Chapter – 19 Pythonic Code

Contents

Chapter – 19 Pythonic Code

Programming ဆိုင်ရာ အခေါ်အဝေါ်အသစ်ဖြစ်သည့် Pythonic ဟူသော ဝေါဟာရကို မကြာခဏ သင်ကြားဖူးနေပေမည်။ Python စတိုင်နှင့် ကုဒ်များကို ရေးသားခြင်းကို Pythonic ဟု ပြောနိုင်သည်။ Pythonic ဆိုတဲ့ အသုံး အနှုန်းက ဘာကိုဆိုလိုတာလဲ။ Python စတိုင်နှင့် code ကို ဘယ်လိုရေးရမလဲ၊ ဆိုတာကို လေ့လာဖို့ စိတ်ဝင်စားမိတယ် ဆိုရင် သင်ဟာ AI နည်းပညာတွေကို စိတ်ဝင်စားနေတယ် ဆိုသေချာပါတယ်။

19.1 Pythonic ဆိုတာ ဘာလဲ?

Python စတိုင်လ်နဲ့ရေးထားတဲ့ code ပါ။ Pythonic code ဆိုတာ Python language design နဲ့လိုက်ဖက်တဲ့ကုဒ် ပါ။(Pythonic code is code that fits well with the design of the Python language.)။ လူတွေက Pythonic code အကြောင်းပြောတဲ့အခါ သူတို့က code က Python idiom ကို ကောင်းကောင်း သုံးထားတယ်။ Python စတိုင်နှင့် code ကို ဘယ်လိုရေးထားတယ်လို့ ပြောကြလေ့ရှိပါတယ်။ Pythonic သည် Python ကျွမ်းကျင်သူများက လက်ခံအသုံးပြုသော အသုံးအနှုန်းသည့် စကားလုံး ဖြစ်သည်။

Pythonic Code ရေးသားခြင်း Pythonic နည်းဖြင့် code များကို ရေးသားရန် သတ်မှတ်ထားသော နည်းလမ်းများ ရှိ မရှိ သင်သိချင်ပေလိမ့်မည်။ PEP 8 တွင် Pythonic နည်းဖြင့် code များကို ရေးသားရန် လမ်းညွှန်ချက်များ ရှိသည်။ Programming language တိုင်းလိုလိုတွင် ပရိုဂရမ်များက လက်ခံထားသည့် စတိုင်လ်နှင့် သမာရိုးကျ လမ်းညွှန်ချက်များ (stylistic and conventional guidelines) ရှိကြသည်။

ဂိုက်လိုင်းများမှာ variable, class နှင့် module တွေကို ဘယ်လိုမျိုး သင့်လျော် နားလည်လွယ်သည့် အမည်ပေးရမယ်ဆိုတွေ၊ looping structure နှင့် code လိုင်းများကို ဘယ်လို စနစ်တကျ အထားအသို ပုံတွေနဲ့ ရေးသားရမယ်ဆိုတာတွေ ပါဝင်သည်။ ထိုဂိုက်လိုင်းများနှင့် ကိုက်ညီအောင်ရေးသားခြင်းသည် “ Pythonic” ဖြစ်သည့် အချက်တစ်ခုဖြစ်သည်။

Pythonic code ကို ဘာကြောင့်ရေးသင့်တာလဲ။ Python ပရိုဂရမ်၏ အလုပ်လုပ်သည့် အမြန်နှုန်းသည် တခြားသော programming language ထက်စာလျှင် မြန်နှုန်း နိမ့်သည်။ Idiomatic pythonic code ဖြင့် python ပရိုဂရမ်များ၏ အမြန်နှုန်းကို ပိုမြန်စေသည်။ PEP 8 ရှိနေပေမယ့် Pythonic ဆိုတဲ့ဝေါဟာရက လူအမျိုးမျိုးအတွက် ကွဲပြားကြသည်။ ရှင်းလင်းပြတ်သားမှု(Clarity)ရှိသည့် efficient ဖြစ်သည့် code များကိုသာ ရေးရန် အာရုံစိုက်ပြီး တခြားသူများ Pythonic ကုဒ်ကိုရေးသည်၊ မရေးသည် လိုက်ဆန်းစစ် နေစရာ မလိုပါ။

Python ၏ နှစ်သက်စရာကောင်းသည့် အချက်တစ်ခုမှာ code များကို ဖတ်ရှုရန် အလွန် လွယ်ကူခြင်း (high level of readability)ဖြစ်သည်။ Python ကို ဘာကြောင့်များ ကြိုက်သလဲလို့ မေးရင် အင်္ဂလိပ်စာဖတ်ရသလို နားလည်လွယ်သည်ဟု မကြာခဏ ကြားရလေ့ရှိသည်။ Python code များကို Code Style guideline များ အတိုင်း ရေးသားထားခြင်းနှင့် “Pythonic” idiom များကို သုံး၍ ရေးသား ထားခြင်း တို့ကြောင့် ဖတ်ရလွယ်ကူခြင်း ဖြစ်သည်။ PEP 8 သည့် Python Code များကို Pythonic စတိုင်ဖြင့် ရေးသားသည့်အခါ အသုံးပြုလေ့ရှိသည် ညွှန်ကြားချက်(Style Guide for Python Code)ဖြစ်သည်။

Python ကို အသုံးပြု၍ ရှင်းလင်းပြတ်သားစွာ ရေးထားသည့်ကုဒ်များ ကုဒ်များကို ရှင်းလင်းပြတ်သား (Clarity)စွာ ရေးသားရမည်။ အကယ်၍ သင်သည် developer တစ်ယောက် ဖြစ်လိုပါက သင်၏ code ၏ရှင်းလင်း ပြတ်သားမှုသည် သင်၏ အောင်မြင်မှုအတွက် အဓိက ဖြစ်သည်။ ကုဒ်ကို ညံ့ဖျင်းစွာ ရေးသားထားပါက အိပ်မပျော် သည့်ညများစွာ ကြုံတွေ့ရလိမ့်မည်။

Python သင်ခန်းစာကို ဖတ်ခြင်းဖြင့် data structure များအကြောင်းကိုပါ တပြိုက်နက် လေ့လာ နိုင်သည်။ အောက်တွင် ပုံမှန် ရေးနေကြနည်းဖြင့် ရေးထားသည့် code များ ဖြစ်သည်။

ဥပမာ - numbers ဆိုသည့် list ထဲမှ ကိန်းများကို အခြေခံ၍ (၂)ဆတိုးထားသည့်ကိန်းများပါဝင်သည့် list တစ်ခု ပြုလုပ်ရမည့် List comprehensions ဖြစ်သည်။

```
numbers = [1, 2, 3, 4, 5]
doubled = [ ]                          # (၂) ဖြင့်မြှောက်ထားသည့်ကိန်းများ append လုပ်ရန်
for number in numbers:
    doubled.append(2 * number)         #ကိန်းတစ်လုံးချင်းကို (၂)ဖြင့် မြှောက်သည်။
print(doubled)
```

အထက်က ကုဒ်များသည် အလုပ်ဖြစ်သော်လည်း အနည်းငယ် ရှုပ်ထွေးသည်။ အောက်တွင် ပို၍ ကျစ်လစ်သော syntax တစ်ခုဖြင့် list အသစ်တစ်ခု တည်ဆောက်ပုံ(list comprehension)ကို ပြန်လည် ရေးသားထားသည်။

```
numbers = [1, 2, 3, 4, 5]
doubled = [2 * number for number in numbers]
print(doubled)
```

```
[2, 4, 6, 8, 10]
```

Pythonic နည်း - for loop သည် list constructor syntax အတွင်းသို့ ပြောင်းသွားသည်။ (၂)ဖြင့် မြောက်ထားသည့်ကိန်းများ ထည့်ရန်(append လုပ်ရန်) empty list တည်ဆောက်ရန် မလိုအပ်တော့ပါ။ ဥပမာအားဖြင့် နှစ်ဆကိန်းများ အစား မကိန်းများသာ ပါသည့် list တစ်ခု တည်ဆောက်ပုံကို အောက်တွင် ဖော်ပြထားသည်။

```
numbers = range(10)
odds = [number for number in numbers if number % 2 != 0]
odds
```

```
[1, 3, 5, 7, 9]
```

ဤဥပမာသည် 0 မှ 4 အထိ ပါသော xs နှင့် ys list နှစ်ခုမှ ကိန်းများကို (x, y) ပုံစံဖြင့်တွဲ၍ list အသစ်တစ်ခု တည်ဆောက်ပေးရန် ဖြစ်သည်။

```
xs = range(5)
ys = range(5)
pairs = []                                  # x နှင့် y ကိန်းများ append လုပ်ရန်
for x in xs:                                # x အတွက် for loop
    for y in ys:                            # y အတွက် for loop
        pairs.append((x, y))
print(pairs)
```

```
[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 1), (1, 2), (1, 3)
, (1, 4), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (3, 0), (3, 1), (3, 2
), (3, 3), (3, 4), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
```

Pythonic code ဖြင့် ပိုတိကျ ကျစ်လစ်စွာ အောက်ပါအတိုင်း ရေးသားနိုင်သည်။ x အတွက် for loopနှင့် y အတွက် for loop တွဲရေးသည်။ Pythonic code(more concise version)

```
xs = range(5)
ys = range(5)
pairs = [(x, y) for x in xs
                for y in ys]
print(pairs)
```

```
[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 1), (1, 2), (1, 3)
, (1, 4), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (3, 0), (3, 1), (3, 2
), (3, 3), (3, 4), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
```

အောက်မှ Pythonic ဥပမာများသည် Raymond's talk မှ Transforming Code into Beautiful, Idiomatic Python ကို မှီငြမ်းထားပါသည်။

19.2 List တစ်ခုအတွင်းရှိ item များကို ထုတ်ယူခြင်း (Looping over a collection)

for loop ပတ်ရန်တွက် len() ဖြင့် list ထဲတွင် ရှိသည့် item အရေအတွက်ကို ယူသည်။

```
colors = ['red', 'green', 'blue', 'yellow']
for i in range(len(colors)):
```

```
    print (colors[i], end= ' ,')
```

```
red, green, blue, yellow ,
```

Pythonic code

```
for color in colors:
    print(color, end= ', ')
```

```
red, green, blue, yellow,
```

List တစ်ခုအတွင်းရှိ item များကို ဖော်ထုတ်ခြင်း (Looping over a collection)

19.3 Looping backwards

for loop ပတ်ရန်တွက် len() ဖြင့် list ထဲတွင် ရှိသည့် item အရေအတွက်ကို ယူသည်။

```
colors = ['red', 'green', 'blue', 'yellow']
for i in range(len(colors)-1, -1, -1):
    print (colors[i], end= ', ')
```

```
yellow, blue, green, red,
```

Pythonic code

```
for color in reversed(colors):
    print(color, end= ', ')
```

```
yellow, blue, green, red,
```

19.4 Looping over a collection and indices

List or Tuple တစ်ခုအတွင်းရှိ item များနှင့် index နံပါတ်များတွဲခြင်း

for loop ပတ်ရန်တွက် len() ဖြင့် list ထဲတွင် ရှိသည့် item အရေအတွက်ကို ယူသည်။

```
colors = ('red', 'green', 'blue', 'yellow')
for i in range(len(colors)):
    print (i, '--->', colors[i])
```

```
0 ---> red
1 ---> green
2 ---> blue
3 ---> yellow
```

Pythonic code

```
for i, color in enumerate(colors):
    print(i, '--->', color)
```

```
0 ---> red
1 ---> green
2 ---> blue
3 ---> yellow
```

enumerate() သည် ပို၍ မြန်သည့်နည်း ဖြစ်သည်။

19.5 List နှစ်ခုအား တွဲခြင်း (Looping over two collections)

for loop ပတ်ရန်တွက် len() ဖြင့် list နှစ်ခုစလုံးမှ item အရေအတွက်ကို ယူသည်။ ထိုနှစ်ခုထဲမှ အနည်းဆုံးအရေအတွက်ကို min() ဖြင့် ရွေးသည်။

```
names = ['raymond', 'rachel', 'matthew']
colors = ['red', 'green', 'blue', 'yellow']
color = ['red', 'green', 'blue', 'yellow']

n = min(len(names), len(colors))
for i in range(n):
    print(names[i], '--->', colors[i])
```

```
raymond ---> red
rachel ---> green
matthew ---> blue
```

Pythonic code

zip() သည် memory ထဲတွင် list အသစ်တစ်ခု တည်ဆောက်သည်။

```
for name, color in zip(names, colors):
    print(name, '--->', color)
```

```
raymond ---> red
rachel ---> green
matthew ---> blue
```

19.6 List အတွင်းရှိ item များကို အစဉ် အတိုင်း ရအောင်ပြန်ထုတ်ခြင်း (Looping in sorted order)

```
colors = ['red', 'green', 'blue', 'yellow']
# Forward sorted order
for color in sorted(colors):
    print(colors)
```

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

Pythonic code

Backwards sorted order

```
for color in sorted(colors, reverse=True):
    print(colors)
```

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

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

19.7 Dictionary အတွင်းရှိ key-value များကို Loop ပတ်ခြင်း(Looping over dictionary keys and values)

```
# Not very fast, has to re-hash every key and do a lookup
d = {'red': 3, 'green': 2, 'blue': 1}
print(d)
for k in d:
    print(k, '--->', d[k])
```

```
{'red': 3, 'green': 2, 'blue': 1}
red ---> 3
green ---> 2
blue ---> 1
```

Pythonic code

```
# Makes a big huge list
```

```
for k, v in d.items():
    print(k, '--->', v)
```

```
red ---> 3
green ---> 2
blue ---> 1
```

iteritems() is better as it returns an iterator. Note: in python 3 there is no iteritems() and items() behaviour is close to what iteritems() had. See documentation.

19.8 Pair များကို Dictionary တစ်ခု ဖြစ်အောင် ပြုလုပ်ခြင်း(Construct a dictionary from pairs)

```
names = ['raymond', 'rachel', 'matthew']
colors = ['red', 'green', 'blue']

d = dict(zip(names, colors))
d
```

```
{'raymond': 'red', 'rachel': 'green', 'matthew': 'blue'}
```

19.9 Dictionary အတွင်းရှိ item များကို ရေတွက်ခြင်း (Counting with dictionaries)

```
colors = ['red', 'green', 'red', 'blue', 'green', 'red']
# Simple, basic way to count. A good start for beginners.
d = {}
for color in colors:
    if color not in d:
        d[color] = 0
    d[color] += 1
```

```
print(d)
```

```
{'red': 3, 'green': 2, 'blue': 1}
```

Pythonic code

```
d = {}
for color in colors:
    d[color] = d.get(color, 0) + 1
print(d)
```

```
{'red': 3, 'green': 2, 'blue': 1}
```

19.10 Sequence များကို unpacking လုပ်ခြင်း (Unpacking sequences)

```
p = ('Raymond', 'Hettinger', 0x30, 'python@example.com')

# A common approach / habit from other languages
fname = p[0]
lname = p[1]
age = p[2]
email = p[3]
print(fname, lname, age, email )
```

```
Raymond Hettinger 48 python@example.com
```

Pythonic code

```
fname, lname, age, email = p
print(fname )
print(lname)
print(age)
print(email)
```

```
Raymond
Hettinger
48
python@example.com
```

19.11 String များကို ပေါင်းခြင်း သို့မဟုတ် ဆက်ခြင်း (Concatenating strings)

```
names = ['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', '
judith', 'charlie']
s = names[0]
for name in names[1:]:
    s += ', ' + name
print(s)
print(names)
```

```
raymond, rachel, matthew, roger, betty, melissa, judith, charlie
['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith',
 'charlie']
```

Pythonic code

```
print( ', '.join(names))
print(names)
```

```
raymond, rachel, matthew, roger, betty, melissa, judith, charlie
['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith',
 'charlie']
```

19.12 Sequences တစ်ခုကို Updat လုပ်ခြင်း(Updating sequence)

```
result = []
for i in range(10):
    s = i ** 2
    result.append(s)
print(sum(result))
```

```
285
```

Pythonic code

```
print(sum(i**2 for i in range(10))) # Pythonic code
```

```
285
```

19.13 If Statement

If statement ကို သမာရိုးကျရေးနည်းဖြင့် ရေးထားသည်။

```
condition = True
if condition:
    x = 1
else:
    x = 0
print(x)
```

```
1
```

Pythonic code

```
condition = True
x =1 if condition else 0
print(x)
```

```
1
```

- End-