Chapter-2

Syntax, Indentation, Comments and Statements

Contents

Chapter-2

Syntax, Indentation, Comments and Statements

Python ကို မူလက သင်ကြားမှု အထောက်အကူပြု programming language အဖြစ် ရည်ရွယ်၍ တီထွင်ခဲ့သည်။ သို့သော် အသုံးပြုရန် လွယ်ကူလွန်းခြင်းနှင့် သန့်ရှင်းသော syntax များကြောင့် လေ့လာခါစ သူများ သာမက ကျွမ်းကျင်သူများ(experts)က လက်ခံ အသုံးပြုကြသည်။

Python syntax များ သန့်ရှင်းမှု(cleanliness)ကြောင့် အချို့က Python code များကို "executable pseudocode" ဟု ခေါ်ဆိုခဲ့ကြသည်။ ကျွမ်းကျင်သူများ(experts)၏ အတွေ့အကြုံအရ Python script ကို ဖတ်ရန် လွယ်ကူပြီး အခက်အခဲမရှိ နားလည်နိုင်သည် ဟုလက်ခံကြသည်။

2. Syntax

Syntax ဆိုတာသတ်မှတ်ထားတဲ့ စည်းကမ်းချက်များ(a set of rules)ပါ။ python က တိတိကျကျ မှန်မှန်ကန်ကန် နားလည်အောင်လို့ ပရိုဂရမ် ရေးတဲ့အခါလိုက်နာရမဲ့ စည်းကမ်းချက်များပါ။ programming language တိုင်းမှာ code တွေကို ရေးတဲ့အခါ လိုက်နာရမဲ့ စည်းကမ်းတွေ ကိုယ်စီ ရှိပါတယ်။ မှန်ကန်အောင်ရေးမှ (syntactically correct ဖြစ်မှ) interpretator က အလုပ်လုပ်မှာပါ။

Syntax သည် programming language တစ်ခု၏ မလွဲမသွေ လိုက်နာရမည် စည်းမျဉ်း စည်းကမ်းများ ဖြစ်သည်။ Python code များကို မည်သို့ ရေးသားရမည်ကို သတ်မှတ်ပေးသည်။ Syntax သည် ဘာသာစကား၏ ဖွဲ့စည်းပုံကို ဆိုလိုသည်။(Syntax refers to the structure of the language)။ Syntax မှန်မှန်ကန်ကန်ဖြင့် ရေးထားသော ပရိုဂရမ်သာ ကောင်းစွာ အလုပ် လုပ်လိမ့်မည်။ Syntax ရေးသားပုံ စကားလုံးများနှင့် သင်္ကေတ များ၏ အဓိပ္ပာယ် နားလည်ထားရမည်။

Python ၏ ရိုးရှင်းသော syntax များကြောင့် လူသိများသည်။ သို့သော် လေ့လာခါစသူများအတွက် သို့မဟုတ် တခြားသော programming language တစ်ခုခုကို မလေ့လာခဲ့ဘူးသူများအတွက် syntax နှင့် ပတ်သက်၍ သိထားရမည့် အချက်များ ရှိသည်။ Python ကုဒ်တစ်ခုခုကို run ဘို့ ကြိုးစားနေချိန်မှာ SyntaxError တစ်ခုခုနှင့် မလွဲမသွေ ကြုံတွေ့ ရနိုင်သည်။ Python ခွင့်မပြုသည့်အရာများကို ပြုလုပ်မိရင် အမှားများနှင့် ကြုံတွေ့ရနိုင်သည်။ ထို့သို့ကြုံတွေ့လာနိုင်သည့် အမှားများကို ဖြေရှင်းနိုင်ရမည့် နည်းများကို ဖော်ပြထားသည်။

ဒီသင်ခန်းစာတစ်လျှောက်မှာ မမှန်ကန်တဲ့ syntax ဥပမာများအကြောင်းကို ရှင်းပြထားသည်။

2.1.1 Invalid Syntax in Python

Python ကုဒ်ကို run သည့်အခါ interpreter က Python byte ကုဒ်အဖြစ် ပြောင်းလဲရန် ပထမဦးဆုံး parse လုပ်သည်။ Parse လုပ်သည်ဆိုသည်မှာ interpreter က ခွဲခြမ်း စိတ်ဖြာပြီး အပိုင်းငယ်လေးများဖြစ်အောင် လုပ်သည်။(Parsing means to make something understandable (by analysing its parts).)။ ထိုသို့ parsing လုပ်နေချိန်တွင် အမှားတစ်ခုခု တွေ့ပါက invalid syntax ထုတ်ပေးလိမ့်မည်။ Python program ကို ပထမဆုံး

execution လုပ်သည့်အဆင့်ကို parsing stage ဟုခေါ်သည်။ ထိုကြောင့် syntax error ကို parsing error ဟုလည်း ခေါ်သည်။

Text parsing ဆိုသည်မှာ စာလုံးများ၊ စာကြောင်းများကို စည်းမျဉ်းများအတိုင်း အက္ခရာများ သို့မဟုတ် တန်ဖိုးများ(စာသားများ)အဖြစ် အစဉ်လိုက် ခွဲခြားသည်။ (Text parsing is a common programming task that splits the given sequence of characters or values (text) into smaller parts based on some rules.)

Python ကိုလေ့လာနေစဉ် ပထမဆုံးအကြိမ် SyntaxError ရရှိပါက စိတ်ရှုပ်ထွေး နိုင်သည်။ Code ၏ မည်သည့်နေရာတွင် syntax ဖြစ်နေသည်ကို Python interpreter က ညွှန်ပြပေးသည်။ ထိုသို့ interpreter က ညွှန်ပြပေးသည့် traceback သည် အနည်းငယ်ရှုပ်ထွေးနိုင်သည်။ တစ်ခါတစ်ရံ တိတိကျကျ မှားနေသည့်နေရာကို ညွှန်ပြပေးသည်။ ထိုသို့ Python code တစ်နေရာရာတွင် interpreter က invalid syntax ဖြစ်နေလျှင် SyntaxError exception ထုတ်ပေး(raise)သည်။ အမှားပြင်ရန်၊ debug လုပ်ရန် အကူအညီပေးသည့် အနေဖြင့် traceback ထုတ်ပေးသည်။ ဥပမာ

```
ages = {
    'pam': 24,
    'jim': 24
    'michael': 43 ←
}
print(f'Michael is {ages["michael"]} years old.')
```

```
File "<ipython-input-6-37a3e16dc343>", line 5
    'michael': 43
             ^
SyntaxError: invalid syntax
```

Line 4 တွင် ကော်မာ မပါသောကြောင့် invalid syntax ဖြစ်ပေါ်နေသည်။ dictionary syntax အရ dictionary တစ်ခုသည် တွန့်ကွင်း အတွင်း၌ key နှင့် value များကို ကော်မာခံ၍ ရေးရမည်။ SyntaxError နှင့် တူညီပြီး နာမည်ကွဲသည့် error နှစ်မျိုးမှာ IndentationError နှင့် TabError တို့ ဖြစ်သည်။

2.1.2 Syntax errors

Syntax error တွေက ဘာတွေလဲဆိုတာကို သိသည်နှင့် တပြိုင်နက် ဖြေရှင်းရန် လွယ်ကူသည်။ သို့သော် အမှားမက်ဆေ့ချ်(error message) အများစုမှာ ဖြေရှင်းရန် အသုံးဝင်သောအရာများ မဟုတ်ကြပါ။ အများဆုံး တွေ့ရလေ့ရှိသည့် SyntaxError မှာ “SyntaxError: invalid syntax”နှင့် SyntaxError: invalid token တို့ဖြစ်သည်။

အများဆုံး တွေ့ရလေ့ရှိသည့် syntax အမှားများကို ရှောင်ရှားရန် နည်းလမ်းများမှာ

(၁) Python keyword များကို variable name အဖြစ် လုံးဝ အသုံးမပြုပါနှင့်။

(၂) `for`, `while`, `if`, and `def` စသည့် statement များ၏ အဆုံးတွင် colon မကျန်ခဲ့စေရန် အမြဲ သတိထားပါ။

(၃) Indentation လုပ်ထားသည့် အကွာအဝေး ညီညာမှုရှိအောင်(consistent ဖြစ်အောင်) ဂရုစိုက်ပါ။ Spaces သို့မဟုတ် tab ကြိုက်သည့်အရာကို သုံးနိုင်ပါသည်။ သို့သော် နှစ်မျိုးစလုံးကို ရောမသုံးပါနဲ့။

(၄) String များကို quotation mark ထဲတွင် ထည့်ရေးပါ။ အဖွင့်နှင့် အပိတ်တို့တွင် single quote သို့မဟုတ် double quote ကို တူညီအောင် ရေးပါ။

(၅) Multiline string များတွင် triple quote သုံးသည့်အခါ အဖွင့်နှင့် အပိတ်တို့၌ တူညီအောင် ရေးပါ။

(၆) - (, {, သို့မဟုတ် [စသည့် ကွင်းများဖြင့် အဖွင့်ကွင်း ရေးပြီး အပိတ်ကွင်း မေ့ကျန်ခဲ့လျှင် `invalid token` error ပေးလိမ့်မည်။

(၇) Assignment operator (=) နှင့် conditional `==` တို့ကို မှန်ကန်အောင် သုံးပါ။

End-of-Line Terminates a Statement

Statement တစ်ခု၏ အဆုံးတွင် semicolon (`;`) ထည့်ပေးရန် မလိုပါ။ C နှင့် C++ တို့တွင် statement တစ်ခု၏ အဆုံးတွင် semicolon (`;`) နှင့် အဆုံးသတ်ပေးရသည်။

```
midpoint = 5
```

Semicolon Can Optionally Terminate a Statement

စာကြောင်းတစ်ကြောင်းစာ နေရာတွင် statement နှစ်ခု သုံးခု(multiple statements) ရေးလိုသည့်အခါ semicolon (`;`)ကို အသုံးပြု၍ ရေးနိုင်သည်။ ဥပမာ-

```
lower = []; upper = []
```

အပေါ်ကကဲ့သို့ စာကြောင်း တစ်ကြောင်းစာ နေရာတွင် statement တစ်ကြောင်းစီရေးမည်အစား statement နှစ်ကြောင်းကို အတူတကွ ရေးနိုင်သည်။

```
lower = []
upper = []
```

စာကြောင်း တစ်ကြောင်းစာ နေရာတွင် statement များစွာကို semicolon (`;`) ခြား၍ ရေးသားခြင်းကို အားမပေးပါ။ တစ်ခါတစ်ရံ သင်ကြားပြသလို၍သာ ရေးသားဖော်ပြခြင်း ဖြစ်သည်။

Indentation: Whitespace Matters!

Python တွင် space (ဘာမှ မပါသည့် နေရာလွတ်, Whitespace) တွေကို ထားချင်သလိုထားပြီး ရေးလို့ မရပါ။ Indentation အကြောင်းကို အသေးစိတ် ရှင်းပြထားသည်။

အောက်တွင် Indentation လုပ်ပုံကို ဖော်ပြထားသည်။

```
for i in range(10):
        if i < midpoint:
                lower.append(i)
        else:
                upper.append(i)
```

Loop နှင့် conditional statement တို့ပါဝင်သည့် compound control-flow statement တစ်ခု ဖြစ်သည်။ Whitespace ကို သုံး၍ indentation ဖြင့် flow control လုပ်သည်။ ထို့ကြောင့် Python syntax တွင် whitespace သည် အလွန်ရေးကြီးသည့် အရာ ဖြစ်သည်။

Whitespace Within Lines Does Not Matter

စာကြောင်းအတွင်း၌ ရှိသည့် Whitespace များသည် ဘာအဓိပ္ပာယ်မှ မဆောင်ပါ။ အောက်က ဥပမာတွင် code များ ရေးပုံ မတူညီသော်လည်း တူညီသည့် ရလဒ်ကို ပေးသည်။

```
x=1+2
x = 1 + 2
x          =      1    +        2
```

Code ဖတ်ရှုနိုင်မှု(readability) ပိုကောင်းစေရန် စာကြောင်းအတွင်း၌ ရှိသည့် whitespace ကြိုက်သလို ထားပြီး အဆင်ပြေသလို ရေးနိုင်သည်။ Whitespace ကို သုံး၍ ဖတ်ရှုရန် လွယ်ကူသည့် code များ ရေးသားနိုင်ရန် ကြိုးစားသင့်သည်။ လက်သည်းကွင်း(parentheses), လေးထောင့်ကွင်း(brackets), တွန့်ကွင်း (braces) စသည့် အဖွင့်ကွင်း ပြီးနောက် space တစ်နေရာစာ မခြားသင့်ပါ။

```
spam(ham[1], {eggs: 2})          # Correct
spam( ham[ 1 ], { eggs: 2 } )    # Wrong
```

နောက်ဆုံးကော်မာ(trailing comma)နှင့် ကွင်းပိတ်(close parenthesis)အကြားတွင် space တစ်နေရာ မခြားသင့်ပါ။

```
foo = (0,)                       # Correct
bar = (0, )                      # Wrong
```

Function တစ်ခုကို ခေါ်(call) သုံးသည့်အခါ function နာမည်နှင့် အဖွင့်ကွင်း(open parenthesis) တစ်ခု အကြားမှာ space တစ်နေရာစာ မခြားသင့်ပါ။

```
spam(1)       # Correct
spam (1)      # Wrong
```

Indexing သို့မဟုတ် slicing လုပ်သည့်အခါ အဖွင့်ကွင်း(open parenthesis) အရှေ့တွင် space တစ်နေရာ မခြားသင့်ပါ။

```
dct['key'] = lst[index]          # Correct
dct ['key'] = lst [index]        # Wrong
```

assignment operator (=) ကို အောက်တွင် ဖော်ပြထားသကဲ့သို့ ညီညီညာညာဖြစ်အောင် မည်သင့်ပါ။

```
# Correct:
x = 1
y = 2
long_variable = 3
```

```
# Wrong
```

```
x             = 1
y             = 2
long_variable = 3
```

Parentheses Are for Grouping or Calling

လက်သည်းကွင်း(parentheses)ကို အစုဖွဲ့ရန်(grouping)နှင့် functionများကို ခေါ်သုံးရန်(being called) ရန်အတွက် အသုံးပြုသည်။ အဖွင့်နှင့် အပိတ်ကွင်း(opening and closing parentheses)အကြားတွင် argument များ ထည့်သွင်း ရေးသားသည်။

```
2 * (3 + 4)
```

2.2 Python Indentation

C, C ++ နှင့် Java ပရိုဂရမ်များတွင် ကုဒ်အစုအဝေး(block of code)တစ်ခုအဖြစ် သတ်မှတ်ရန်(define လုပ်ရန်) သို့မဟုတ် တွန့်ကွင်း(brace {})ကို အသုံးပြုကြသည်။ Python တွင် indentation ကို အသုံးပြု၍ ကုဒ်အစုအဝေး(block of code)တစ်ခုအဖြစ် ပြုလုပ်(define လုပ်)သည်။

Function တစ်ခု သို့မဟုတ် loop ကို ရေးသည့်အခါ ကုဒ်တစ်ကြောင်းတည်းဖြင့် ရေး၍ မရပါ။ ထို့ကြောင့် ကုဒ် နှစ်ကြောင်း၊ သုံးကြောင်းပါသည့် ကုဒ်အစုအဝေး(block of code)ဖြင့် function တစ်ခု သို့မဟုတ် loop တစ်ခုကို တည်ဆောက် ရေးသားရသည်။

Indentation သည် code line ၏ အစနေရာတွင် ရှိသော space များ ဖြစ်သည်။(Indentation refers to the spaces at the beginning of a code line.)။ Indentation ကို အခြား programming language များတွင် code အတွင်းရှိ စာများကို အလွယ်တကူ ဖတ်ရှုနိုင်ရန်အတွက်သာ သုံးသည်။ readability အတွက်သာ သုံးသည်။ သို့သော် Python တွင် indentation သည် အလွန်အရေးကြီးသည်။ Python ၏ class, function, conditional နှင့် loop စသည့် တည်ဆောက်မှုများတွင် indentation ကိုသုံးသည်။

```
if True:
        # execute this block of statements
        print("Block 1")
else:
        # execute other block of statements
        print("Block 2")
```

For loop တစ်ခုကို ဥပမာပေး၍ indentation အကြောင်းကို ရှင်းပြမည်။ For loop တစ်ခု၏ ပထမဆုံး စာကြောင်းသည် indentation မလုပ်ထားသည့် ပုံမှန် ဘေးမျဉ်းဘေးတွင် ကပ်ရေးသည့် စာကြောင်း ဖြစ်သည်။ indentation မလုပ်ထားသည့် စာကြောင်းကို unindented line ဟု ခေါ်သည်။

ပထမ စာကြောင်းမှ လွဲ၍ for loop အတွင်းရှိ ကျန်စာကြောင်းများ အားလုံးကို indentation ဖြင့် ရေးရသည်။ တူညီသည့် indentation ရှိသည့် စာကြောင်းများ အားလုံးသည် for loop တွင် ပါဝင်သည်ဟု အဓိပ္ပာယ် သက်ရောက် သည်။ indentation ရေးပုံမှာ ဘေးမျဉ်းမှ whitespace လေးခု ခွာ၍ ရေးခြင်း ဖြစ်သည်။

တူညီသည့် indentation ရှိသည့် စာကြောင်းများ အားလုံးသည် သက်ဆိုင်သည့် ကုဒ်အစု အဝေး(block of code)တစ်ခုတွင် ပါဝင်သည်။

ယေဘုယျအားဖြင့် whitespace လေးခုကို indentation အတွက် အသုံးပြုသည်။ Whitespace လေးခုအစား tabs ကို သုံးခြင်းသည် ပိုကောင်းသည်။ အောက်တွင် for loop အတွက် indentation လုပ်ပုံ နှင့် if statement အတွက် indentation လုပ်ပုံတို့ကို အောက်တွင် ဖော်ပြထားသည်။

```
for i in range(1,11):
    print(i)
    if i == 3:
        break
```

```
1
2
3
```

Python တွင် indentation ဖြင့် ရေးထားသောကြောင့် code ကိုသပ်ရပ်သန့်ရှင်းစေသည်။ ထို့အပြင် Python program များ၏ code မှာ တူညီပြီး တသမတ်တည်း ဖြစ်နေသည်။ Line continuation လုပ်သည့်အခါ indentation စည်းကမ်းနှင့် မသက်ဆိုင်တော့ပေ။ သို့သော် အလေ့အကျင့်ကောင်းအဖြစ် line continuation လုပ်သော်လည်း indentation ဖြင့်ရေးသားခြင်းဖြင့် code များ ပို၍ ဖတ်ရ လွယ်ကူစေသည်။

```
if True:
    print('Hello')
    a = 5
```

```
Hello
```

```
if True: print('Hello'); a = 5
```

အပေါ်မှ if statement ကုဒ်နှစ်မျိုးလုံးသည် မှန်ကန်(valid) ကြသော်လည်း ပထမကုဒ်သည် ရှင်းလင်းပြီး ဖတ်ရန်လွယ်ကူသည်။ Indentation စည်းကမ်းများကို မလိုက်နာဘဲ ရေးထားသည့် ကုဒ်များပါခဲ့လျှင် Python interpreter က IndentationError ထုတ်ပေးလိမ့်မည်။

Python တွင် ကုဒ်များကို အတူတကွ execute လုပ်ရန် အတွက် whitespace ကိုအသုံးပြုသည်။ အတူတကွ execute လုပ်ရမည့် ကုဒ်လိုင်းများကို block သို့မဟုတ် code block ဟုခေါ်သည်။ အတူတကွ execute လုပ်ရမည့် ကုဒ်လိုင်းများဖြစ်ကြောင်းကို whitespace များထာသည့် အကွာအဝေး သို့မဟုတ် Indentation ဖြင့် ဖော်ပြသည်။ တစ်နည်းအားဖြင့် indentation တူလျှင်(equal amount of indentation) ထိုကုဒ်လိုင်းများကို အတူတကွ execute လုပ်ရမည်။

2.2.1 Python program flow

တခြားသော programming language များကဲ့သို့ပင် Python တွင် ပရိုဂရမ် တစ်ခု၏ execution ကို ထိန်းချုပ်နိုင်သည့် နည်းလမ်းများစွာရှိသည်။ control statement ကိုသုံး၍ execution ကို ထိန်းချုပ်နိုင်သည်။

- Program flow
- Control statement

Program flow ဆိုသည်မှာ code များ မည်ကဲ့သို့ မည်သူက အရင် မည်သူကနောက် execute လုပ်ရမည်ကို ဖော်ပြထားခြင်း ဖြစ်သည်။ ဆောင်ရွက်ရမည့် ဦးစားပေး(priority) အဆင့်များကိုလည်း Program flow တွင် ဖော်ပြထားသည်။

Python သည် ရိုးရှင်းသည့် အပေါက်မှ အောက်သို့(top-down)ဆင်းသည့် program flow အသုံးပြုထားသည်။ ပရိုဂရမ် အပေါ်ဆုံး အကြောင်းမှ အောက်ဆုံး အကြောင်းအထိ execute လုပ်ပြီးဆင်းလာသည့် program flow ဖြစ်သည်။ အပေါ်က code လိုင်းများ execute လုပ်နေချိန်တွင်အောက်မှ code လိုင်းများ ဘာမှ မလုပ်ဘဲ စောင့်နေရမည်။ ပထမ code လိုင်း execute လုပ်ပြီးမှ ဒုတိယ code လိုင်း execute လုပ်ရသည်။ ဒုတိယ code လိုင်း execute လုပ်ပြီးမှ တတိယ code လိုင်း စသည်ဖြင့် အဆင့်ဆင့် execute လုပ်ရသည်။

ဤသို့ အပေါ်မှအောက်သို့ စီးဆင်းမှု(top-down program flow)သည် Python ပရိုဂရမ်များကို လွယ်ကူစွာ နားလည်စေနိုင်ပြီး အခက်အခဲမရှိ debug လုပ်နိုင်သည်။ code ကို မြင်ရုံဖြင့် လှမ်းကြည့်ခြင်းဖြင့် ပရိုဂရမ်တစ်ခု၏ သဘောကို ခန့်မှန်းနိုင်သည်။

Top-down နည်းဖြင့် ရေးသည့်အခါ ဖြေရှင်းရမည့်ပြဿနာကို module ငယ်များ ဖြစ်အောင် ခွဲခြားရသည်။ Module ငယ်များသည် ဖြေရှင်းရမည့်ပြဿနာ၏ တစိတ်တပိုင်းကို တာဝန်ယူရသည်။ ထို module ငယ်များ တစ်ခုနှင့် တစ်ခု ဆက်စပ်ပြီး ပြဿနာကြီးတစ်ခုလုံးကို ဖြေရှင်းရသည်။

2.2.2 Python Statement

Python interpreter ထဲသို့ အလုပ်တစ်ခု လုပ်စေရန်အတွက် ညွှန်ကြားချက်(instruction)ကို statement ဟုခေါ်သည်။ ဥပမာအားဖြင့် a = 1 သည် assignment statement တစ်ခုဖြစ်သည်။ a ဟုသည့် ထည့်စရာ နေရာတွင် 1 ကို ထည့်ထားသည်။ တစ်နည်းအားဖြင့် a ဆိုသည့် memory နေရာတွင် 1 ကို assign လုပ်သည့် ညွှန်ကြားချက်(instruction) ဖြစ်သောကြောင့် assignment statement ဟုခေါ်ဆိုခြင်း ဖြစ်သည်။

ဥပမာ- if statement, for statement, while statement, စသည့် statement ရှိသည်။ If statement သည် ဆုံးဖြတ်ချက် တစ်ခုခုကို ချမှတ်ပေးရန် ညွှန်ကြားချက်(instruction) ဖြစ်သည်။ For statement သည် ထပ်ခါထပ်ခါ လုပ်ရမည့်ကိစ္စများကို ကုဒ်လိုင်းအနည်းငယ်ဖြင့် for loop ပတ်ပြီး ဆောင်ရွက်ပေးရန် ညွှန်ကြားချက်(instruction) ဖြစ်သည်။ While statement သည် ထပ်ခါထပ်ခါ လုပ်ရမည့် ကိစ္စများကို ကုဒ်လိုင်း အနည်းငယ်ဖြင့် while loop ပတ်ပြီး ဆောင်ရွက်ပေးရန် ညွှန်ကြားချက်(instruction) ဖြစ်သည်။

2.2.3 Multi-line statement

တချို့ statement(ကုဒ်တစ်ကြောင်း)သည် အလွန်ရှည်လျားသဖြင့် စာကြောင်း (၂)ကြောင်း ၊ (၃) ကြောင်း ရေးရန်လိုအပ်သည်။ ထိုအခါ line continuation character (\)ကို အသုံးပြုသည်။ Line continuation character (\) ကို newline character ဟူ၍လည်း ခေါ်သည်။ Python မှာ statement ၏အဆုံးတွင် newline character ထည့်၍ အမှတ်အသား ပြုထားသည်။

\ သည် ကုဒ်များကို တစ်ကြောင်း ဖြစ်အောင် ဆက်ပေးသည်။

\ သည် line continuation character (\) ဖြစ်သည်။

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
a
```

```
File "<ipython-input-1-b14d338f1d6e>", line 1
a = 1 + 2 + 3 + \ # \ သည် ကုဒ်များကို တစ်ကြောင်း ဖြစ်အောင် ဆက်ပေးသည်။
SyntaxError: unexpected character after line continuation character
```

လက်သည့်ကွင်း(parentheses ( )) ၊ လေးထောင်ကွင်း(brackets [ ]) နှင့် တွန့်ကွင်း(braces { }) တို့ အတွင်း၌ ထည့်ရေးလျှင် line continuation character () ကို အသုံးပြုရန် မလိုအပ်ပါ။ ထိုကွင်းများ ( ) ,[ ], { } တွင်း၌ ထည့်ရေးလျှင် ကုဒ်တစ်ကြောင်းတည်းဟု အဓိပ္ပာယ်သက်ရောက်သည်။ ဥပမာ- လက်သည့်ကွင်း( )တွင်း၌ ထည့်ရေးလျှင် newline character မလိုအပ်ပါ။

```
a = (1 + 2 + 3 + 4 + 5
     + 6 + 7 + 8 + 9)
a
```

( ) တွင်း၌ ထည့်ရေးလျှင် newline character မလိုအပ်ပါ။

```
colors = ['red','blue',
          'green']
colors
```

```
['red', 'blue', 'green']
```

semicolons ကို သုံး၍ code line များစွာကို စာကြောင်းတစ်ကြောင်းတည်းတွင် ထားရေးနိုင်သည်။

စာကြောင်းတစ်ကြောင်းနေရာတွင် semicolon ခံ၍ statement (၃)ခု ရေးနိုင်သည်။

```
a = 1; b = 2; c = 3
```

2.3 Python Comments

ပရိုဂရမ်တစ်ခုတွင် comment အလွန်အရေးကြီးသည်။ သူတို့က ပရိုဂရမ်တစ်ခုအတွင်း ဘာတွေ ဖြစ်နေသည်ကို ဖော်ပြသည့် ရှင်းပြချက်များ ၊ ကုဒ်တွေကို ဘယ်လိုရေးသားထားသည်ဆိုသည့် ရှင်းပြချက်များ ဖြစ်သည်။ တခြားသောသူ တစ်ယောက်ယောက် comment တွင် ရေးသားထားသည့် ရှင်းပြ ချက်များကို ဖတ်ရှုပြီး ပရိုဂရမ်ကို အလွယ်တကူ နားလည်နိုင်သည်။ သင်ကိုယ်တိုင်လည်း လွန်ခဲ့သည့် တစ်လ အတွင်း သင်ရေးခဲ့သော ပရိုဂရမ်၏ အဓိက အသေးစိတ်အချက်များကို သင်မေ့သွားနိုင်သည်။ ထိုအခါ comment များက အထောက်အကူ

ပေးပါလိမ့်မည်။ ထို့ကြောင့် မည်ကဲ့သို့ ရေးခဲ့သည် မည်သည့် လောဂျစ်ကို သုံးခဲ့သည် စသည်တို့ကို မှတ်ချက်ပုံစံဖြင့် ရှင်းပြရန် အချိန်ယူခြင်းသည် အမြဲပင်အကျိုးရှိသည်။

Comments များသည် ပရိုဂရမ်ထဲ၌ ပါဝင်သည့်အချက်များ(intent of program) နှင့် လုပ်ဆောင်ချက်များ (functionality of program)ကို ပိုနားလည်အောင် အထောက်အကူ ပေးနိုင်သည်။

Comment ကိုအသုံးပြုခြင်း၏အားသာချက်များ ကုဒ်ကို ပိုမိုနားလည်စေသည်။ ပရိုဂရမ်ကို ဖတ်လို့လွယ်ကူအောင်(more readable)လုပ်ပေးသည် ။ ကုဒ်အချို့ကို ဘာကြောင့် ထိုသို့ ရေးရသည် ဆိုသည့် အကြောင်းပြချက်များကို သိနိုင်သည်။ အချို့သော Comment များ program ၏ pseudo-code များ ဖြစ်ကြသည်။

Python တွင် comment statement တစ်ခုကို သင်္ကေတ(hash, #) ဖြင့် စာကြောင်းအစတွင် စတင်ရေးသားသည်။ Python Interpreter ကို (hash (#) သင်္ကေတဖြင့် စထားသည့် comment statement များကို လစ်လျူရှုသည်။ comment statement များ အလုပ် လုပ်လိမ့်မည် မဟုတ်ပေ။ စာကြောင်း တစ်ကြောင်းတည်းသာ ပါသည့် comment onf stand-alone comment ဖြစ်သည်။

```
#This is a comment
#print out Hello
print('Hello')
```

```
Hello
```

```
x += 2  # shorthand for x = x + 2
```

2.3.1 Inline Comments

Inline comment ဆိုသည်မှာ statement နှင့် အတူ ထည့်ရေးသည့် comment ဖြစ်သည်။ Inline comment ရေးသည့်အခါ အနည်းဆုံး space (၂)နေရာ၊ (၃) နေရာစာ ချန်ထားသင့်သည်။ # နှင့် စပြီးနောက် space တစ်နေရာစာ ချန်ထားပြီးနောက် မိမိ ရေးချင်သည့် comment ကို ရေးနိုင်သည်။

Comment များသည် ပြည့်စုံသော စာကြောင်းများ ဖြစ်သင့်သည်။ Identifier မဟုတ်ခဲ့လျှင် ပထမစာလုံးကို စာလုံးအကြီးဖြင့် ရေးသားသင့်သည်။ အင်္ဂလိပ်စကား မပြောသောနိုင်ငံများမှ Python ပရိုဂရမ်မာများ (coder များ)လည်း အင်္ဂလိပ်လို comment ရေးပါ။

2.3.2 Multi-line comments

စာကြောင်းတစ်ကြောင်းစာထက် ပိုများသည့် comment statement များ အတွက် စာကြောင်းတိုင်း၏အစတွင် hash (#) symbol ကိုသုံး၍ ရေးနိုင်သည်။ ဥပမာ-

```
# This is a long comment
# and it extends
# to multiple lines
```

Another way of doing this is to use triple quotes, either ''' or """အခြားနည်းလမ်းတစ်ခုမှာ multi-line string များအတွက် triple quotes ကို အသုံးပြု၍ ရေးနိုင်သည်။ ' ' ' သို့မဟုတ် " " " ကို အသုံးပြုနိုင်သည်။

```
"""This is also a
perfect example of
multi-line comments"""
```

```
'This is also a\nperfect example of\nmulti-line comments'
```

Triple quote အသုံးပြု၍ ရေးထားသည့် Multi-line Comment တစ်ခုကို အောက်တွင် ဖော်ပြထားသည်။

```
'''
I am a
multiline comment!
'''
print("Hello World")
```

```
Hello World
```

3.3 Comment များကို ပိုကောင်းအောင် ရေးနည်း

လုပ်ဆောင်ချက်(function)ကို ဖော်ပြသည့် Comment မျိုးရေးပါ။ မည်ကဲ့သို့ လုပ်ဆောင်သည်ကို အသေးစိတ် ရေးရန် မလိုပါ။ မလိုအပ်သော မှတ်ချက်များကို တတ်နိုင်သမျှ ဖယ်ရှားရန်ကြိုးစားပါ။

ဖြစ်နိုင်လျှင် ပိုကောင်းသည့် function name သို့မဟုတ် variable name ကို ရွေးချယ် အသုံးပြုပြီး သူ့ဟာသူရှင်းပြနိုင်တဲ့ ကုဒ်မျိုးရေးပါ။ တစ်နည်းအားဖြင့် ပေးချင်တဲ့ ပေးပြီး ကုဒ်ကို ရေးချင်သလိုရေးပြီး comment တွေနဲ့ လိုက်ရှင်းပြတာမျိုး မဖြစ်သင့်ပါ။ မှတ်ချက်များကို တတ်နိုင်သမျှ အတိုဆုံးနှင့် အနှစ်ချုပ် ရေးရန် ကြိုးစားပါ။

2.4 Docstrings in Python

Docstring သည် documentation strin ၏ အတိုကောက်ဖြစ်သည်။ Docstring ကို function, method, class, module သည်တို့၏ definition အပြီးတွင် ထည့်သွင်းရေးသားထားလေ့ရှိသည်။ Triple quotes ဖြင့် docstrings များကို ရေးသားထားလေ့ရှိသည်။

Python Moduleတစ်ခု၏ docstring သည် module တစ်ခုကို import လုပ်လိုက်သည့်အခါ ရရှိနိုင်သည့် (available) classes, functions, objects and exceptions များ၏ စာရင်းဖြစ်သည်။ တစ်ကြောင်းတည်းဖြင့် ရေးထားသည် အနှစ်ချုပ်(one-line summary)လည်း ပါဝင်ရမည်။ docstring ကို Python ဖိုင်ရဲ့အစမှာ ရေးထားရမည်။ ဥပမာ: Python ရှိ builtin module အတွက် pickstrings ကိုကြည့်ကြစို့။

Docstrings of Python module တစ်ခုဖြစ်သည့် pickle module ၏ docstrings ကို အောက်တင် ဖော်ပြထားသည်။

```
In [11]:
```

```
import pickle
print(pickle.__doc__)
```

```
Create portable serialized representations of Python objects.
See module copyreg for a mechanism for registering custom picklers.
      See module pickletools source for extensive comments.
      Classes:
          Pickler
          Unpickler
      Functions:
          dump(object, file)
          dumps(object) -> string
          load(file) -> object
          loads(string) -> object
      Misc variables:
          __version__
          format_version
          compatible_formats
```

```
import math
print(math.__doc__)
```

```
      This module provides access to the mathematical functions
      defined by the C standard.
```

```
def double(num):
    """Function to double the value"""
    return 2*num
```

Docstrings appear right after the definition of a function, class, or a module. This separates docstrings from multiline comments using triple quotes.

The docstrings are associated with the object as their **doc** attribute.

So, we can access the docstrings of the above function with the following lines of code:

```
def double(num):
    """Function to double the value"""
    return 2*num
print(double.__doc__)
```

```
      Function to double the value
```

2.4.1 Docstrings for Python Functions¶

function တစ်ခု သို့မဟုတ် method အတွက် docstring သည် ၎င်းက လုပ်ဆောင်နိုင်သည်အချက်များကို အတိုချုံးဖော်ပြထားသည်။ arguments, return values, exceptions နှင့် optional argument များကိုလည်း ဖော်ပြထားသည်။

```
def add_binary(a, b):
    '''
    Returns the sum of two decimal numbers in binary digits.

            Parameters:
                    a (int): A decimal integer
                    b (int): Another decimal integer

            Returns:
                    binary_sum (str): Binary string of the sum of a and
 b
    '''
    binary_sum = bin(a+b)[2:]
    return binary_sum


print(add_binary.__doc__)
```

```
Returns the sum of two decimal numbers in binary digits.

        Parameters:
            a (int): A decimal integer
            b (int): Another decimal integer
        Returns:
            binary_sum (str): Binary string of the sum of a and b
```

2.4.2 Docstrings for Python Classes¶

Class များ၏ docstring ထဲတွင် public methods နှင့် instance variable များကို ဖော်ပြသည့်စာရင်း ပါဝင်သည်။ subclasse များ, constructor များ နှင့် method များ တွင်လည်း ကိုယ်ပိုင် docstring များရှိရမည်။

- End-