရသလို မရဘူးလည်း ဆိုလို့ ရပါတယ်။ ဒီ စာအုပ်ဟာ အခြေခံ ဖြစ်တဲ့ အတွက် အခြေခံ သဘောတရားကို အဓိက ထား ရေးသားထားပါတယ်။ Programming အတွက် python language ကို အသုံးပြုပြီး ရေးသားထားပါတယ်။ သို့ပေမယ့် windows form တွေ ui button တွေ စသည့် UI ပိုင်းဆိုင်ရာတွေ မပါဝင်ပါဘူး။ Database ပိုင်းဆိုင်ရာတွေ လည်း ပါဝင်မှာ မဟုတ်ပါဘူး။

စာအုပ်ဟာ အခြေခံ ပိုင်းဆိုင်ရာ ကို အဓိက ထားတဲ့အတွက် Python language သင်ကြားပေးတဲ့ စာအုပ်မဟုတ်တာကို သတိပြုစေလိုပါတယ်။ Programming ဆိုတာ ဘာလဲဆိုတာ သိချင် စမ်းချင်သူတွေ ၊ နောက်ပြီး Programming ဆိုတာ ကို လေ့လာချင်သူတွေ အတွက် အခြေခံ ပိုင်း ဆိုင်ရာတွေ ရေးသားထားပါတယ်။ အခြေခံတွေ ဖြစ်တဲ့ အတွက်ကြောင့် ဒီ စာအုပ် ဖတ်ပြီးတာနဲ့ လုပ်ငန်းခွင် ဝင်လို့ မရပါဘူး။ တခြား နှစ်သက်ရာ language တွေကို စပြီး လေ့လာနိုင်အောင် တော့ အထောက် အကူပြုမယ်လို့ မျှော်လင့်ပါတယ်။

Programming အနေနဲ့ လုပ်ငန်းခွင် ဝင်ဖို့အတွက် အနည်းဆုံး ၁ နှစ်လောက် လေ့လာဖို့ လိုပါတယ်။ ဒီစာအုပ်ဟာ programming ကို လေ့လာလိုသူတွေ အတွက် ပထမဆုံး လေ့ကားထစ် တစ်ခုမျှသာ ဖြစ်ပါတယ်။ Programming ကို ရေးသားရာမှာ စဉ်းစား တွေးခေါ်တတ်ဖို့ ပြဿနာတွေ ဖြေရှင်းတတ်ဖို့ အတွက် အခြေခံ အဆင့် အဖြစ်သာ ရှိပါတယ်။ ဒီ စာအုပ်ကို ပြီးအောင် ဖတ်ဖြစ်ခဲ့ရင်တော့ နောက် အဆင့်တွေကို လွယ်လင့် တကူ လေ့လာနိုင်မယ်လို့ မျှော်လင့် ပါတယ်။

Chapter 1 : Programming ဆိုတာ

Programming ဆိုတာကတော့ process တွေ ဖြစ်ပြီးတော့ အလုပ်ပြီးမြောက်အောင် computer ကို ခိုင်းစေခြင်း ဖြစ်ပါတယ်။ ကျွန်တော်တို့ အသုံးပြုနေတဲ့ OS , Microsoft Word , Viber Messenger စတာတွေက programming ကို အသုံးပြုပြီးတော့ ရေးသားထားခြင်း ဖြစ်ပါတယ်။

ဒီစာအုပ်မှာတော့ Python 3 ကို အသုံးပြုပြီး သင်ကြားပါမယ်။ Python 3 ကို အဓိက သင်ရတဲ့ ရည်ရွယ်ချက်ကတော့ ရိုးရှင်း လွယ်ကူသည့်အတွက် programming မသိတဲ့ သူတွေ အနေနဲ့ လွယ်ကူစွာ လေ့လာနိုင်ပါတယ်။ ဒီစာအုပ်မှာ python 3 ကို run time ထည့်ထားပေးတဲ့ အတွက် python 3 ကို စက်ထဲမှာ မထည့်ထားပဲ စမ်းလို့ ရပါတယ်။

အခု Chapter မှာတော့ အခြေခံ အဆင့်တွေ ရေးသား သွားမှာ ဖြစ်တဲ့ အတွက် နားလည် သဘောပေါက်ဖို့ အရမ်း အရေးကြီးပါတယ်။ နားမလည်တာတွေကို Github မှာ [issue](#) ဖွင့်ပြီးတော့ မေးမြန်းနိုင်ပါတယ်။

1.1 Programming ဆိုတာ

Programming ဆိုတာ ဘာလဲ ဆိုတဲ့ မေးခွန်းက စတင်လေ့လာမယ့် သူတွေ အတွက် မေးနေကျ မေးခွန်းပါပဲ။

ကျွန်တော်တို့ Computer မသုံးရင်တောင် Phone တွေကို နေ့စဉ် အသုံးပြုဖူးမှာပါ။ ကျွန်တော်တို့ phone တွေ အသုံးပြုရင် App တွေကိုလည်း အသုံးပြုမိကြမှာပါ။ App တွေက ကျွန်တော်တို့ အတွက် မျက်လှည့် ပစ္စည်းလိုပါပဲ။ လိုချင်တာတွေကို ထွက်လာဖို့ screen ပေါ်မှာ လက်နဲ့ နှိပ်လိုက်ရုံပါပဲ။

Programmer တွေ က App တွေ Program တွေကို ရေးစွဲထားပြီးတော့ အသုံးပြုတဲ့ အခါမှာ လွယ်ကူအောင် ဖန်တီးထားကြပါတယ်။ Programmer တွေဟာ programming language တစ်ခုခုကို အသုံးပြုပြီး app တွေကို ဖန်တီးကြပါတယ်။ Programming language ကို အသုံးပြုပြီး program တွေကို ရေးသားပြီး နောက်ဆုံး App အနေနဲ့ ထွက်လာတာပါ။

Game တွေဟာလည်း programming language နဲ့ ရေးသားထားပါတယ်။ ဒါကြောင့် App တွေ Game တွေကို ဖန်တီးချင်တယ်ဆိုရင် Programming ကို သိဖို့ လိုအပ်ပါတယ်။

ဘယ်လို အလုပ်လုပ်လဲ ?

Computer ဟာ အလိုအလျောက် အလုပ်မလုပ်နိုင်ပါဘူး။ Computer နားလည်တဲ့ ဘာသာစကား နဲ့ computer ကို ခိုင်းစေရပါတယ်။ ဥပမာ။။ Viber မှာ call ကို နှိပ်လိုက်ရင် ဒီလူ ရဲ့ ဖုန်းကို သွားခေါ်ဆိုပြီး ရေးသားထားရပါတယ်။ ဒါမှ သုံးစွဲ သူက Call ဆိုတဲ့ ခလုတ်ကို နှိပ်လိုက်တဲ့ အခါမှာ ဖုန်း သွားခေါ်ပေးပါတယ်။

Microsoft Words မှာလည်း ထိုနည်းတူပါပဲ။ Print ဆိုတာကို နှိပ်လိုက်ရင် printer ကနေ စာထွက်လာအောင် ဆိုပြီး programming နဲ့ ရေးသားထားရပါတယ်။ သုံးစွဲ သူတွေ အနေနဲ့ ကတော့ print ဆိုတာကို နှိပ်လိုက်တာနဲ့ printer ကနေ print ထုတ်ပေးပါတယ်။

Computer ဟာ 0 နဲ့ 1 ကို သာ သိပါတယ်။ ကျွန်တော်တို့ အနေနဲ့ 0 နဲ့ 1 နဲ့ ရေးဖို့ အရာမှာ မလွယ်ကူလှတဲ့ အတွက် high level language တွေကို အသုံးပြုပြီး computer ကို ခိုင်းစေအောင် ရေးသားကြပါတယ်။ Computer ကို ခိုင်းစေတတ်တဲ့သူဟာ programmer ဖြစ်လာပါတယ်။

Programmer ဟာ သုံးစွဲ သူ နဲ့ computer ကြားမှာ ကြားခံ အနေနဲ့ သုံးစွဲ သူ ခိုင်းစေလိုတာတွေကို computer နားလည်အောင် ရေးသားပေးရတဲ့ သူပါ။ Programming language ကတော့ ဘာသာစကား တစ်ခုပါပဲ။ computer နဲ့ programmer ကြားမှာ ဆက်သွယ်ပေးတဲ့ ဘာသာစကားပါ။ Computer ဟာ အလိုအလျောက် ဘာမှ မလုပ်နိုင်ပါဘူး။ Programmer ဟာ computer ကို ဒါလုပ် ဒါလုပ် စသည် ဖြင့် ခိုင်းစေရပါတယ်။



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

run

အောက်က code မှာ computer ကို screen ပေါ်မှာ Hello World! ဆိုပြီး ရိုက်ပြခိုင်းပါတယ်။ Run ဆိုတာလေးကို နှိပ်ကြည့်လိုက်ပါ။ Hello World! ဆိုပြီး ပေါ်လာတာ တွေ့ရပါလိမ့်မယ်။

1.2 Programming Language

Programming ကို ရေးသားရာမှာ သက်ဆိုင် ရာ ဘာသာ စကားနဲ့ ရေးသားရပါတယ်။ Computer ဟာ 0 နဲ့ 1 ကိုပဲ သိပါတယ်။ 0 နဲ့ 1 ကို နားလည်အောင် ကြားခံ ဘာသာစကား တစ်ခု ကို အသုံးပြုပေးရပါတယ်။ ထိုမှသာ computer က နားလည်ပြီး မိမိ လိုအပ်တာတွေကို ဖန်တီးနိုင်ပါလိမ့်မယ်။

Generation

programming language generation နဲ့ ပတ်သက်ပြီးတော့ programming ကို စတင် သင်တဲ့ သူတွေ တော်တော်များများ သိထားသင့်ပါတယ်။ မသိလို့ ဘာဖြစ်လည်း ဆိုတော့ ဘာမှတော့ မဖြစ်ပါဘူး။ သိထားတော့ လက်ရှိ ကိုယ် သုံးနေတာ ဘယ် generation ရောက်နေပြီလဲ။ ဒီ generation မတိုင်ခင်က ဘယ် language တွေ ရှိခဲ့လဲ။ အခု ကိုယ်လေ့လာနေတာက ဘယ် generation လဲ။ စတာတွေကို သိရှိနိုင်ပါတယ်။

First Generation Language (1GL)

1950 မတိုင်ခင်က UNIVAC I နဲ့ IBM 701 တို့ဟာ ပထမဆုံး machine language program လို့ ဆိုလို့ရပါတယ်။ သို့ပေမယ့် 1GL ဟာ လျင်မြန်စွာ ကုန်ဆုံးသွားပြီး 2GL ကို ကူးပြောင်းလာခဲ့ပါတယ်။

Second Generation Language (2GL)

2GL ကတော့ လူသိများတဲ့ assembly language သို့မဟုတ် assembler ပေါ့။ assembler ကတော့ အခုထက်ထိတော့ အချို့နေရာတွေမှာ အသုံးချနေဆဲပါပဲ။

Third Generation Language (3GL)

အဲဒီနောက်ပိုင်းမှာတော့ 3GL တွေ ဖြစ်တဲ့ FORTRAN , LISP, COBOL တွေ ထွက်ခဲ့ပါတယ်။ 3GL ဟာ ပိုမို ရေးသားရမှာ လွယ်ကူလာပြီး အရင်တုန်းက machine code တွေနဲ့ မတူညီတော့ပါဘူး။ 3GL ဟာ general use အနေနဲ့ အသုံးပြုလာနိုင်ခဲ့ပါတယ်။ 3GL နဲ့ အတူတူ general purpos language တွေကိုလည်း ပေါ်ထွက်လာခဲ့ပါတယ်။

C language ကို 1969 နဲ့ 1973 ကြားမှာ developed လုပ်ခဲ့ပြီးတော့ အခုအချိန်ထိ popular ဖြစ်နေသေးတဲ့ language တစ်ခုပါ။ C ကို ထပ်ပြီးတော့ version အသစ်တိုးကာ 1980 မှာ C++ ကို ထုတ်ခဲ့ပါတယ်။ C++ က object-oriented နဲ့ system programming တွေ ပါဝင်လာပါတယ်။

Third Generation နဲ့ အတူ လက်ရှိ အသုံးပြုနေတဲ့ general purpose programming language တွေကတော့ PHP, ASP, C, C++, Java, Javascript, Perl, Python, Pascal, Fortran တို့ ဖြစ်ပြီး သူတို့ဟာလည်း Third generation Language တွေပါပဲ။

Fourth Generation Language (4GL)

Fourth generation language ကိုတော့ စီးပွားရေးဆိုင်ရာ business software တွေအတွက် ရည်ရွယ်ပြီး ဖန်တီးခဲ့ကြပါတယ်။ အချို့ 3GL ဟာ 4GL ထဲမှာ General Use အနေနဲ့ ပါဝင်လာပါတယ်။

အောက်မှာ ဥပမာ အချို့ ဖော်ပြပေးထားပါတယ်။

- General Use
 - Perl
 - Python
 - Ruby
- Database
 - SQL

- Report generators
 - Oracle Report
- Data manipulation, analysis, and reporting languages
 - SQL PL
 - SPSS
- GUI creators
 - XUL
 - OpenROAD
- Mathematical optimization
 - AIMMS
 - GAMS
- Database-driven GUI application development
 - Action Request System
 - C/AL
- Screen painters and generators
 - SB+/SystemBuilder
 - Oracle Forms
- Web development languages
 - CFML

Fifth Generation Language (5GL)

5GL ကတော့ အဓိကအားဖြင့် programmer မလိုပဲနဲ့ program တွေကို တည်ဆောက်ဖို့အတွက် ရည်ရွယ်ထားတာပါ။ 5GL တွေကို အဓိကအားဖြင့် Artificial Intelligence research တွေ မှာ အဓိက အသုံးပြုပါတယ်။ Prolog , OPS5, Mercury တို့က 5GL example တွေပေါ့။

ref: [Wikipedia](#)

1.3 Sequential

Programming မှာ code တွေက တစ်ကြောင်းခြင်းစီ အလုပ်လုပ်ပါတယ်။ တစ်ခုပြီးမှ နောက်တစ်ခုက အလုပ်လုပ်တယ်။

ဥပမာ အောက်က code လေးကို တချက်ကြည့်လိုက်ပါ။

```
i = 5 + 4
```

```
i = i + 6
```

5 နဲ့ 4 ကို ပေါင်းပြီးတော့ i ထဲကို ထည့်တယ်။ ပြီးမှ ရလာတဲ့ အဖြေကို 6 နဲ့ ပေါင်းတယ်။ အဲလို တစ်ကြောင်းစီ အလုပ်လုပ်ပေးပါတယ်။ 5+4 ကို ပေါင်းတာ မပြီးသေးပဲ 6 နဲ့ သွားပေါင်းသည့် အဆင့်ကို မကျော်သွားပါဘူး။

ဒါကြောင့် programming အတွက် စဉ်းစားသည့် အခါမှာ တဆင့်ပြီး တဆင့် စဉ်းစားပြီးတော့ ရေးရပါတယ်။ ကျွန်တော်တို့ ခိုင်းလိုတဲ့ အရာတွေကို တဆင့်ပြီးတဆင့် ရေးသားပြီးတော့ ခိုင်းစေရပါတယ်။ ထို့မှသာ ကျွန်တော်တို့ လိုချင်တဲ့ ရလဒ် ကို ရရှိမှာ ဖြစ်ပါတယ်။

1.4 Variable

Programming ကို စလေ့လာတော့မယ်ဆိုရင် ပထမဆုံး သိဖို့လိုတာကတော့ variable ပါပဲ။ variable ဆိုတာကတော့ data တွေကို ခဏတာ memory ပေါ်မှာ သိမ်းထားပေးတဲ့ နေရာပေါ့။ variable ကို နာမည်ပေးဖို့ လိုပါတယ်။ ဒါ့အပြင် variable အမျိုးအစား သတ်မှတ်ပေးဖို့လည်း လိုအပ်ပါတယ်။

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

run

အထက်ပါ code မှာ ဘာ variable မှ မပါပါဘူး။

```

1 counter = 100          # An integer assignment
2 miles   = 1000.0       # A floating point
3 name    = "John"       # A string
4 boolean = True         # Boolean Value True and False only
5
6 print (counter)
7 print (miles)
8 print (name)
9 print(boolean)
10

```

run

ဒီ code လေးမှာ ဆိုရင်တော့ variable ၃ ခု ပါတာကို တွေ့ပါလိမ့်မယ်။

counter ကတော့ integer ပါ။ Integer ဆိုတာကတော့ ဒဿမ မပါတဲ့ ကိန်းပြည့် တန်ဖိုး တွေကို ဆိုတာပါ။

miles ကတော့ floating ပါ။ ဒဿမ တန်ဖိုး တွေပေါ့။

name ကတော့ String ပါ။ စာလုံး စာကြောင်းတွေ အတွက်ပါ။

boolean ကတော့ Boolean ပါ။ True နဲ့ False value ပဲ ရှိပါတယ်။ True , False တွေကို နောက်ပိုင်းမှာ condition တွေ နဲ့ တွဲသုံးတာကို တွေ့ ပါလိမ့်မယ်။

print ကတော့ value ကို ပြန်ပြီး ထုတ်ထားပေးတာပါ။

variable တွေကို နာမည်ပေးရမှာ သက်ဆိုင်ရာ နာမည်တွေ ပေးရပါတယ်။ x , y ,z ဆိုပြီး ပေးမည့် အစား ဒီ တန်ဖိုးကတော့ counter ဖြစ်ပါတယ်။ ဒီ တန်ဖိုးကတော့ miles ဖြစ်ပါတယ်။ ဒီ စာ ကတော့ name ဖြစ်ပါတယ် ဆိုပြီး variable name ကို ပေးလိုက်တဲ့ အတွက် ဖတ်လိုက်တာနဲ့ သဘောပေါက်လွယ်သွားပါတယ်။

Bit and Storage Data on Memory

Variable က memory ပေါ်မှာ နေရာ ယူပြီး ခဏ သိမ်းထားပါတယ်။ program ပိတ်လိုက်တဲ့ အခါမှာ memory ပေါ်ကနေလည်း ရှင်းလင်းလိုက်ပါတယ်။

Computer မှာ 0 နဲ့ 1 ပဲ ရှိပါတယ်။ 0 မဟုတ် ရင် 1 ပေါ့။ အဲဒါကို bit လို့ ခေါ်ပါတယ်။

8 Bit ကို 1 Byte လို့ ခေါ်ပါတယ်။ 8 Bit ဆိုတာကတော့ binary system အရ 00000000 ကနေ ပြီးတော့ 11111111 ထိ ရှိပါတယ်။

binary value 11111111 ကို decimal ပြောင်း 255 ရလာပါတယ်။

ဒါဟာ 8 Bit မှာ သိမ်းနိုင်တဲ့ အများဆုံး တန်ဖိုးပါ။

Integer ဟာ 32 Bit ရှိပါတယ်။ Integer တန်ဖိုးမှာ unsigned နဲ့ signed ဆိုပြီး ရှိပါတယ်။ Unsign ဟာ အပေါင်း ကိန်းတွေ ပဲ ဖြစ်တဲ့ အတွက်ကြောင့် 32 bit အပြည့် အသုံးပြုနိုင်ပါတယ်။ Signed ကတော့ +/- စတာပါလာတဲ့ အတွက်ကြောင့် 31 Bit ပဲ အသုံးပြုနိုင်ပါတယ်။ 1 bit ကိုတော့ အပေါင်း လား အနှုတ်လား ဆုံးဖြတ်ဖို့ အတွက်ပေးလိုက်ရပါတယ်။

Bit ပေါ်မှာ မူတည်ပြီး Integer တန်ဖိုးကို တွက်တဲ့ ပုံသေနည်း ရှိပါတယ်။

$(2 ^ {Total Bit}) - 1$
^ သည် power ဖြစ်သည်။

ဒါကြောင့် 8 Bit အတွက်ဆိုရင်တော့

$(2 ^ 8) - 1$
= 256 - 1
= 255

Sign Integer အမြင့်ဆုံး တန်ဖိုး တွက်ကြည့်ရင်တော့

31 Bit ကို အများဆုံးထားတယ်။

ဒါကြောင့်

$$\begin{aligned} & (2 ^ 31) - 1 \\ & = 2147483648 - 1 \\ & = 2,147,483,647 \end{aligned}$$

ဆိုပြီး ရလာပါတယ်။

ကျွန်တော်တို့ အနှုတ် ရဲ့ အနည်းဆုံး တန်ဖိုး မတွက် ခင် အောက်က ဇယားလေးကို တချက်ကြည့်လိုက်ပါ။

Binary Value	Two's complement interpretation	Unsigned interpretation
00000000	0	0
00000001	1	1
⋮	⋮	⋮
01111111	127	127
10000000	-128	128
10000001	-127	129
10000010	-126	130
⋮	⋮	⋮
11111110	-2	254
11111111	-1	255

ဒီ table မှာ ဘယ်ဘက် ဆုံးကတော့ Binary တန်ဖိုး နဲ့ အလယ်က Signed တန်ဖိုး ၊
နောက်ဆုံးကတော့ Unsigned တန်ဖိုးပါ။

အနှုတ် ဖြစ်ပြီဆိုတာနဲ့ ရှေ့ဆုံး binary ကို 1 ပြောင်းလိုက်ပါတယ်။ Sign မှာ 0 အတွက် က အပေါင်း
အနေနဲ့ တည်ရှိနေပေမယ့် အနှုတ် 0 ဆိုတာ မရှိပါဘူး။ ဒါကြောင့် ကျွန်တော်တို့ အနေနဲ့ အနှုတ်
တန်ဖိုး တစ်ခု ပို ပြီး သိမ်းလို့ရပါတယ်။

ဒါကြောင့် အောက်က equation နဲ့ Integer ရဲ့ Max range ကို တွက်လို့ ရပါတယ်။

$$-(2^{\text{[Total Bit]}}) \text{ to } (2^{\text{[Total Bit]}}) - 1$$

ဒါကြောင့် 32 Bit Integer Signed ကို တွက်ကြည့်ရင်တော့

$$\begin{aligned} &-(2^{31}) \text{ to } (2^{31}) - 1 \\ &= -2,147,483,648 \text{ to } 2,147,483,647 \end{aligned}$$

Unsigned ကို တွက်ရင်တော့ 32 Bit အပြည့် နဲ့ တွက်ရပါမယ်။

0 to $(2^{32}) - 1$
 = 0 to 4,294,967,295

ရ လာပါမယ်။

Float ကတော့ 32 Bit ရှိပါတယ်။

32 bit မှာ

sign 1 bit exponent 8 bit fraction 23 bit

binary value 0 01111100 011000000000000000000000

ကို ၃ ပိုင်း ခွဲထုတ်ပါမယ်။

Sign 0
 exponent 01111100
 fraction က 011000000000000000000000

sign 0 ဖြစ်တဲ့ အတွက်ကြောင့် +

$1 + \text{SUM} (i=1 \text{ To } 23) \text{ of } b(23-i) 2^{-i}$

ဒါကြောင့်

$1 + 2^{-2} + 2^{-3} = 1.375$

exponent က 01111100

$2^{(e - 127)} = 2^{124-127} = 2^{-3}$
 value = $1.375 \times 2^{-3} = 0.171875$

ဒါကြောင့် 0.171875 ကို binary 0 01111100 011000000000000000000000 အနေနဲ့ သိမ်းဆည်းပါတယ်။

Float ဟာ ဒဿမ ၇ နေရာထိ သိမ်းနိုင်ပါတယ်။

နောက်ထပ် ဒဿမ တန်ဖိုးကတော့ Double ပါ။

Double ကတော့ 64 Bit ရှိပါတယ်။ Double ကတော့ ဒဿမ 16 နေရာထိ သိမ်းနိုင်ပါတယ်။

String တန်ဖိုးကတော့ character storage ပေါ်မှာ မူတည်ပါတယ်။

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

အထိ နေရာ ယူပါတယ်။

ကျွန်တော်တို့ အနေနဲ့ storage တွေ အကြောင်း အနည်းငယ် သိထားခြင်းအားဖြင့် variable တွေ အများကြီး ဘာကြောင့် မသုံးသင့်တယ်ဆိုတာကို သဘောပေါက်စေဖို့ပါ။ memory အသုံးပြုပုံ အနည်းဆုံး ဖြစ်အောင် ဘယ်လို ရေးရမလဲ ဆိုတာကို စဉ်းစားစေနိုင်ဖို့ ရည်ရွယ်ပါတယ်။ တခြား အသေးစိတ်ကိုတော့ Computer Science ပိုင်းနဲ့ သက်ဆိုင်သွားပါပြီ။ ကျွန်တော့် အနေနဲ့ Programming Basic ပိုင်းမှာ တော့ ဒီလောက် ပဲ သင်ကြားပြီးတော့ programming နဲ့ သက်ဆိုင်ရာတွေကို ဆက်လက် ရေးသားသွားပါမယ်။

1.5 Operators

Operators ဆိုတာကတော့ ပေါင်းနှုတ်မြှောက်စား ပါ။ Programming မှာ

- အပေါင်း +
- အနှုတ် -
- အမြှောက် *
- အစား /
- အကြွင်း %

ဆိုပြီး သုံးပါတယ်။ ကျွန်တော်တို့ သင်္ချာ မှာ အသုံးပြုသည့် \times နှင့် \div အစားကို အသုံးမပြုပါဘူး။

အပေါင်း

အပေါင်း အတွက် ဥပမာ လေး အောက်မှာ ကြည့်ကြည့်ပါ။

```

1 k = 5 + 4
2 print(k)
3

```

run

ကိုး ၂ ခု ကို ပေါင်းထားပြီးတော့ ရလဒ် ကို k ထဲကို ထည့်ထားတာပါ။

programming မှာ data တွေကို ထည့်သွင်းမယ်ဆိုရင် ဘယ်ဘက်မှာ ရေးပါတယ်။

k = 5

အဲဒီ အဓိပ္ပာယ်ကတော့ k ထဲကို 5 ထည့်လိုက်လို့ ဆိုလိုတာပါ။

သင်္ချာမှာကတော့

$$5 + 1 = 6$$

ဆိုပြီး ရပါတယ်။ Programming မှာတော့

$$6 = 5 + 1$$

ဆိုပြီး ရေးရပါတယ်။ 6 က ရလဒ်ပါ။ ရလာတဲ့ အဖြေကို k ဆိုတဲ့ variable ထဲ အစား သွင်းဖို့ အတွက်

$$k = 5 + 1$$

ဆိုပြီး ရေးပါတယ်။ အဲဒါဆိုရင် k ထဲမှာ 6 ဝင်သွားပါပြီ။

```
1 a = 3
2 b = 4
3 c = a + b
4 print (c)
5
```

run

a ထဲကို 3 ထည့်။ b ထဲ ကို 4 ထည့်။ ပြီးလျှင် a နဲ့ b ကို ပေါင်း။ ရလာတဲ့ အဖြေကို c ထဲ ထည့်ပြီးတော့ ရလဒ် ပြန်ထုတ်ပြထားပါတယ်။

အနှုတ်

အပေါင်း အတိုင်းပါပဲ။ အနှုတ် အတွက် - ကို အသုံးပြုပါတယ်။

```
1 a = 10
2 b = 4
3 c = a - b
4 print (c)
5
```

run

အမြောက်

အမြောက်အတွက် * ကို အသုံးပြုပါတယ်။

```
1 a = 3
2 b = 4
3 c = a * b
4 print (c)
5
```

run

အစား

အစား အတွက် / ကို အသုံးပြုပါတယ်။

```
1 a = 10
2 b = 2
3 c = a / b
4 print (c)
5
```

run

အကြွင်း

အကြွင်းကို % ကို အသုံးပြုပါတယ်။

```
1 a = 13
2 b = 8
3 c = a % b
4 print (c)
5
```

run

1.6 Problem Solving

Programming ကို ရေးသားရာမှာ သင်္ချာကဲ့သို့ပင် ပြဿနာတွေ ကို ဖြေရှင်း ရတာတွေ ပါဝင်ပါတယ်။ အသုံးပြုသူတွေ ဖြစ်နေတဲ့ ပြဿနာတွေကို လွယ်လင့်တကူ ဖြေရှင်းပေးဖို့ program တွေကို စဉ်းစား တွေးခေါ် ရေးရပါတယ်။

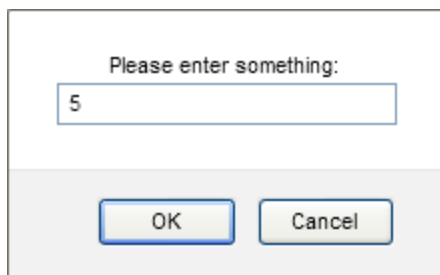
ဥပမာ။ ကိန်း ၂ လုံးကို လက်ခံပါ။ ပြီးရင် ၂ ခု ပေါင်းလဒ်ကို ထုတ်ပြပါ။

လွယ်လွယ်လေးပါ။ ကျွန်တော် တို့ အနေနဲ့ ကိန်း ၂ လုံး လက်ခံမယ်။ ပြီးရင် ပေါင်း ပြီး ရတဲ့ အဖြေကို ထုတ်ပေးလိုက်ရုံပါပဲ။

အသုံးပြုသူကို input ထည့်ပေးဖို့ အတွက် python3 မှာတော့ input ကို အသုံးပြုပါတယ်။

```
1 user_input = input("Please enter something: ")
2 print ("you entered", user_input)
3
```

run



```
you entered 5
```

ကျွန်တော်တို့ user input လက်ခံ တတ်ပြီ ဆိုရင် ကိန်း ၂ လုံး လက်ခံရအောင်။ ပြီးတော့ ပေါင်းပြီးတော့ ရလဒ်ကို ထုတ်ပေးရုံပါပဲ။

```
1 input1 = int(input("Please enter first number: "))
2 input2 = int(input("Please enter second number: "))
3 result = input1 + input2
4
5 print (input1,"+",input2,"=", result)
6
```

run

Please enter first number:

OK Cancel

Please enter second number:

☐ Prevent this page from creating additional dialogs

OK Cancel

5 + 8 = 13

ကျွန်တော်တို့ user ဆီကနေ data ကို လက်ခံတဲ့ အခါ string value အနေနဲ့ ရလာပါတယ်။ integer အနေနဲ့ လိုချင်တဲ့ အတွက်ကြောင့် int() ကို အသုံးပြုထားပါတယ်။

```
input1 = int(input("Please enter first number: "))
```

input ကနေ user အနေနဲ့ နံပါတ်ကို ရိုက်ထည့်ပေးလိုက်ပေမယ့် string အနေနဲ့ ဝင်လာပါတယ်။ int() နဲ့ ပြောင်းလိုက်တဲ့ အတွက်ကြောင့် နံပါတ်ရပါတယ်။

```

1 a = "5"
2 b = "6"
3 print(a+b)
4

```

run

56

string ၂ ခု ကို ပေါင်းသည့် အခါမှာ 11 အစား 56 ဖြစ်သွားတာကို တွေ့ရမှာပါ။

String နံပါတ်ကို int ပြောင်းချင်တာကြောင့် int() ကို အသုံးပြုရပါတယ်။

```

1 a = "5"
2 b = "6"
3 print(int(a)+int(b))
4

```

run

11

အခု ဟာ ဥပမာ အသေးလေး တစ်ခုပါ။

နောက်ပြီး စဉ်းစား ရမှာ က အသုံးပြုသူက ဂဏန်းတွေ မထည့်ပဲ စာတွေလည်း ရိုက်ထည့် နိုင်တယ်။ ဂဏန်းတွေ မဟုတ်ရင် ဂဏန်းသာ ထည့်ပါဆိုပြီး message ပြဖို့ လိုလာတယ်။ ဒီလိုမျိုး ဖြစ်နိုင်ခြေ ရှိတာတွေကို programming ရေးတဲ့ အခါ ထည့်စဉ်းစားရပါတယ်။

အဲဒီလိုမျိုး စစ်ဖို့ အတွက် နောက် အခန်းမှာမှ looping တွေ condition တွေ အကြောင်း ရေးသွားပါမယ်။

1.7 Installing Python 3

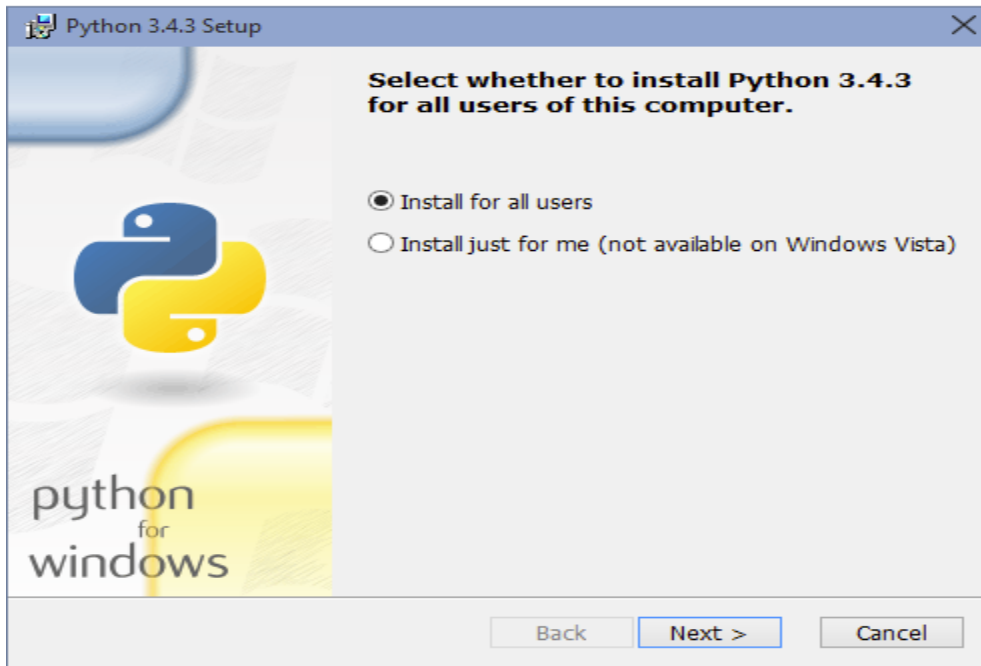
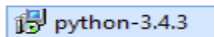
ကျွန်တော် ဒီ စာအုပ်မှာ သင်ကြားမှာက programming အကြောင်းပါ။ Python programming language ကို သင်ကြားတာ မဟုတ်ပါဘူး ။ Programming language တစ်ခု နဲ့ တစ်ခုက အများအားဖြင့် စဉ်းစားတွေးတောရသည့် အခြေခံက အတူတူပါပဲ။ ဒါကြောင့် တစ်ခုကို တတ်မြောက်ထားရင် နောက်ထပ် တစ်ခုကိုလည်း လွယ်လင့်တကူ လေ့လာနိုင်ပါတယ်။

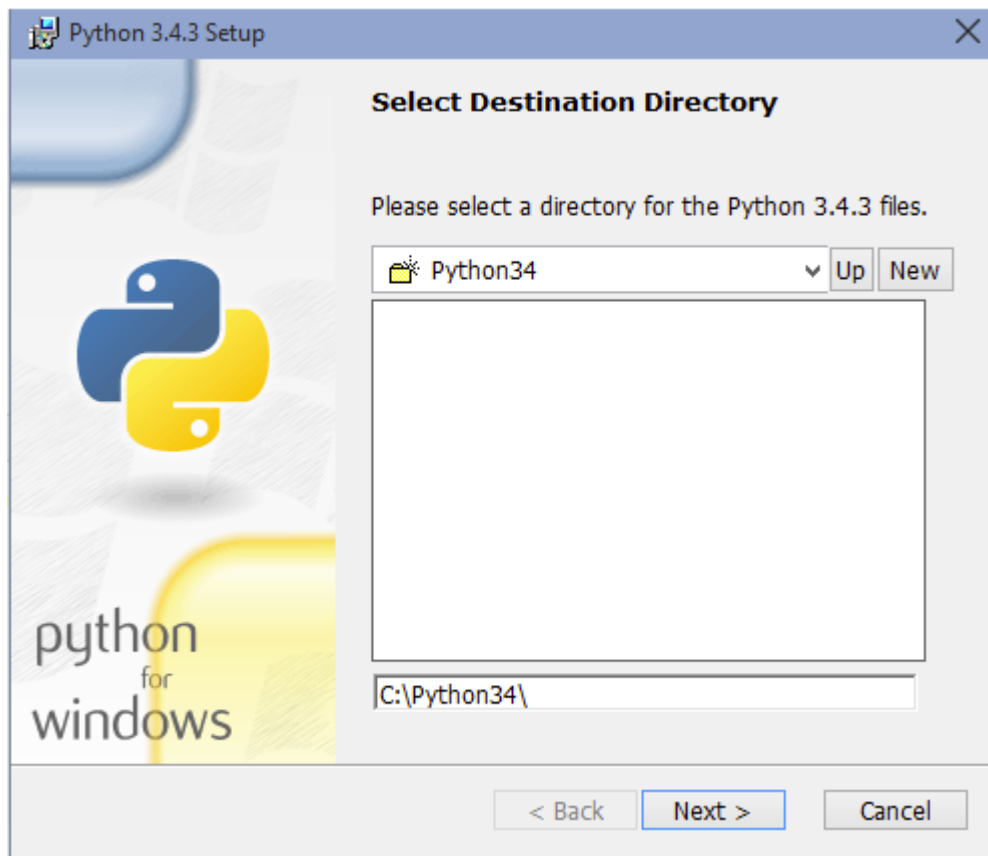
Download

Python ကို <https://www.python.org/downloads/> မှာ download ချယူနိုင်ပါတယ်။ Python 3 သို့မဟုတ် နောက် အသစ် version ကို download ချပါ။ လက်ရှိ စာအုပ် က code တွေ ကို python 3 နဲ့ ရေးသားထားသောကြောင့် ဖြစ်ပါတယ်။

Windows

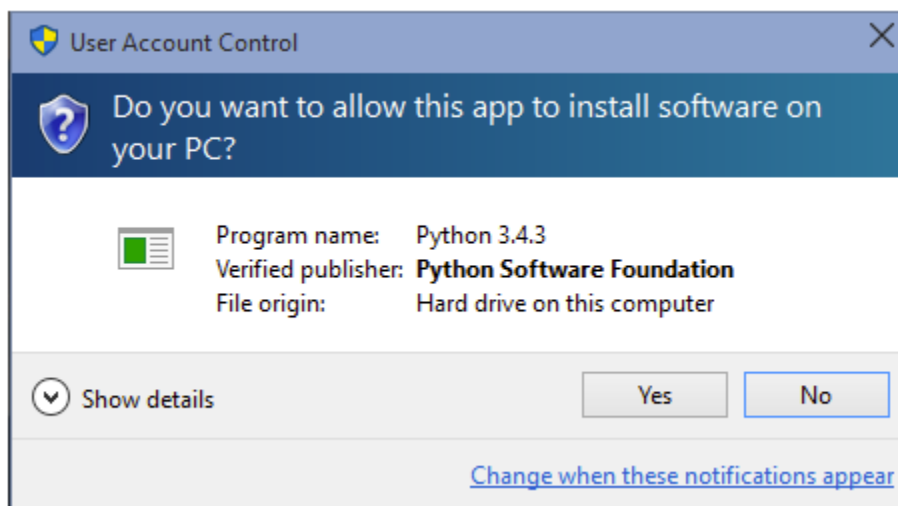
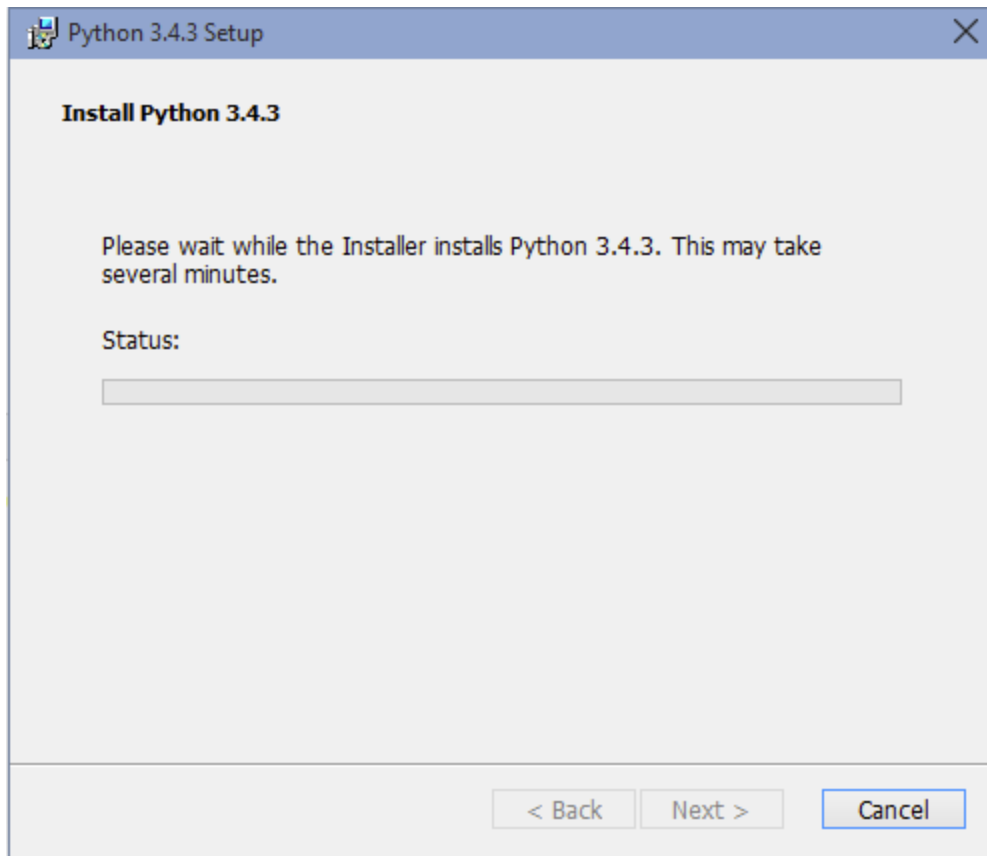
Python ကို download ချပြီးတော့ ရလာတဲ့ installer ကို double click ပြီး install သွင်းပါ။ ပြီးလျှင် Next , Next သာ လုပ်သွားပါ။



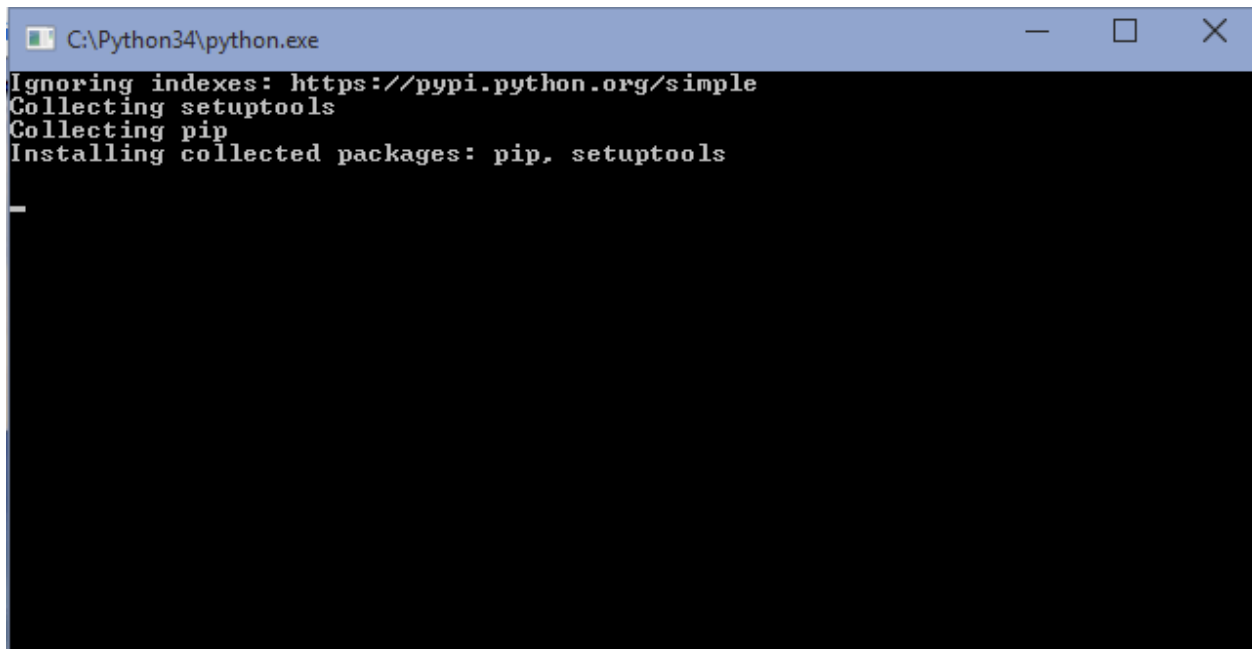


C:\Python34 မှာ python ကို သွင်းထားတယ်ဆိုတာကို မှတ်ထားဖို့ လိုပါတယ်။





ဒီလို dialog တက်လာရင် Yes သာ နှိပ်လိုက်ပါ။



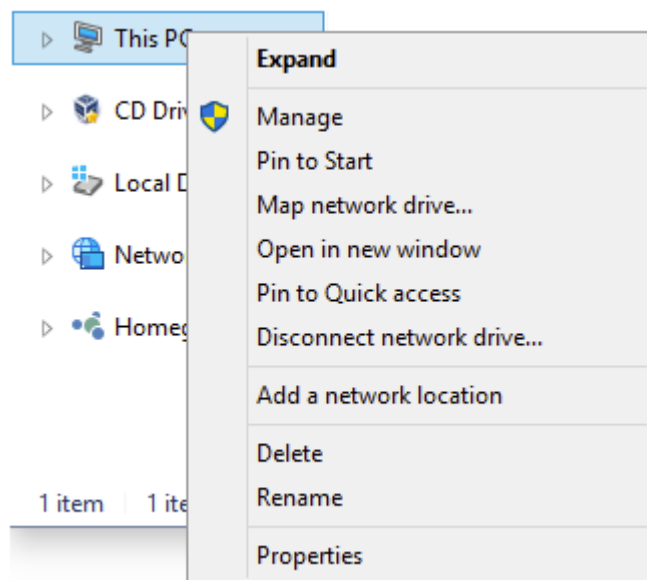
```

C:\Python34\python.exe
Ignoring indexes: https://pypi.python.org/simple
Collecting setuptools
Collecting pip
Installing collected packages: pip, setuptools

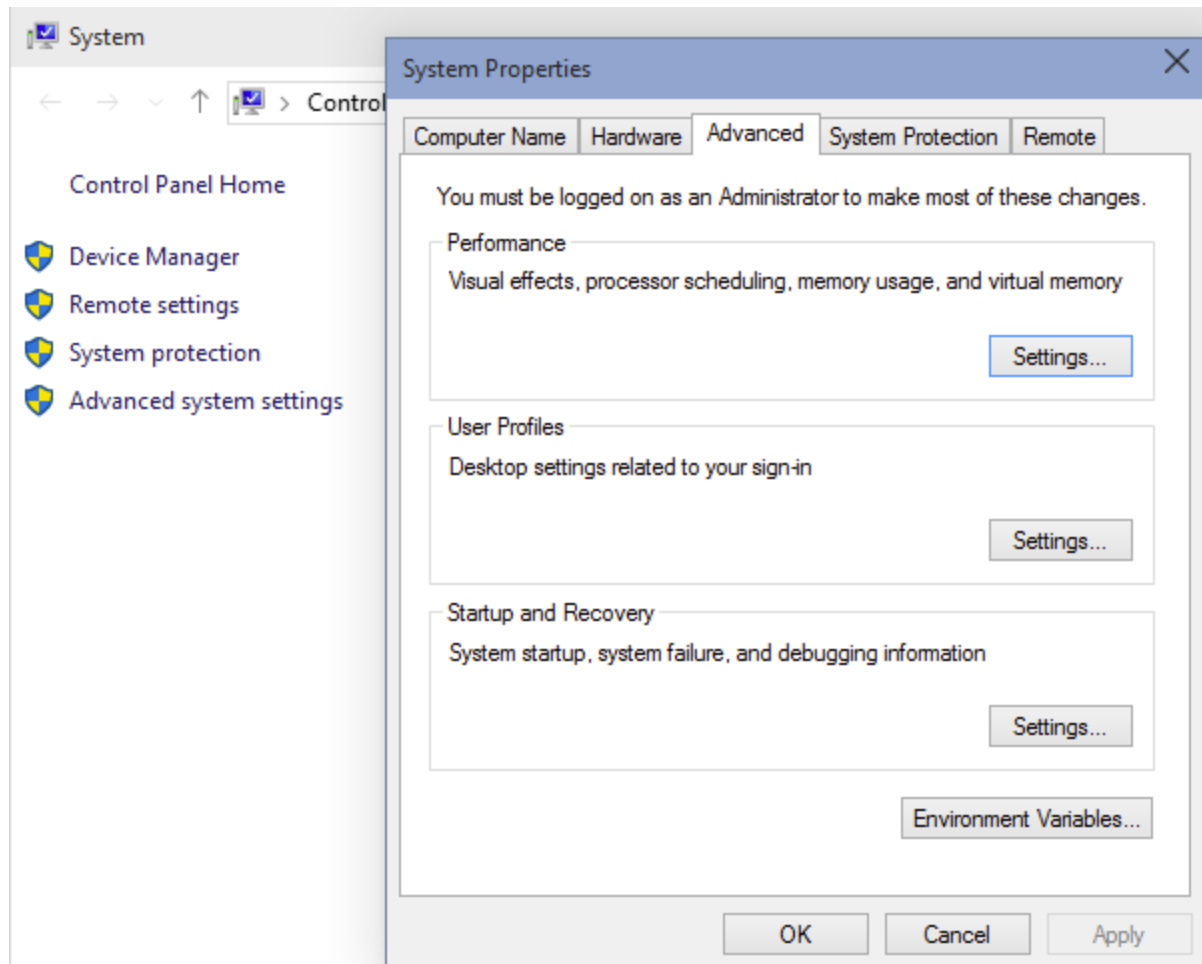
```

အခု Python ကို Install ပြီးပါပြီ။ သို့ပေမယ့် command prompt မှာ Python မရသေးပါဘူး။

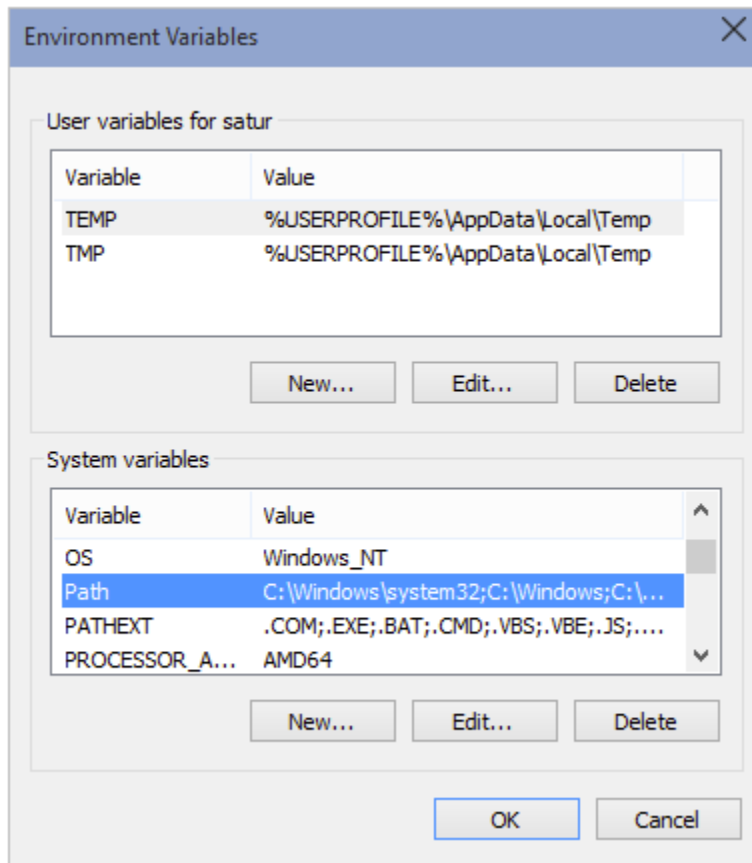
My Computer ကို Right Click နှိပ်။



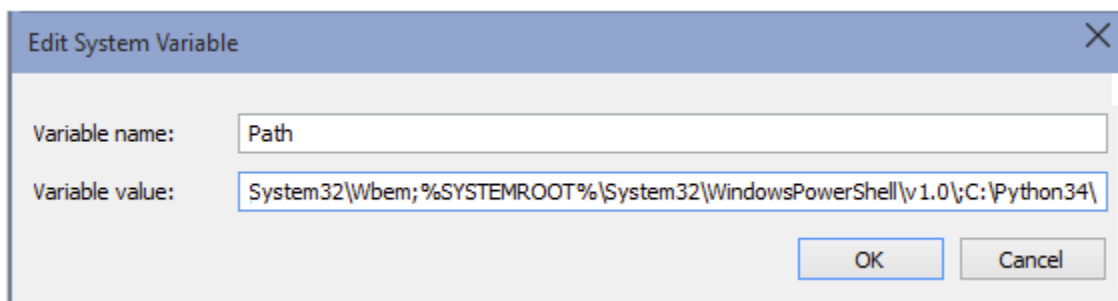
Properties ကို နှိပ်။



Environment Variables... ကို ထပ်နှိပ်ပါ။



Path ကို select လုပ်ပြီးတော့ Edit လုပ်ပါ။



နောက်ဆုံး မှာ ;C:\Python34 ကို ထည့်ပေးပြီး OK လုပ်ပါ။ semi comman (;) ပါဖို့ မမေ့ပါနဲ့။ OK လုပ်ပြီး dialog တွေ အကုန် ပြန်ပိတ်လိုက်ပါ။

```

C:\> Command Prompt - python
Microsoft Windows [Version 10.0.10240.17134]
Copyright (c) 2015 Microsoft Corporation
C:\Users\satur>python
Python 3.4.3 (v3.4.3:9b73f1c526a0006434eb666579e0036fb2767943, Apr 4 2015)
Type "help", "copyright", "credits()" and "quit()" to get more help
>>>

```

command prompt မှာ python ကို ရိုက်လိုက်ပါ။ ပုံထဲက အတိုင်း မြင်ရရင် python ကို စတင် အသုံးပြုနိုင်ပါပြီ။

Linux

Ubuntu , Debian စတဲ့ Linux တွေမှာ Python 3 က default အနေနဲ့ သွင်းထားပြီးသားဖြစ်ပါတယ်။ Terminal မှာ python3 -V လို့ ရိုက်ကြည့်ပါ။ Python 3.4 သို့မဟုတ် နောက်ထပ် version အသစ်ဖြစ်တာကို တွေ့ရပါမယ်။

Mac

.pkg file ကို သွင်းပြီးသွားပါက terminal မှာ python3 -V လို့ ရိုက်ကြည့်ပါ။ Python 3.4 ဒါမှမဟုတ် နောက်ထပ် version အသစ်ကို တွေ့ရပါမယ်။

Testing Python

command prompt (Windows) သို့မဟုတ် Terminal (Mac , Linux) ကို ဖွင့်ပြီး python3 , (Windows တွင် python) ရိုက်လိုက်ပါ။

Python version number နဲ့ python ရိုက်ဖို့ နေရာ တက်လာပါမယ်။

```

Python 3.4.2 (v3.4.2:ab2c023a9432,
[GCC 4.2.1 (Apple Inc. build 5666)
Type "help", "copyright", "credits"
>>> █

```

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

ရိုက်လိုက်ပါ။ ပြီးရင် Enter ခေါက်ရင် Hello World ဆိုတာ ထုတ်ပြတာကို မြင်ရပါမယ်။


```

type help , copyright ,
>>> print("Hello World")
Hello World
>>> █

```

ပြန်ထွက်ဖို့ အတွက်

```
exit()
```

ရိုက်ပြီး enter ခေါက်လိုက်ပါ။

helloworld.py ကို ဖန်တီးပါ။

အထဲမှာ

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

ရိုက်ပြီး save လုပ်ထားပါ။

ပြီးရင် file ကို terminal ကနေ

Linux, Mac

```
python3 helloworld.py
```

Windows

```
python helloworld.py
```

လို့ ခေါ်ကြည့်ပါ။

Hello World ထွက်လာရင် ရပါပြီ။

ဒါဆိုရင် ကျွန်တော်တို့ စက်ထဲမှာ python သွင်းပြီးပါပြီ။

Chapter 2 :: Programming

ဒီအခန်းမှာတော့ Programming နဲ့ သက်ဆိုင်ရာ စဉ်းစားတွေးခေါ်ပုံတွေ ရေးပုံတွေ ကို ရေးသားသွားမှာပါ။ နောက်ပြီး ကိုယ်တိုင် စဉ်းစားပြီး ရေးရမယ့် အပိုင်းတွေ ပါပါတယ်။ Programming က သင်္ချာလို လက်တွေ့ လေ့ကျင့် စဉ်းစား ရပါတယ်။ စာဖတ်ရုံနဲ့ မရပါဘူး။ ကျွန်တော် ဒီ အခန်းမှာ Pseudo code အကြောင်း ရေးထားပေးပြီးတော့ နောက်ပိုင်းမှာ Pseudo code က ကိုယ်ပိုင် program ကို python3 နဲ့ ပြန်ပြီး ရေးကြည့်ဖို့ လေ့ကျင့်ခန်းတွေ ပါဝင်ပါမယ်။

2.1 Pseudo

Pseudo code ဆိုတာကတော့ အတုအယောင် code ပေါ့။ programming မှာ language အမျိုးမျိုး ရှိပြီးတော့ language တစ်ခု နဲ့ တစ်ခုမှာ ပါဝင်တဲ့ function တွေ မတူပါဘူး။ ဒါကြောင့် ကျွန်တော်တို့တွေဟာ Pseudo code ကို အသုံးပြုပြီးတော့ တစ်ယောက် နဲ့ တစ်ယောက် နားလည်အောင် ရေးသားပေးကြပါတယ်။ Pseudo code ဆိုတာက ဘယ်သူ့ မဆို နားလည်အောင် ရေးသားထားတဲ့ language တစ်မျိုး ဖြစ်တဲ့ အတွက် ဘယ်လိုမျိုး ရေးရမယ် ဆိုတာကို အတိအကျ သတ်မှတ်ပြီး ပြောလို့ မရပါဘူး။ တချို့ကလည်း C++ style အသုံးပြုသလို တချို့ကလည်း javascript style အသုံးပြုပါတယ်။ ဒါပေမယ့် pseudo code က တကယ် run ကြည့်လို့ မရဘူး။ အသုံးပြုလို့ မရဘူး။ pseudo code ကို ပြန်ကြည့်ပြီးတော့ မိမိ နှစ်သက်ရာ language နဲ့ ပြန်ရေးပြီး run မှ သာ ရပါလိမ့်မယ်။

pseudo code ဥပမာ လေးကို ကြည့်ရအောင်

If student's grade is greater than or equal to 40

Print "passed"

else

Print "failed"

ဒီ code လေးမှာ ဆိုရင် ရေးထားတာက english လိုပါပဲ။ ကျောင်းသား ရဲ့ အမှတ်က ၄၀ ကျော်ရင် အောင် တယ်လို့ ပြမယ်။ မဟုတ်ခဲ့ရင် ကျတယ်လို့ ပြောမယ်။ ရှင်းရှင်းလေးပါပဲ။

Pseudo code မှာ ကျွန်တော်တို့

- **SEQUENCE** လုပ်မယ့် အလုပ် အဆင့်ဆင့် ကို ရေးသားခြင်း
- **WHILE** ကတော့ loop အတွက်ပါ။ ထပ်ခါ ထပ်ခါ အကြိမ်ကြိမ် လုပ်ဖို့ အတွက်ပါ။ ဘယ်အထိ လုပ်ဖို့ ဆိုတာကို စစ်ဆေးထားပြီး စစ်ဆေးတဲ့ အဆင့် မဟုတ်တော့ဘူးဆိုမှသာ looping ထဲက ထွက်ပါလိမ့်မယ်။ ဥပမာ။ ထပ်ခါ ထပ်ခါ လုပ်မယ်။ စုစုပေါင်း ရမှတ် ၁၀၀ မပြည့်မချင်း လုပ်မယ် ဆိုတာ မျိုးပေါ့။
- **IF-THEN-ELSE** စစ်ဆေးပြီးတော့ ဖြစ်ခဲ့ရင် ဒါလုပ် မဖြစ်ခဲ့ရင်တော့ ဒါကို လုပ်ပါ ဆိုတဲ့ condition တွေ အတွက်ပါ။
- **CASE** ကတော့ condition အတွဲလိုက် စစ်ဖို့ပါ။ 1 ဖြစ်ခဲ့ရင် ဒါလုပ်။ 2 ဖြစ်ခဲ့ရင် ဒါလုပ်။ ၃ ဖြစ်ခဲ့ရင် ဒါလုပ် စတာတွေ အတွက်ပါ။

- **FOR** ကတော့ **while** နဲ့ အတူတူပါပဲ။ သို့ပေမယ့် **FOR** ကတော့ ဘယ်ကနေ ဘယ်အတွင်း ဆိုတာ ရှိပါတယ်။ ဥပမာ ။ ထပ်ခါ ထပ်ခါ လုပ်မယ်။ ဒါပေမယ့် ၁ ကနေ ၅ အတွင်း လုပ်မယ် ဆိုတာ မျိုးပေါ့။

SEQUENCE

Programming ဆိုတာက sequential ဆိုတာကို ကျွန်တော် အခန်း ၁ မှာ ပြောခဲ့ပါတယ်။ တစ်ခုပြီးမှ တစ်ခုလုပ်မယ်။ ဒါကြောင့် Pseudo code က programming အတွက် ဖြစ်တဲ့ အတွက်ကြောင့် တစ်ဆင့်ပြီး တစ်ဆင့် သွားရပါတယ်။

ဥပမာ

READ height of rectangle

READ width of rectangle

COMPUTE area as height times width

ဒီ code လေးကို ကြည့်လိုက်တာနဲ့ ဒါဟာ area တွက်ထားတဲ့ code လေး ဆိုတာ နားလည် သွားတယ်။ height ကို လက်ခံမယ်။ width ကို လက်ခံမယ်။ ပြီးရင် height နဲ့ width ကို မြှောက်ပြီးတော့ area ရလာမယ်။

ဒါကို python နဲ့ ပြန်ရေးကြည့်ရအောင်။


```

1 height = input("Enter Height Of Rectangle: ")
2 width = input("Enter Width Of Rectangle: ")
3 area = int(height) * int(width)
4 print("Area is ",area)
5

```

run

Area is 30



ကျွန်တော် ရေးထားတဲ့ python code ဟာ programming မတတ်တဲ့ သူ တစ်ယောက်အတွက် ဖတ်လိုက်ရင် နားလည်ဖို့ ခက်ခဲတယ်။ Pseudo code ကတော့ ဘယ်သူ မဆို နားလည်နိုင်အောင် ရေးသားထားပါတယ်။

Input, output, processing တွေ အတွက် အောက်ပါ keyword တွေကို ကျွန်တော်တို့ အသုံးပြုပါတယ်။

- **Input:** READ, OBTAIN, GET
- **Output:** PRINT, DISPLAY, SHOW
- **Compute:** COMPUTE, CALCULATE, DETERMINE
- **Initialize:** SET, INIT
- **Add one:** INCREMENT, INCREASE, DECREMENT , DECREASE

စတာတွေကို အသုံးပြုနိုင်ပါတယ်။

2.2 Flow Chart

Programming ကို လေ့လာရာမှာ အခြေခံ အနေနဲ့ Pseudo code အပြင် Flow chart ကို ပါ သိထားသင့်တယ်။ အခုအချိန်ထိ coding အကြောင်းကို ကျွန်တော် မရေးသေးပါဘူး။ အခြေခံ အဆင့်တွေ ဖြစ်တဲ့ Flow Chart , Pseudo စတာတွေ ကို နားလည် သွားတဲ့ အခါမှာ programming ကို လွယ်လင့် တကူ စဉ်းစားနိုင်အောင် အထောက် အကူပြုနိုင်ပါတယ်။

Flow Chart ဆိုတာကိုတော့ ကျွန်တော်တို့ တွေ ဘာပြီး ရင် ဘာလုပ်မယ် ဆိုတာကို အဆင့်ဆင့် ပုံတွေနဲ့ ဆွဲပြထားပါတယ်။ Flow Chart ဆွဲရာမှာ သက်ဆိုင်ရာ သတ်မှတ် ချက်တွေ ရှိပါတယ်။ အရင်ဆုံး ဘယ်ပုံတွေက ဘာကို ကိုယ်စားပြုတယ်ဆိုတာကို အောက်မှာ ဖော်ပြထားပါတယ်။

Terminal

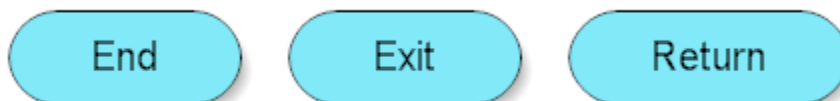


Flowchart အစ သို့မဟုတ် အဆုံး စသည့် နေရာတွေ မှာ အသုံးပြုပါတယ်။

အစကို **Start** , **Begin** စသည်ဖြင့် အသုံးပြုပါတယ်။

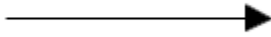


အဆုံးကို တော့ **End** , **Exit**, **Return** တွေကို အသုံးပြုပါတယ်။

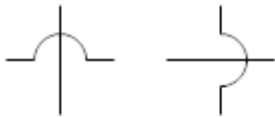


Lines with Arrows

တစ်ခုကနေ နောက်တစ်ခုကို သွားဖို့ ညွှန်ပြထားတာပါ။ ဒါအဆင့် ပြီးရင် ဘယ်ကို သွားမလဲ ဆိုတာကို ညွှန်ပြထားပါတယ်။



Line တွေ ဆွဲတဲ့ အခါမှာ cross ဖြစ်နေရင် မျဉ်းကို မဖြတ်သွားပဲ အခုလို ဂဏ် ပုံလေးနဲ့ ဆွဲပါတယ်။



Rectangle



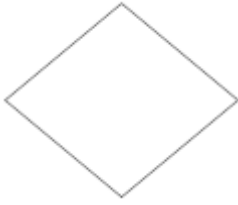
Flowchart မှာ စတုရန်းကို process, task, action, operation စသည့်အတွက် အသုံးပြုပါတယ်။ စတုရန်းဟာ action တစ်ခုခုလုပ်ဖို့ တစ်ခုခုပြီးမြောက်ဖို့အတွက် ညွှန်ပြထားပါတယ်။

Send the order

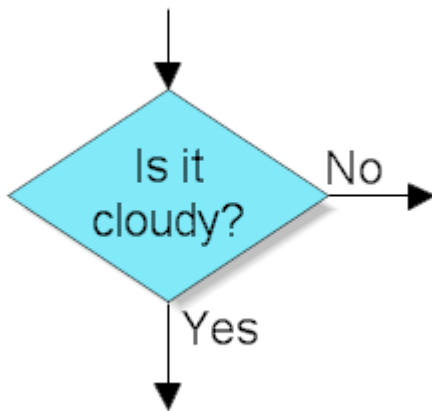
Check the address

Install the
muffler

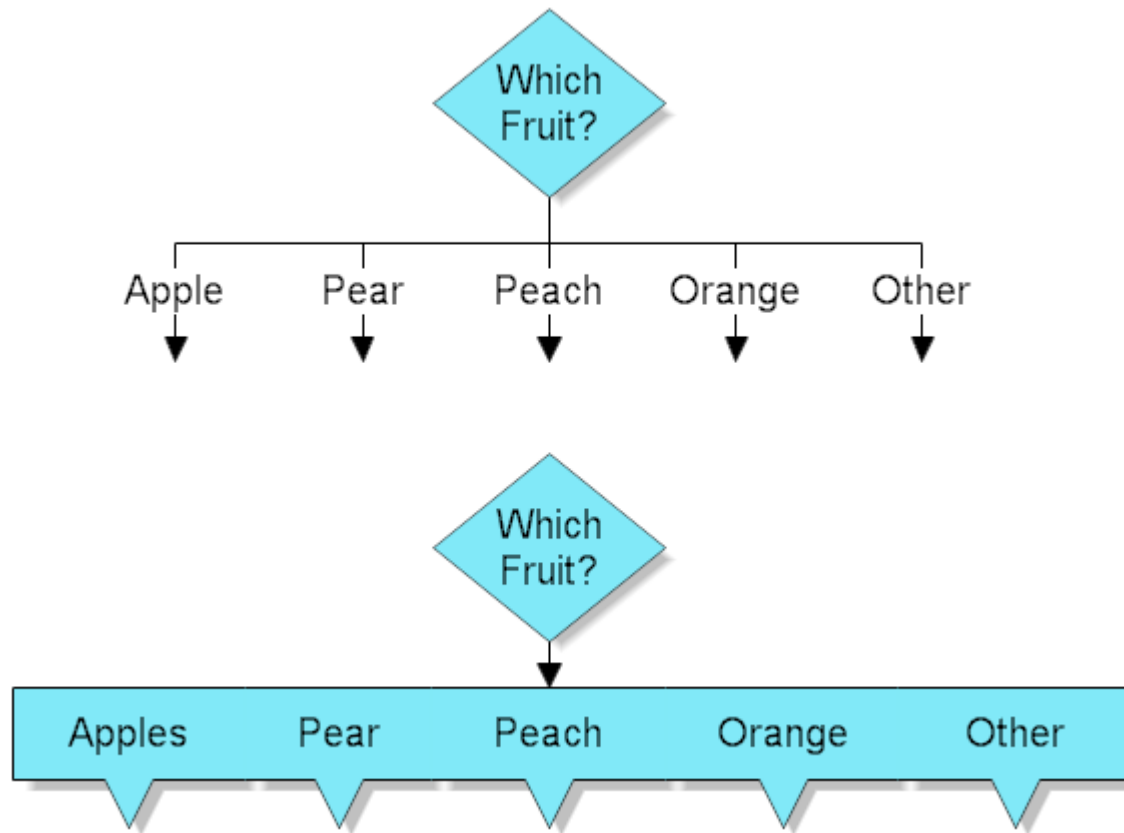
Decision



အသုံးပြုသူကို မေးခွန်းမေးပြီးတော့ အဖြေပေါ်မှာ မူတည်ပြီး အလုပ်လုပ်စေချင်တဲ့အခါမှာ decision ကို အသုံးပြုရပါတယ်။ input တစ်ခု ဝင်ပြီးတော့ output မှာ YES,NO ဖြစ်ပါတယ်။ ဒါဖြစ်ခဲ့ရင် ဒါလုပ်။ မဟုတ်ခဲ့ရင် ဘာလုပ် စသည်အတွက် အသုံးပြုနိုင်ပါတယ်။



တစ်ခါတစ်လေ တစ်ခုထက် မက ဖြစ်နိုင်တဲ့ အဖြေတွေ အတွက်လည်း အသုံးပြုပါတယ်။



Circle



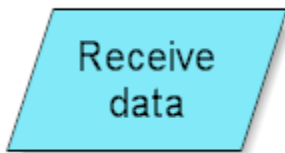
Flow chat က အရမ်းရှည်သွားရင် သီးသန့် ဆွဲဖို့အတွက် circle ကို အသုံးပြုပါတယ်။ flow chart တစ်ခုနဲ့ တစ်ခုကို connect လုပ်ထားတယ်ဆိုတာကို ဖော်ပြထားသည့် သဘောပါ။ Circle အတွင်းမှာ နာမည်ပါဝင်ပါတယ်။



Input/Output



User ဆီကနေ Data ကို လက်ခံတော့မယ်ဆိုရင်တော့ အနားပြိုင် စတုရံ ကို အသုံးပြုပါတယ်။



ပုံမှန် အခြေခံ အားဖြင့် Flow chart အတွက် ဒါလေးတွေ သိထားရင် လုံလောက်ပါတယ်။
ကျွန်တော်တို့ Example တွေ နဲ့ တချက်ကြည့်ရအောင်။

ref: http://www.rff.com/flowchart_shapes.htm

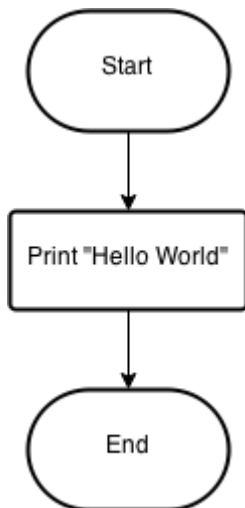
2.3 Hello World

Programming ကို စလိုက်တိုင်း ပထမဆုံး ရေးသားကြတဲ့ code ကတော့ Hello World ပါပဲ ။
Screen ပေါ်မှာ ဘယ်လိုပေါ်အောင် ဖန်တီးရမယ် ဆိုတာရဲ့ အစပါပဲ။

Pseudo code နဲ့ ဆိုရင်တော့

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

တကယ်လို့ flow chart နဲ့ ဆိုရင်တော့



Python နဲ့ ဆိုရင်တော့

```
1 print("Hello World")
2
```

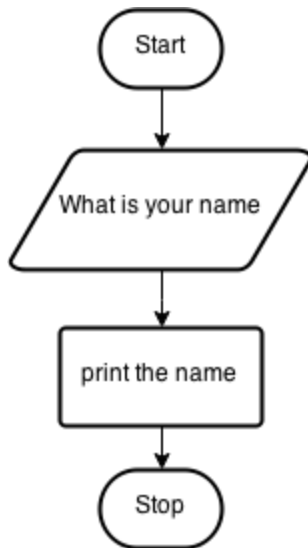
run

Hello World

2.4 What is your name

အခု ကျွန်တော်တို့ user ကို နာမည် ဘယ်လို ခေါ်လဲ မေးမယ်။ ပြီးရင် user ဆီကနေ ပြီးတော့ လက်ခံမယ်။ နာမည်ကို ရိုက်ထည့်ပြီးတာနဲ့ Your name is ဆိုပြီး နာမည်ကို ထုတ်ပြမယ်။

အရင်ဆုံး flow chart ကို ကြည့်ပါ။



ပြီးရင် flow chart အတိုင်း code ကို ရေးထားပါတယ်။

```
username = input("What is your name ? : ")
print("Your name is ",username)
```

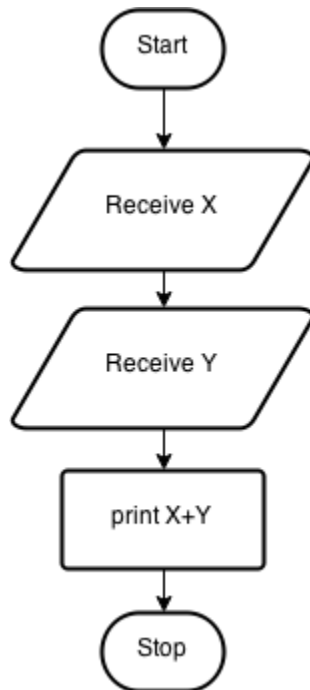
The screenshot shows a Python IDE with the following code:

```
1 username = input("What is your name ? : ")
2 print("Your name is ",username)
3
```

A 'run' button is visible. Below the code, the output is displayed: 'Your name is Saturngod'. A dialog box is also shown, titled 'What is your name ? :', with a text input field containing 'Saturngod' and 'OK' and 'Cancel' buttons.

2.5 Sum

အခု ကျွန်တော်တို့တွေ user ဆီကနေ ကိန်း ၂ ခု ကို လက်ခံမယ်။ ပြီးရင် ပေါင်းမယ်။ ပေါင်းပြီး ရလာတဲ့ အဖြေကို ပြပေးမယ်။



Flow chat ထဲမှာ ပြထားသလို လွယ်လွယ်လေးပါပဲ။

```
x = input("Enter first value : ")
y = input("Enter second value : ")
print("X + Y = ", x + y)
```

```
1 x = input("Enter first value : ")
2 y = input("Enter second value : ")
3 print("X + Y = ",x + y)
4
```

run

X + Y = 1020

Enter first value :

OK Cancel

Enter second value :

☐ Prevent this page from creating additional dialogs

OK Cancel

ဒီ code မှာ ဆိုရင် ကိန်း ၂ လုံးကို ပေါင်းပေးမယ့် စာလုံး ၂ လုံး ပေါင်းတာဖြစ်နေတာ ကို တွေ့ရပါလိမ့်မယ်။ ဥပမာ ၅ နဲ့ ၆ ထည့်လိုက်ရင် ၅၆ ထွက်လာပါလိမ့်မယ်။

ကျွန်တော်တို့တွေ အနေနဲ့ နံပါတ်ကို လက်ခံမယ်ဆိုတဲ့ အတွက်ကြောင့် အောက်ကလို ပြောင်းရေးပါတယ်။

```
x = input("Enter first value : ")
y = input("Enter second value : ")

try :
    x = int(x)
    y = int(y)

    print("X + Y = ",x + y)

except ValueError:
    print("Please enter number only")
```

```

1 x = input("Enter first value : ")
2 y = input("Enter second value : ")
3
4 try :
5     x = int(x)
6     y = int(y)
7
8     print("X + Y = ", x + y)
9
10 except ValueError:
11     print("Please enter number only")
12

```

run

X + Y = 30

Enter first value :

10

OK Cancel

Enter second value :

20

☐ Prevent this page from creating additional dialogs

OK Cancel

အခု code မှာ `try` , `except` ဆိုတာကို တွေ့ပါလိမ့်မယ်။ `try` ထဲမှာ Error တစ်ခုခု ဖြစ်တာနဲ့ `except` ကို ရောက်လာပါမယ်။ အခု code ထဲမှာ `int` ပြောင်းထားတဲ့ နေရာမှာပဲ ပြဿနာ ဖြစ်နိုင်ပါတယ်။ ဒါကြောင့် `except ValueError:` ကို အသုံးပြုထားတာပါ။

2.6 Condition

ဒါမှမဟုတ်ခွဲရင် ဒါလုပ်မယ် စသည်ဖြင့် အခြေအနေကို စစ်သည့်အရာတွေ အတွက် ကျွန်တော်တို့တွေ if , switch စတဲ့ syntax ကို အသုံးပြုရပါတယ်။

သို့ပေမယ့် Python မှာ switch ကို support မလုပ်ပါဘူး။

အခု ကျွန်တော်တို့ သည် ဆီက ဂဏန်း ၂ ခု ကို လက်ခံမယ်။ ကိန်း ၂ ခု ကို စားမယ်။ ဒုတိယ ကိန်း က သည်ဖြစ်နေရင် အသုံးပြုသူကို သည် ထည့်လို့ မရဘူးဆိုရင် error message ပြမယ်။

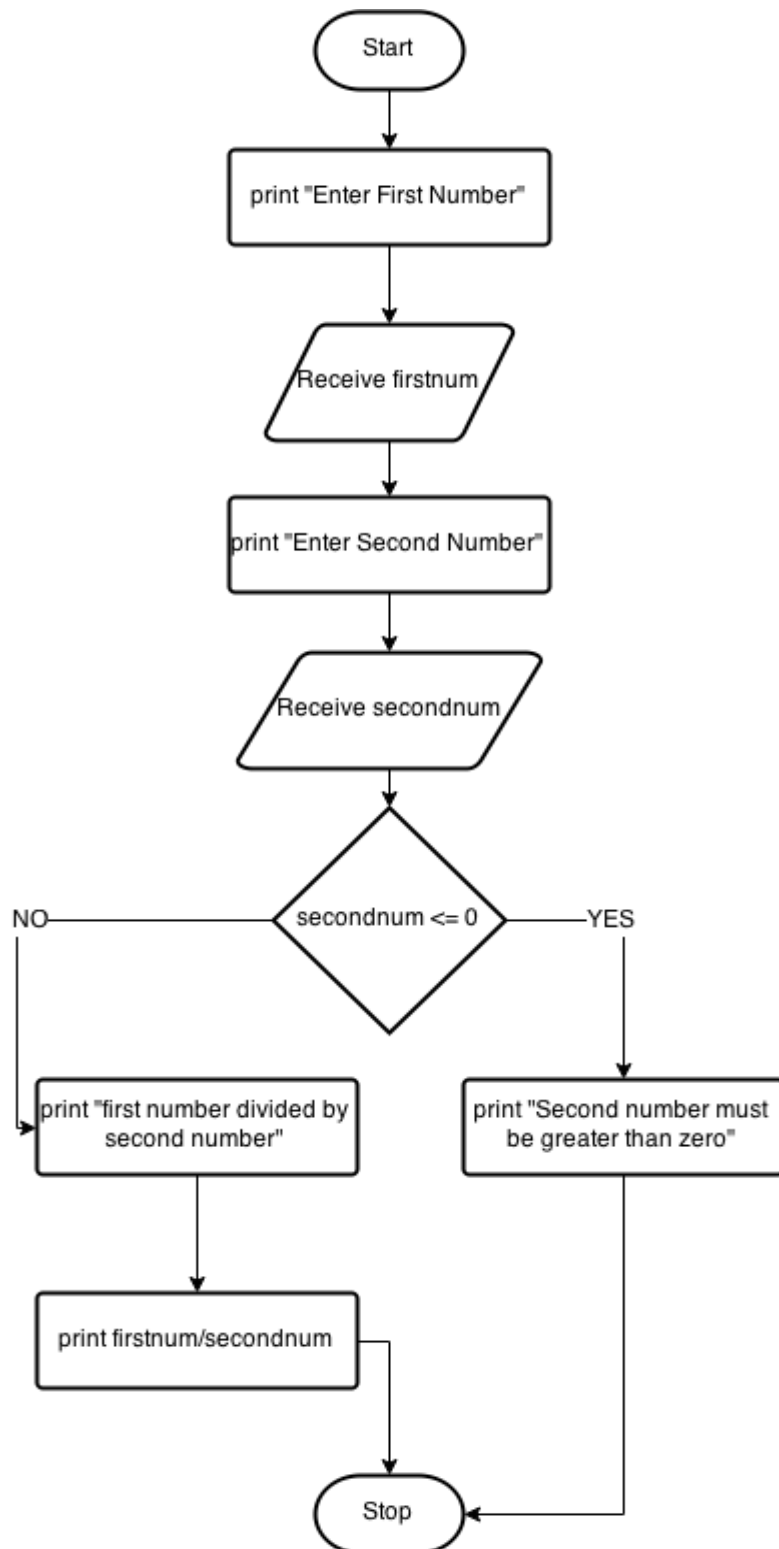
Pseudo code အရ ဆိုရင်တော့

```
Print "Enter First Number"
READ firstnum

Print "Enter Second Number"
READ secondnum

if secondnum is less than or equal zero
    Print "Second number must be greater than zero"
else
    Print firstnum + "divied by " + secondnum
    Print firstnum/secondnum
```


flowchart ကို ကြည့်ရအောင်။



Python ကို အောက်မှာ ကြည့်ရအောင်

```
1 firstnum = input("Enter First Number ? : ")
2 secondnum = input("Enter Second Number ? : ")
3
4 try :
5     firstnum = int(firstnum)
6     secondnum = int(secondnum)
7
8     if secondnum <= 0 :
9         print ("Second number must be greater than 0")
10    else:
11        print (firstnum, " divided by ",secondnum)
12        print (firstnum/secondnum)
13
14 except ValueError:
15     print ("Please enter number only")
```

run

```
10 divided by 5
2.0
```

Enter First Number ? :

OK Cancel

Enter Second Number ? :

☐ Prevent this page from creating additional dialogs

OK Cancel

အခု code မှာ ဒါမဟုတ်ခဲ့ရင် ဒါလုပ်ဆိုတာပဲ ရှိပါသေးတယ်။ ကျွန်တော်တို့တွေ တစ်ခု ထက် မက condition တွေကို စစ်နိုင်ပါတယ်။ အဲဒီ အတွက် pseudo code နဲ့ flow chart ကိုတော့ မဆွဲပြတော့ပါဘူး။

အောက်က python code လေးကို တချက်လောက် လေ့လာကြည့်ပါ။

```

1 firstnum = input("Enter First Number ? : ")
2 secondnum = input("Enter Second Number (between 1-10) ? : ")
3
4 try :
5     firstnum = int(firstnum)
6     secondnum = int(secondnum)
7
8     if secondnum <= 0 :
9         print ("Second number must be greater than 0")
10    elif secondnum < 1 or secondnum > 10 :
11        print ("Second number must be between 1-10")
12    else:
13        print (firstnum, " divided by ",secondnum)
14        print (firstnum/secondnum)
15
16 except ValueError:
17     print ("Please enter number only")
18
19
20

```

run

Second number must be between 1-10

Enter First Number ? :

5

☐ Prevent this page from creating additional dialogs

OK Cancel

Enter Second Number (between 1-10) ? :

12

☐ Prevent this page from creating additional dialogs

OK Cancel

2.7 Calculator

အခု ကျွန်တော်တို့တွေ အပေါင်း အနှုတ် အမြောက် အစား လုပ်တဲ့ calculator လေး တစ်ခု ရေးရအောင်။

အရင်ဆုံး Pseudo code စရေးပါမယ်။

```
Print "Enter First Number"
READ firstnum

Print "Enter Operator (+,-,*,/)"
READ operator

Print "Enter Second Number"
READ secondnum
output = true
if operator is + then
    result = firstnum + secondnum
else if operator is - then
    result = firstnum - secondnum
else if operator is * then
    result = firstnum * secondnum
else if operator is / then
    result = firstnum/secondnum
else
    Print "Wrong Operator"
    output = false

if output == true
    Print "Result is " , result
```

Code က တော့ ရှင်းရှင်းလေးပါ။ ကျွန်တော်တို့တွေ နံပါတ် ၂ ခု လက်ခံမယ်။ ပြီးရင် Operator ကို လက်ခံမယ်။ operator ပေါ်မှာ မူတည်ပြီးတော့ result ကို ထုတ်ပြမယ်။

Operator က + - * / ထဲက မဟုတ်ရင် မထုတ်ပြပါဘူး။ အဲဒီ အတွက် ကျွန်တော်တို့တွေ boolean variable ကို အသုံးပြုပါတယ်။ output ကို ထုတ်ပြမယ် ဆိုပြီး output = true ဆိုပြီး ရေးထားတာပါ။ ဒါပေမယ့် Operator မှားနေရင် result ကို ထုတ်ပြလို့ မရတော့ပါဘူး။ ဒါကြောင့် false ပြောင်းလိုက်တာ ကို တွေ့ရပါလိမ့်မယ်။

Python နဲ့ရေးကြည့်ရအောင်။

```

1 x = input("Enter first value : ")
2 y = input("Enter second value : ")
3 op = input("Operator (+ - * /) : ")
4 try :
5     x = int(x)
6     y = int(y)
7
8     output = True
9     if op == "+" :
10         result = x+y
11     elif op == "-" :
12         result = x-y
13     elif op == "*" :
14         result = x*y
15     elif op == "/" :
16         result = x/y
17     else :
18         output = False
19         print("Wrong Operator")
20
21     if output :
22         print("Result is ",result)
23
24
25 except ValueError:
26     print("Please enter number only")
27     print(ValueError);
28

```

run

Result is 5.0

Enter first value :

OK Cancel

Enter second value :

☐ Prevent this page from creating additional dialogs

OK Cancel

Operator (+ - * /) :

☐ Prevent this page from creating additional dialogs

OK Cancel

2.8 Looping

Looping ဆိုတာကတော့ ထပ်ခါ ထပ်ခါ လုပ်တဲ့ အခါတွေ မှာ အသုံးပြုပါတယ်။

ဥပမာ ၀ ကနေ ၉ အထိ ကို ထုတ်ပြချင်တယ်။ ဒါဆိုရင်တော့ print ၁၀ ကြောင်းရေးရပါမယ်။
ဒါမှမဟုတ် ၅ ကနေ ၉ အထိ ထုတ်ပြချင်ရင်တော့ ၄ ခါ ရေးရမယ်။

တစ်ခါတစ်လေ user ဆီကနေ နောက်ဆုံး ဂဏန်းကို လက်ခံပြီး အဲဒီ အကြိမ် အရေ အတွက် လိုက် ထုတ်ပြရမယ်။ အဲဒီ အခါမှာ ကျွန်တော်တို့တွေ print ဘယ်နှစ်ကြိမ် ရိုက်ထုတ်ပြရမယ်ဆိုတာကို မသိတော့ဘူး။ အဲဒီလို အခြေအနေတွေ အတွက် Looping ကို အသုံးပြုနိုင်ပါတယ်။

Looping အမျိုးစားက ပုံမှန်အားဖြင့်

- For Loop
- While Loop
- Do While Loop

ဆိုပြီး ၃ မျိုး ရှိပါတယ်။ Python မှာတော့ Do While Loop ကို support မလုပ်ထားပါဘူး။

For Loop

အကြိမ် အရေ အတွက် အတိအကျ ရှိတယ်ဆိုရင်တော့ For Loop ကို အသုံးပြုနိုင်ပါတယ်။
ဘယ်ကနေ ဘယ်ကနေ စပြီး ဘယ် အထိ သိတယ်ဆိုရင်တော့ For Loop ကို အသုံးပြုနိုင်ပါတယ်။

```

1 # will print 0 to 9
2 for i in range(10):
3     print(i)
4
5 print("----")
6
7 # will print 5 to 9
8 for i in range(5,10):
9     print(i)

```

run

```

5
6
7
8
9

```

ဒီ code လေးမှာ ရေးထားတာလေးက အတော်ကို ရှင်းပါတယ်။

While Loop

အကြိမ်အရေအတွက်ကို ကျွန်တော်တို့ မသိဘူး ဒါမှမဟုတ် condition တစ်ခုခု ကို ထားပြီးတော့ loop လုပ်မလား မလုပ်ဘူးလား ဆိုတာကို သိချင်တဲ့ အခါမှာတော့ While Loop ကို အသုံးပြုလို့ရပါတယ်။

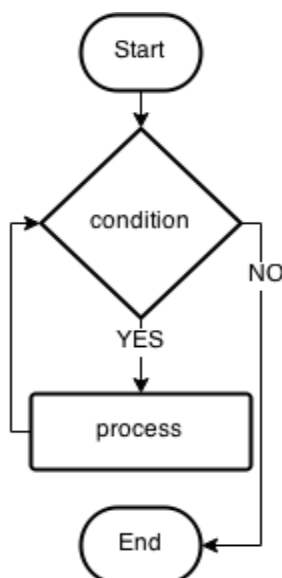
```
1 count = 0
2 while (count < 9):
3     print('The count is:', count)
4     count = count + 1
```

run

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
```

အထက်ပါ code လေး အတိုင်းဆိုရင်တော့ 0 ကနေ 8 အထိ ကို ထုတ်ပြပါလိမ့်မယ်။

While Loop ကို ပိုရှင်းသွားအောင် Flow Chart လေးကို ကြည့်ကြည့်ပါ။



condition က မှန်နေသ၍ ဒါကို ထပ်ခါ ထပ်ခါ လုပ်နေမယ်လို့ ဆိုတာပါ။

ဥပမာ။ ကျွန်တော်တို့ User ဆီကနေ 0 ကနေ 9 အတွင်း နံပါတ် တောင်းတယ်။ 0 ကနေ 9 အတွင်း ဂဏန်း မဟုတ်သ၍ ကျွန်တော်တို့တွေ user ဆီကနေ တောင်းနေမှာပဲ။ အဲဒီလိုမျိုး အကြိမ်အရေအတွက် အတိအကျ မရှိတဲ့ Looping တွေအတွက် while loop ကို အသုံးပြုပါတယ်။

```

1 x = False
2
3 while x == False :
4     value = input("Enter the number between 0 and 9: ")
5
6     try:
7         value = int(value)
8
9         if value > 9:
10            print("Your value is over 9")
11        elif value < 0:
12            print("Your value is less than 0")
13        else:
14            print("Your number is ",value)
15            x = True
16    except ValueError:
17        print("Please enter the number between 0 and 9")

```

run

Your number is 7

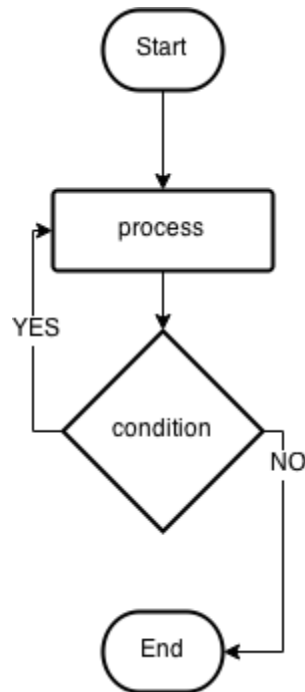
Enter the number between 0 and 9:

OK Cancel

ဒီ code လေးကို စစ်ကြည့်လိုက်ရင် while loop က ဘယ်လို နေရာတွေ မှာ အသုံးဝင် သလဲ ဆိုတာကို တွေ့နိုင်ပါတယ်။

Do While Loop

Do while loop က while loop လိုပဲ။ ဒါပေမယ့် အနည်းဆုံး တစ်ကြိမ် အလုပ်လုပ်ပါတယ်။



Python မှာကတော့ do while loop အတွက် သီးသန့် looping မရှိပါဘူး။

ဒါကြောင့် Java code လေး ကို ကြည့်ကြည့်ပါ။

```

int count = 1;
do {
    System.out.println("Count is: " + count);
    count = count + 1;
} while (count < 11);
  
```

System.out.println က python က print နဲ့တူပါတယ်။

အဲဒီ code မှာဆိုရင်တော့ Count is 1 ကို အရင်ဆုံး ထုတ်ပြပါတယ်။ ပြီးတော့ count ကို ၁ တိုးတယ်။ တိုးပြီးမှ count က ၁၁ ထက် ငယ်နေလား ဆိုပြီး စစ်ပါတယ်။

Python နဲ့ အနီးစပ် ဆုံး ရေးပြရင်တော့

```
1 count = 1
2 print('The count is:', count)
3 count = count + 1
4 while (count < 11):
5     print('The count is:', count)
6     count = count + 1
```

run

```
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
The count is: 9
The count is: 10
```

Exercise

အခု ကျွန်တော်တို့တွေ အောက်ကလိုမျိုး ပုံလေး ထုတ်ကြည့်ရအောင်

```
*
**
***
****
*****
```

ကျွန်တော်တို့ တစ်ကြောင်းဆီ ရိုက်ထုတ်မယ် ဆိုရင် ရပါတယ်။ ဒါပေမယ့် Looping နဲ့ ပတ်ပြီး ထုတ်ကြည့်ရအောင်။

ပထမဆုံးအဆင့်က Looping ဘယ်နှစ်ကြိမ် ပတ်ရမလဲဆိုတာကို စဉ်းစားဖို့ လိုတယ်။

1. *
2. *
3. *
4. *
5. *

ဒေါင်လိုက်က ငါးခါ ပတ်ရမယ်။ ဒီတော့

```
1 for x in range(5):
2     print("*")
```

run

```
*
*
*
*
*
```

Loop ၅ ခါ ပတ်ပြီးတော့ print ထုတ်လိုက်တယ်။ * ငါးလိုင်းတော့ ရပြီ။

အလျားလိုက်က တစ်ခါ။

ဒေါင်လိုက်က တစ်ခါ ဆိုတော့ looping ၂ ခါ ပတ်ရမယ်။

```
1 for x in range(5):
2     for k in range(5):
3         print("*", end="")
```

run

```
*****
```

```
1 for x in range(5):
2     for k in range(5):
3         print("*", end="")
4     print("")
```

run

```
*****
*****
*****
*****
*****
```

```
1 for x in range(5):  
2     for k in range(x):  
3         print("*", end="")  
4     print("")
```

run

```
*  
**  
***  
****
```

```
1 for x in range(1,6):  
2     for k in range(x):  
3         print("*", end="")  
4     print("")
```

run

```
*  
**  
***  
****  
*****
```

Questions

Code: Q1

```
total = 0;  
for x in range(10):  
    total = total + x  
print(total)
```

Code: Q2

```
total = 0;  
for x in range(10):  
    total = total + 1  
print(total)
```

Quiz

Quiz ကို ဖြေဆိုရန် http://books.saturngod.net/programming_basic/CH2/looping.html ကို နှိပ်ပါ။

Quiz

အမေးအဖြေလေးတွေ နည်းနည်း လုပ်ကြည့်ရအောင်။

Question 1 of 5

Code Q1 က program ကို run လိုက်ရင် Total က

- ☐ 0
- ☐ 10
- ☐ 45

Question 2 of 5

range(5) ဆိုရင်

- ☐ 0 ကနေ 5 ထိ
- ☐ 0 ကနေ 4 ထိ
- ☐ 1 ကနေ 5 ထိ
- ☐ 1 ကနေ 4 ထိ

Question 3 of 5

Code Q2 က program ကို run လိုက်ရင် Total က

- ☐ 0
- ☐ 10
- ☐ 45

Question 4 of 5

While loop က အနည်းဆုံး တစ်ကြိမ် အလုပ်လုပ်တယ်။

- ☐ မှန်
- ☐ မှား

Question 5 of 5

While loop အသုံးပြုဖို့ အခေါက် အရေအတွက် အတိအကျ ရှိရမယ်။

- ☐ မှန်
- ☐ မှား

[Submit](#)
[Solution](#)

အခု ကျွန်တော်တို့တွေ for loop ကို နည်းနည်း နားလည် လောက်ပါပြီ။

အခု နောက်ထပ် exercise လေးကို ကိုယ်တိုင် ရေးကြည့်ပါ။

Questions 1

Fibonacci program လို့ ခေါ်ရအောင်။

ကျွန်တော်တို့တွေ user ဆီက နံပါတ် လက်ခံမယ်။

နံပါတ် က 5 ဖြစ်ရင် Fibonacci sequence အရ နံပါတ် ၅ ခု ထုတ်ပြမယ်။

1 1 2 3 5

နံပါတ် 7 ဖြစ်ခဲ့ရင်တော့

1 1 2 3 5 8 13

လို့ ထုတ်ပြမယ်။

Fibonacci sequence ဆိုတာကတော့ ရှေ့က နံပါတ် ၂ ခု ကို ပေါင်းပြီးတော့ နောက်ထပ် ဂဏန်း တစ်ခု ရပါတယ်။

Question 2

Even/odd စစ်ထုတ်တဲ့ program ပါ။ user ဆီကနေ ဂဏန်း လက်ခံမယ်။ ပြီးရင် 1 ကနေ စပြီးတော့ even ဖြစ်လား odd ဖြစ်လား ဆိုပြီး ထုတ်ပြရမယ်။

ဥပမာ။ ။ User က 3 လို့ ရိုက်လိုက်ရင်

```
1 is Odd
2 is Even
3 is Odd
```

ဆိုပြီး ထုတ်ပြမယ်။ တကယ်လို့ 5 လို့ ရိုက်လိုက်ရင်

```
1 is Odd
2 is Even
3 is Odd
4 is Even
5 is Odd
```

ဆိုပြီး ထုတ်ပြရပါမယ်။

2.9 Array

အခု အရေးကြီးတဲ့ အခန်းကို ရောက်လာပါပြီ။ Array ကို သေသေချာချာ နားလည် ဖို့ လိုအပ်ပါတယ်။ နားမလည်တာတွေကို Github မှာ [issue](#) ဖွင့်ပြီး မေးနိုင်ပါတယ်။

Array ဆိုတာ ဘာလဲ ?

Array ဆိုတာကတော့ variable တွေ အများကြီးကို သိမ်းထားတဲ့ variable တစ်ခုပါပဲ။ Array က variable တွေကို အခန်းနဲ့ သိမ်းပါတယ်။ နားလည်အောင် ပြောရရင်တော့ variable ဆိုတာက ရေခွက် တစ်ခု ဆိုပါတော့။ အဲဒီ ရေခွက်ကို ခဲလိုက်ရင် ရေခဲ တစ်ခုပဲ ရမယ်။ Array ဆိုတာကတော့ ရေခဲ ခဲတဲ့ ခွက် (ice cube trays) နဲ့တူပါတယ်။ ရေခဲ ခဲ ဖို့ အတွက် အကန့်လေးတွေ ပါတယ်။ ရေခဲခဲ လိုက်ရင် တစ်ခု ထက် မက ၊ အခန်း ရှိသလောက် ရေခဲ ရနိုင်ပါတယ်။



Ref: http://en.wikipedia.org/wiki/Ice_cube

Array က အဲဒီလိုပါပဲ။ ကျွန်တော်တို့ computer memory ပေါ်မှာ သိမ်းဖို့ အတွက် အခန်းလေးတွေ ယူလိုက်တယ်။ ပြီးတော့ အခန်းတွေ ထဲမှာ variable တွေ ထည့်ပြီး သိမ်းတယ်။

String array ဆိုရင်တော့ String value တွေ ပဲ ထည့်တဲ့ အခန်းပေါ့။

Integer array ဆိုရင်တော့ Integer value တွေပဲ ထည့်တဲ့ အခန်းတွေပေါ့။

အခန်းတွေကို ရေတွက်တဲ့ အခါမှာတော့ သုည ကနေ စပါတယ်။ အခန်း ၃ ခန်း ရှိရင်တော့ 0,1,2 ဆိုပြီး ရေတွက်ပါတယ်။

Example

ဥပမာ code လေးကို ကြည့်ရအောင်

```
1 list = [1,5,2,7,8,9,200,155]
2
3 # First Room
4 print(list[0])
5
6 # 9 will show
7 print(list[5])
8
9 # Last Room
10 print(list[7])
11
```

run

```
1
9
155
```

အထက်ပါ code မှာဆိုရင် အခန်းပေါင်း 8 ခန်း ရှိပါတယ်။ အခန်း တစ်ခုခြင်းဆီမှာ နံပါတ်တွေ ထည့်ထားပါတယ်။

ပထမ အခန်းကို လိုချင်တဲ့ အခါမှာ `list[0]` လို့ ဆိုပြီး ခေါ်သုံးထားပါတယ်။ အခြား အခန်းတွေ ကိုတော့ နံပါတ်လိုက် ခေါ်ထားတာကို တွေ့နိုင်ပါတယ်။

ဒီ code မှာဆိုရင်တော့ Array တစ်ခုမှာ အခန်း ဘယ်လောက်ရှိလဲဆိုတာကို ထုတ်ပြထားတာပါ။

```
1 list = [1,5,2,7,8,9,200,155]
2
3 #Total Room
4 print("Total room in array is",len(list))
```

run

Total room in array is 8

ကျွန်တော်တို့တွေ ထပ်ပြီးတော့ အခန်းထဲမှာ ရှိတဲ့ data တွေကို looping ပတ်ပြီး ထုတ်ကြည့်ရအောင်။

```
1 list = [1,5,2,7,8,9,200,155]
2
3 t = (1,2,3)
4
5 for i in range(len(list)):
6     print(list[i])
```

run

1
5
2
7
8
9
200
155

နောက်ထပ် တဆင့် အနေနဲ့ အခန်းထဲမှာရှိတဲ့ နံပါတ်တွေ အားလုံးပေါင်း ရလဒ်ကို ထုတ်ကြည့်ရအောင်

```
1 list = [1,5,2,7,8,9,200,155]
2
3 x = 0
4 for i in range(len(list)):
5     x = x + list[i]
6
7 print("Total:",x)
```

run

Total: 387

ကျွန်တော်တို့တွေ Array အတွက် loop ကို အောက်က code လိုမျိုး python မှာ loop ပတ်လို့ရပါတယ်။ တခြား language တွေမှာ ဆိုရင်တော့ **for each** loop လို့ခေါ်ပါတယ်။

```
1 list = [1,5,2,7,8,9,200,155]
2
3 x = 0
4 for i in list:
5     x = x + i
6
7 print("Total:",x)
```

run

Total: 387

အဓိပ္ပာယ်ကတော့ Array အခန်းတွေက အစကနေ အဆုံး ထိ loop ပတ်မယ်။

ရောက်နေတဲ့ index ထဲက data ကို i ထဲကို ထည့်မယ်။ အဲဒီ code မှာ ကျွန်တော်တို့တွေ လက်ရှိ index ကို မသိနိုင်ပါဘူး။

for each loop ပတ်တဲ့ အခါမှာ လက်ရှိ index ပါ သိအောင် အောက်က code လိုမျိုး ရေးလို့ရပါတယ်။

```
1 list = [1,5,2,7,8,9,200,155]
2
3 for (i,item) in enumerate(list):
4     print("Index :",i,"And Value :",item)
```

run

```
Index : 0 And Value : 1
Index : 1 And Value : 5
Index : 2 And Value : 2
Index : 3 And Value : 7
Index : 4 And Value : 8
Index : 5 And Value : 9
Index : 6 And Value : 200
Index : 7 And Value : 155
```

Immutable And Mutable

Array မှာ ၂ မျိုး ရှိတယ်။ မပြောင်းလို့ရတဲ့ Array နဲ့ ပြောင်းလဲလို့ရတဲ့ Array Type ၂ ခု ရှိပါတယ်။

Python မှာကတော့ tuple နဲ့ list ဆိုပြီး ၂ မျိုး ရှိတယ်။

Tuple ကို လက်သညးကွင်း နဲ့ ရေးပြီးတော့ list ကိုတော့ လေးထောင့် ကွင်းနဲ့ ရေးပါတယ်။

```
t = (1,2,3) # immutable
l = [1,2,3] # mutable
```

Tuple နဲ့ ရေးထားရင်တော့ အခန်း တွေကို ပြောင်းလဲလို့ မရပါဘူး။ ဒါပေမယ့် list ကတော့ အခန်းထဲက data တွေကို ပြောင်းလို့ ရသလို အခန်း အသစ်တွေ ထည့်တာ ဖျက်တာ စတာတွေကို လုပ်လို့ရပါတယ်။

Finding Max Number

အခု ကျွန်တော်တို့တွေ array အခန်းထဲက အကြီးဆုံး ဂဏန်းကို ရှာတဲ့ code လေး ရေးကြည့်ရအောင်။

```
1 list = [1048,1255,2125,1050,2506,1236,2010,1055]
2
3 maxnumber = list[0]
4
5 for x in list:
6     if maxnumber < x :
7         maxnumber = x
8
9 print("MAX number in array is",maxnumber)
```

run

MAX number in array is 2506

ဒီ Code လေးကို ကြည့်ကြည့်ပါ။ ရှင်းရှင်းလေးပါ။ ပထမဆုံး အခန်းကို အကြီးဆုံးလို့ သတ်မှတ်လိုက်တယ်။ ပြီးရင် အခြား အခန်းတွေနဲ့ တိုက်စစ်တယ်။ ကြီးတဲ့ ကောင်ထဲကို maxnumber ဆိုပြီး ထည့်ထည့်သွင်းထားတယ်။

နောက်ဆုံး Looping ပြီးသွားရင် အကြီးဆုံး ဂဏန်းကို ကျွန်တော်တို့တွေ သိရပြီပေါ့။

Questions

- Max Number လိုမျိုး အငယ်ဆုံး ဂဏန်း Min Number ကို ရှာတဲ့ code ရေးကြည့်ပါ။
- Array [3,4,1,2,9,7] ဆိုပြီး ရှိပါသည်။ user ဆီက နံပါတ်ကို လက်ခံပြီး array အခန်းထဲတွေ တွေ့မတွေ့ user ကို print ထုတ်ပြပါမယ်။ တွေ့ခဲ့ပါက အခန်း ဘယ်လောက်မှာ တွေ့ခဲ့သည်ကို print ထုတ်ပြပါမယ်။
- Max number ကို ရှာပါ။ ဘယ်နံပါတ်က အကြီးဆုံးလဲ။ ဘယ်အခန်းမှာ ရှိတာလဲ ဆိုတာကို print ရိုက် ပြပါ။

2.10 Funcation

ကျွန်တော်တို့တွေ programming နဲ့ပတ်သက်ပြီးတော့ အတော်လေးကို သိပြီးပါပြီ။ အခု အပိုင်းမှာတော့ function အကြောင်းကို ပြောပြပါမယ်။ ကျွန်တော်တို့ ထပ်ခါထပ်ခါ ခေါ်လုပ်နေရတဲ့ ကိစ္စတွေမှာ ကျွန်တော်တို့တွေ looping သုံးခဲ့ပါတယ်။ အဲလိုပဲ code တွေ ထပ်နေရင် ဒါမှမဟုတ် ပိုပြီးတော့ အဓိပ္ပာယ် ပြည့်စုံအောင် ကျွန်တော်တို့တွေ function ခွဲရေးပါတယ်။

```
1 def printHello():
2     print("HELLO")
3
4 printHello()
```

run

HELLO

ဒီ code လေးမှာ ဆိုရင် ကျွန်တော်တို့တွေ printHello ဆိုတဲ့ function လေး ရေးထားတာကို တွေ့နိုင်ပါတယ်။ Hello ကို ခဏခဏ print ရိုက်နေမယ့် အစား printHello ဆိုတဲ့ function လေးကို ခေါ်လိုက်တာနဲ့ HELLO ဆိုပြီး ထုတ်ပြပေးနေမှာပါ။

Python မှာ function ကို ရေးတဲ့ အခါမှာတော့ def နဲ့ စတယ်။ ပြီးတော့ function နာမည်။ အခု ဥပမာမှာ printHello က function နာမည်ပါ။

လက်သည်းကွင်းစ နဲ့ ကွင်းပိတ်ကို တွေ့မှာပါ။ အဲဒါကတော့ function စီကို data တွေ ပို့ဖို့အတွက် အသုံးပြုပါတယ်။ ဘာ data မှ မထည့်ပေးလိုက်ချင်ဘူးဆိုရင်တော့ () နဲ့ အသုံးပြုနိုင်ပါတယ်။

```
1 def printHello(val):
2     print("HELLO",val)
3
4 printHello("WORLD")
5 printHello("Python")
```

run

HELLO WORLD
HELLO Python

ဒီ ဥပမာမှာတော့ World ဆိုပြီး value လေးကို function ဆီ ပို့ပေးလိုက်ပါတယ်။ function ကနေ Hello ကို ရှေ့မှာ ထားပြီးတော့ HELLO World ဆိုပြီး ထုတ်ပေးပါတယ်။ နောက်တစ်ခေါက်မှာတော့ Python ဆိုတာကို ပို့ပေးလိုက်တဲ့ အတွက် HELLO Python ဆိုပြီး ထပ်ထွက်လာပါတယ်။ တူညီနေတဲ့ code ၂ ခေါက်ရေးနေမယ့် အစား function နဲ့ ခွဲထုတ်လိုက်တာပါ။

```
1 def sum(val1, val2) :
2     return val1+val2
3
4 print("SUM : ", sum(1,4))
```

run

SUM : 5

ဒီ code လေးကို ကြည့်ကြည့်ပါ။ ပုံမှန် အပေါင်းကို ကျွန်တော်တို့တွေ function ခွဲထုတ်ပြီးတော့ ရေးထားတာပါ။ ကျွန်တော်တို့တွေ 1+4 ဆိုပြီး လွယ်လင့်တကူ ရေးလို့ရပါတယ်။ သို့ပေမယ့် ပေါင်းတယ်ဆိုတဲ့ အဓိပ္ပာယ်သက်ရောက်အောင် sum ဆိုပြီး function သီးသန့် ခွဲထုတ်လိုက်ပါတယ်။ ကိန်း ၂ လုံး ကို ပေါင်းပြီးတော့ ရလဒ်ကို ပြန်ပေးထားပါတယ်။ ကျွန်တော်တို့တွေ function မှာ parameter တစ်ခုမှ မပို့ပဲ နေလို့ရသလို တစ်ခု သို့မဟုတ် တစ်ခု ထက် မက ပို့လို့ရပါတယ်။

အခု ဆိုရင်တော့ function ကို နည်းနည်း သဘောပေါက်လောက်ပါပြီ။

နောက်ထပ် ဥပမာ ကြည့်ရအောင်။ Array တုန်းက max number ကို ကျွန်တော်တို့တွေ ရေးခဲ့ဖူးပါတယ်။

```

1 list = [1048,1255,2125,1050,2506,1236,2010,1055]
2
3 maxnumber = list[0]
4
5 for x in list:
6     if maxnumber < x :
7         maxnumber = x
8
9 print("MAX number in array is",maxnumber)

```

run

```
MAX number in array is 2506
```

အဲဒီမှာ list ကသာ ၂ ခု ရှိမယ်ဆိုပါစို့။ ကျွန်တော်တို့တွေ max number ရ ဖို့အတွက် ဒီ code ကို ပဲ ၂ ခေါက်ထပ်ရေးရမယ်။

```

1 list = [1048,1255,2125,1050,2506,1236,2010,1055]
2
3 maxnumber = list[0]
4
5 for x in list:
6     if maxnumber < x :
7         maxnumber = x
8
9 print("MAX number in array list is",maxnumber)
10
11 list2 = [1,2,5,6,9,3,2]
12
13 maxnumber = list2[0]
14
15 for x in list2:
16     if maxnumber < x :
17         maxnumber = x
18
19 print("MAX number in array list 2 is",maxnumber)

```

run

```
MAX number in array list is 2506
MAX number in array list 2 is 9
```


တကယ်လို့ Array ခု အတွက် ဆိုရင် ဒီ code ကို ပဲ ခေါက်ထပ်ရေးနေရမယ်။ အဲလို ထပ်ခါ ထပ်ခါ မရေးရအောင် ကျွန်တော်တို့တွေ function ခွဲပြီး ရေးလို့ရပါတယ်။

```

1 def max(lst):
2     maxnumber = lst[0]
3
4     for x in lst:
5         if maxnumber < x :
6             maxnumber = x
7     return maxnumber
8
9
10
11 list = [1048,1255,2125,1050,2506,1236,2010,1055]
12 list2 = [1,2,5,6,9,3,2]
13
14 print("MAX number in array list is",max(list))
15 print("MAX number in array list2 is",max(list2))

```

run

```

MAX number in array list is 2506
MAX number in array list2 is 9

```

အဲဒီမှာ code က ပိုပြီး ရှင်းသွားတာကို တွေ့နိုင်ပါတယ်။ Array ဘယ်နှစ်ခုပဲ ဖြစ်ဖြစ် ဒီ function ကို ခေါ်ရုံပါပဲ။ function ကို သုံးချင်းအားဖြင့် ထပ်ခါထပ်ခါ ခေါ်နေတာတွေကို သက်သာသွားစေပါတယ်။

Questions

triangle_star ဆိုတဲ့ function ကို ရေးပါ။ user ဆီက နံပါတ်တောင်းပါ။ 3 လို့ရိုက်ရင် triangle_star(3) ဆိုပြီး ပို့ပေးပါ။ triangle_star မှ အောက်ပါ အတိုင်း ရိုက်ထုတ်ပြပါ။

```

*
**
***

```

အကယ်၍ 5 လို့ ရိုက်ထည့်လျှင် ၅ လိုင်း ထုတ်ပြပါမည်။

2.11 Exercise Answers

Total Number

```
1 total = 0;
2 for x in range(10):
3     total = total + x
4 print(total)
5
6 total = 0;
7 for x in range(10):
8     total = total + 1
9 print(total)
```

```
45
10
```

Fibonacci

```
1 x = input("Total number : ")
2 try :
3     x = int(x)
4     f = 0
5     s = 1;
6     for k in range(x):
7         print(s, end=" ") # end is for ending with space and don't use another
8         t = s
9         s = f + s
10        f = t
11    print ("") #just for line break in terminal
12
13 except ValueError:
14     print("Please enter number only")
15
```

```
1 1 2 3 5
```

Total number :

5

OKCancel

Even/Odd

```
1 x = input("Enter Number : ")
2 try :
3     x = int(x)
4     for k in range(1,x+1):
5         if k % 2 == 0 :
6             print(k , " is Even")
7         else:
8             print(k, " is Odd")
9
10 except ValueError:
11     print("Please enter number only")
```

run

```
1 is Odd
2 is Even
3 is Odd
4 is Even
5 is Odd
```

Enter Number :

OK Cancel

Min Number

```
1 list = [1048,1255,2125,1050,1001,1236,2010,1055]
2
3 minnumber = list[0]
4
5 for x in list:
6     if minnumber > x :
7         minnumber = x
8
9 print("Min number in array is",minnumber)
```

run

```
Min number in array is 1001
```

Search In Array

```
1 list = [3,4,1,2,9,7]
2
3 x = input("Enter number to search : ")
4 try :
5     x = int(x)
6     found = False
7     for (i,item) in enumerate(list):
8         if(item==x):
9             found = True
10            print("Found at",i)
11            break;
12
13     if found==False :
14         print ("not found in the list")
15
```

run

Found at 5

Enter number to search :

7

OK

Cancel

Max Numer With Room

```
1 list = [1048,1255,2125,1050,2506,1236,2010,1055]
2
3 maxnumber = list[0]
4 room = 0;
5 for (i,item) in enumerate(list):
6     if maxnumber < item :
7         maxnumber = item
8         room = i
9
10 print("Min number in array is",maxnumber,"Found at room",room)
```

run

Min number in array is 2506 Found at room 4

Function Triangle Star

```
1 def triangle_star(row):
2     for x in range(1,row+1):
3         for k in range(x):
4             print("*", end="")
5         print("")
6
7 x = input("How many rows : ")
8 try :
9     x = int(x)
10    triangle_star(x)
11
12 except ValueError:
13     print("Please enter number only")
14
```

run

How many rows :
5

OK Cancel

*
**

Chapter 3 :: Object Oriented

Python ဟာ Object Oriented Programming Language တစ်ခုပါ။ နောက်လာမယ့် အခန်းမှာ Stack , Queue စတာတွေကို python ကို အသုံးပြုပြီး ရေးသားမှာ ဖြစ်သည့်အတွက်ကြောင့် OOP ကို အနည်းအကျဉ်းတော့ သိထားဖို့ လိုပါတယ်။ OOP အကြောင်းကို ပြောမယ်ဆိုရင်တော့ ဒီစာအုပ်မှာ မလောက်ပါဘူး။ ဒီစာအုပ်က programming အခြေခံအတွက် ဖြစ်တဲ့ အတွက်ကြောင့် နောက်အခန်းတွေ အတွက် OOP အကြောင်း အနည်းငယ်မျှသာ ဖော်ပြသွားပါမယ်။

3.1 Overview

Object Oriented ဆိုတာကတော့ programming ကို ရေးသားရာမှာ သက်ဆိုင်ရာ အစုလိုက် ခွဲထုတ်ပြီး ရေးသားထားတာပါ။ ဥပမာ။။ လူတစ်ယောက် ဆိုပါဆို။ လူ ဟာ Object တစ်ခုပါ။ လူတစ်ယောက်မှာ ကိုယ်ပိုင် function တွေ ရှိမယ်။ ပိုင်ဆိုင်တဲ့ properties တွေ ရှိမယ်။ properties တွေကတော့ မျက်လုံး ၊ ပါးစပ် စတာတွေ ဖြစ်ပြီးတော့ function တွေကတော့ စကားပြောတာ အိပ်တာ စတာတွေပါ။

နောက်ထပ် နားလည်အောင် ထပ်ပြီး ရှင်းပြရရင်တော့ ကားတစ်စီးကို object လို့ သတ်မှတ်လိုက်ပါ။ သူ့မှာ ဘာ properties တွေ ရှိမလဲ။ ဘာ function တွေ ရှိမလဲ။ ရှိနိုင်တဲ့ properties တွေကတော့ ဘီး ၄ ခု ရှိမယ်။ တံခါး ရှိမယ်။ function တွေကတော့ ရှေ့သွားတာပါမယ်။ နောက်သွားတာပါမယ်။ ဘယ်ကွေ့ ညာကွေ့တွေ ပါမယ်။

Programming မှာ ဖန်တီးတဲ့ အခါမှာလည်း Object ကို ဖန်တီးတယ်။ ပြီးရင် properties တွေ function တွေကို သက်ဆိုင်ရာ Object မှာ ထည့်သွင်းပါတယ်။

3.2 Classes

Class ဆိုတာကတော့ object တစ်ခု ဖန်တီးဖို့ အတွက် user-defined လုပ်ထားတာပါ။ class ထဲမှာ attributes , class variables , method စတာတွေ ပါဝင်ပါတယ်။

Defining a Class

Python မှာ class ကို ဖန်တီးတော့မယ်ဆိုရင်

```
class ClassName:
```

ClassName ကတော့ နှစ်သက်ရာ class နာမည်ပါ။ ဥပမာ

```
class animal:
```

အဲဒါဆိုရင်တော့ animal ဆိုတဲ့ class တည်ဆောက်ပြီးပါပြီ။ ပြီးရင် class ထဲမှာ ပါဝင်မယ် variable ကို သတ်မှတ်ပါမယ်။ animal ဖြစ်တဲ့ အတွက်ကြောင့် ခြေထောက် ၄ ချောင်းဖြစ်နိုင်သလို ၂ ချောင်းတည်းရှိတဲ့ တိရစ္ဆာန်လည်း ဖြစ်နိုင်ပါတယ်။ ဒါကြောင့်

```
class animal:
    number_of_legs = 0
```

Instances

Class ကြီး သက်သက်ဆိုရင်တော့ class တစ်ခု ကို ဖန်တီးထားတာပဲ ရှိပါတယ်။ class ကို အသုံးပြုချင်ရင်တော့ instance တစ်ခုကို တည်ဆောက်ရပါတယ်။ တည်ဆောက်ပြီးသား instance ကို variable ထဲမှာ သိမ်းရပါတယ်။ class ထဲက variable တွေကို ခေါ်ယူ အသုံးပြုလိုရင်တော့ instance ဆောက်ထားတဲ့ variable ထဲက နေ တဆင့် ခေါ်ယူ အသုံးပြုနိုင်ပါတယ်။

```
class animal:
    number_of_legs = 0
```

```
dog = animal()
```

အခု ဆိုရင် dog variable က animal object တစ်ခု ဖြစ်သွားပါပြီ။ animal ထဲက variable ကို ခေါ်ယူ အသုံးပြုလို ရင်တော့

dog.number_of_legs

အခု ကျွန်တော်တို့တွေ variable ကို အသုံးပြုကြည့်ရအောင်။

```
1 class animal:
2     number_of_legs = 0
3
4     dog = animal()
5     dog.number_of_legs = 4
6
7     print ("Dog has {} legs".format(dog.number_of_legs))
8
9     '''
10    you can also write like that
11
12    print("Dog has " + str(dog.number_of_legs) + " legs")
13    '''
14
```

run

Dog has 4 legs

ကျွန်တော်တို့တွေ နောက်ထပ် ဥပမာ တစ်ခု ထပ်စမ်း ကြည့်ရအောင်။

```

1 class animal:
2     number_of_legs = 0
3
4     dog = animal()
5     dog.number_of_legs = 4
6
7     print ("Dog has {} legs".format(dog.number_of_legs))
8
9     chicken = animal()
10    chicken.number_of_legs = 2
11
12    print ("Chicken has {} legs".format(chicken.number_of_legs))
13
14

```

run

```

Dog has 4 legs
Chicken has 2 legs

```

ကျွန်တော်တို့တွေ dog နဲ့ chicken object ၂ ခုကို တည်ဆောက်ပြီးတော့ ခြေထောက် ဘယ်နှစ်ချောင်း ရှိတယ်ဆိုတာကို print ထုတ်ပြပေးထားပါတယ်။

Function in Class

Function တွေကို class ထဲမှာ ကြေငြာလို့ ရပါတယ်။

```

1 class animal:
2     number_of_legs = 0
3     def sleep(self) :
4         print("zzz")
5
6     dog = animal()
7     dog.sleep()

```

run

```

zzz

```

class ထဲမှာ function ကို ဖန်တီးတဲ့ အခါမှာ (self) ဆိုပြီး ထည့်ထားတာကို တွေ့ရပါမယ်။ အဲဒီလို ထည့်ထားမှသာ class ထဲမှာ ရှိတဲ့ variable ကို လှမ်းခေါ်လို့ ရပါလိမ့်မယ်။

3.3 Inheritance

ပြီးခဲ့တဲ့ အခန်းကတော့ ကျွန်တော်တို့တွေ class အကြောင်း အနည်းငယ် သိပြီးပါပြီ။ အခု အခန်းမှာတော့ Inheritance အကြောင်း အနည်းငယ် ဖော်ပြပေးပါမယ်။

Inheritance ဆိုတာကတော့ အမွေဆက်ခံခြင်း တနည်းအားဖြင့် ပင်မ class ရဲ့ child class ဖန်တီးခြင်းပါပဲ။ ပြီးခဲ့တဲ့ အခန်းက animal class ကို ကျွန်တော်တို့ ဖန်တီးပြီးတော့ dog object တွေ ဆောက်ခဲ့ကြတယ်။ အခု ကျွန်တော်တို့ dog class ဖန်တီးပါမယ်။ dog ဆိုတာက animal ဆိုတဲ့ class ရဲ့ child ပါပဲ။

```

1 class animal:
2     number_of_legs = 0
3
4     def sleep(slef) :
5         print("zzz")
6
7     def count_legs(self) :
8         print("I have {} legs".format(self.number_of_legs))
9
10 class dog(animal):
11     def bark(self):
12         print("Woof")
13
14 mydog = dog()
15 mydog.bark();
16 mydog.sleep();

```

run

Woof
zzz

ဒီ code မှာ ဆိုရင်တော့ Woof နဲ့ zzz ကို တွေ့နိုင်ပါတယ်။ dog class ဟာ သူ့ parent မှာ လုပ်လို့ရတဲ့ function တွေကို ခေါ်ပြီး အသုံးပြုနိုင်တာကို တွေ့နိုင်ပါလိမ့်မယ်။

အခြား language တွေမှာတော့ class ရဲ့ function တွေကို private , public , protected ဆိုပြီး ပေးထားလို့ ရပေမယ့် python language မှာတော့ အဲဒီ feature မပါဝင်ပါဘူး။

Object Oriented နဲ့ ပတ်သက်ပြီးတော့ ဒီစာအုပ်မှာတော့ ဒီလောက်ပါပဲ။ နောက်ထပ် အခန်းတွေမှာ လက်တွေ့တည်ဆောက်ရင်းနဲ့ OOP ကို ပိုပြီး နားလည်လာပါလိမ့်မယ်။

Chapter 4: DataStructure

4.1 Basic Data Structures

Data Structure ရဲ့ အဓိက ရည်ရွယ်ချက်ကတော့ stack, queue , deque, list စတာတွေကို သိရှိနားလည် စေဖို့ပါ။ ဒီ အခန်းမှာ အဓိက အားဖြင့် Stack ဆိုတာဘာလဲ။ Queue ဆိုတာဘာလဲ စတာတွေကို မိတ်ဆက်ပေးသွားပြီးတော့ အဓိက Array ကို နားလည်ပြီး အသုံးပြုတတ်ဖို့ အတွက်ပါ။

4.2 What is a Stack ?

stack ဆိုတာကတော့ အစီအစဉ်ကျ စီထားထားတော့ items collection လို့ ဆိုရပါမယ်။ အသစ်အသစ်တွေက ကျန်နေတဲ့ data ပေါ်မှာ ထပ်ဖြည့်သွားပါတယ်။ ပြန်ထုတ်မယ်ဆိုရင် နောက်ဆုံး ထည့်ထားတဲ့ data ကနေ ပြန်ထုတ်ရပါတယ်။ **LIFO (last-in-first-out)** လို့ ဆိုပါတယ်။ ဥပမာ။။ ကျွန်တော်တို့ စာအုပ် ပုံနဲ့ တူပါတယ်။ စာအုပ် ပုံမှာ အောက်ကလို ရှိပါတယ်။

- python
- javascript
- css
- html

နောက်ထပ် စာအုပ် တစ်အုပ်ဖြစ်တဲ့ Data Structure ဆိုတဲ့ စာအုပ်ကို စာအုပ်ပုံမှာ ထပ် တင်လိုက်ရင်တော့

- Data Structure
- python
- javascript
- css
- html

ဆိုပြီး ဖြစ်သွားပါမယ်။ စာအုပ်ပုံကနေ စာအုပ်ကို ထုတ်မယ်ဆို အပေါ်ဘက်ကနေ ပြန်ထုတ်မှ ရပါမယ်။

ဥပမာ javascript စာအုပ်ကို လိုချင်ရင် Data Structure နှင့် Python ဆိုတဲ့ စာအုပ် ၂ အုပ်ဖယ်ပြီးမှ Javascript စာအုပ်ကို ဆွဲထုတ်လို့ ရပါလိမ့်မယ်။ Javascript စာအုပ်ကို ယူလိုက်ရင် stack က css, html ဆိုပြီး ဖြစ်သွားပါလိမ့်မယ်။

အခု ဆိုရင် stack ဆိုတာကို စာသဘော အားဖြင့် နားလည်လောက်ပါပြီ။ ကျွန်တော်တို့ stack ကို python နဲ့ ဖန်တီးကြည့်ရအောင်။

4.3 Stack Abstract Data Type

Stack မှာ ပါဝင်မယ့် data type တွေ လုပ်ဆောင်မယ့် အရာတွေ အကို အရင် ဆုံး ကျွန်တော်တို့ စဉ်းစားကြပါမယ်။ stack က LIFO ဖြစ်တဲ့ အတွက် List လိုမျိုး ကြိုက်တဲ့ နေရာကနေ ဆွဲထုတ်လို့ မရပါဘူး။ အပေါ်က data ကို ပဲ ဆွဲထုတ်ခွင့်ရှိပါတယ်။

Stack() Stack ဆိုတဲ့ class ကို ကျွန်တော်တို့ တွေ ဖန်တီးပါမယ်။ constructor တွေ မလိုအပ်ပါဘူး။

push(item) ကတော့ item အသစ်ကို ဖန်တီးထားတဲ့ stack ထဲကို ထည့်ဖို့ အတွက်ပါ။

pop() ကတော့ stack ထဲကနေ အပေါ်ဆုံး item ကို ထုတ်ဖို့ အတွက်ပါ။ ထုတ်လိုက်တဲ့ value ကိုတော့ return မလုပ်ပါဘူး။

peek() ကတော့ stack ထဲက အပေါ်ဆုံး item ကို ထုတ်မယ်။ ပြီးတော့ return ပြန်ပေးပါမယ်။

is_empty() ကတော့ stack က empty ဖြစ်လား မဖြစ်ဘူးလား ဆိုတာကို စစ်ပြီးတော့ boolean value ကို return ပြန်ပေးပါမယ်။

size() ကတော့ stack ထဲမှာ စုစုပေါင်း data ဘယ်လောက် ရှိသလဲ ဆိုတာကို return ပြန်ပေးမယ်။ ပြီးတော့ 4 နဲ့ dog ကို ထည့်တယ်။ peek လုပ်တယ်။ နောက်ဆုံး ထည့်ထားတဲ့ dog ကို ရတယ်။ နောက်ပြီးတော့ True ထည့်တယ်။ အခန်းက ၃ ခု ဖြစ်သွားတာ ဟုတ်မဟုတ် စစ်ကြည့်တယ်။ ပြီးတော့ 8.4 ထည့်ပြီးတော့ pop ၂ ခု လုပ်လိုက်တယ်။ ဒါကြောင့် 4,dog,True,8.4 stack ကနေ pop ၂ ခု လုပ်လိုက်တော့ 4,dog ဆိုတဲ့ stack ဖြစ်သွားပါတယ်။ Size က 2 ပြပါလိမ့်မယ်။

4.4 Implementing A Stack

အခု ကျွန်တော်တို့တွေ stack ထဲမှာ ဘာတွေ ပါမယ်ဆိုတာကို သိပြီးပါပြီ။ လကတွေ့ Stack class တစ်ခုကို တည်ဆောက်ကြည့်ရအောင်။

```

1 class Stack:
2     def __init__(self):
3         self.items = []
4     def is_empty(self):
5         return self.items == []
6     def push(self, item):
7         self.items.append(item)
8     def pop(self):
9         return self.items.pop()
10    def peek(self):
11        return self.items[len(self.items) - 1]
12    def size(self):
13        return len(self.items)

```

run

အဲဒီ code ထဲမှာ ပါတဲ့ `__init__(self)` ဆိုတာကတော့ constructor ပါ။ Object တစ်ခုကို စပြီး တည်ဆောက်တာ နဲ့ ဦးစွာ constructor ကို ခေါ်ပါတယ်။ stack မှာတော့ Object စ ဆောက်တာနဲ့ object ရဲ့ items variable ကို empty ထည့်လိုက်ပါတယ်။

`len(self.items)` ဆိုတာကတော့ len ဆိုတဲ့ function ဟာ item ရဲ့ array size ကို ဖော်ပြတာပါ။ item ထဲမှာ စုစုပေါင်း array အခန်း ဘယ် ၂ ခုကို ရှိတယ်ဆိုတာကို သိနိုင်ပါတယ်။

အခု ကျွန်တော်တို့ stack ကို စမ်းကြည့်ရအောင်။

```
1 class Stack:
2     def __init__(self):
3         self.items = []
4     def is_empty(self):
5         return self.items == []
6     def push(self, item):
7         self.items.append(item)
8     def pop(self):
9         return self.items.pop()
10    def peek(self):
11        return self.items[len(self.items) - 1]
12    def size(self):
13        return len(self.items)
14
15 s = Stack()
16
17 print(s.is_empty())
18 s.push(4)
19 s.push('dog')
20 print(s.peek())
21 s.push(True)
22 print(s.size())
23 s.push(8.4)
24 print(s.pop())
25 print(s.pop())
26 print(s.size())
27
28
```

run

```
True
dog
3
8.4
True
2
```

- The End -

Saturngod ရေးသား ပြုစုသည်။