Python for Competitive Programmers: Tips and Tricks

python 3.7

1 Conditions

1.1 Check Emptiness of Containers

There is a better way of checking if the list is empty without using $\boxed{len()}$ function.

```
# Typical way
if len(my_list) > 0:
   print("list is not empty")
else:
   print("list is empty"

# Pythonic way
if my_list:
   print("list is not empty")
else:
   print("list is empty")
```

This is applied to the major container data types such as sets, dictionaries as well as on strings.

1.2 Ternary Expression

If you're *if...else* code is very small, you can use Ternary expression and fit into one line.

```
# Typical way
if accuracy > 90:
   reward = "positive"
else:
   reward = "negative"

# Cleaner way
reward = "positive" if accuracy > 90 else "negative"
```

1.3 Multiple Condition

Multiple Comparison:

We can have a chain comparisons for the numeric values where we need to check multiple comparison for the same variable.

```
# Typical way
if i > 0 and i < 10:
   pass

if i > 0 and i < j and j < 10:
   pass

# Pythonic way
if 0 < i < 10:
   pass

if 0 < i < < 10:
   pass
```

Evaluate multiple conditions:

We can use the built-in functions $\boxed{all()}$ and $\boxed{any()}$ as follow

```
# Typical way
if i < 10 and j > 0 and k == 5:
   pass

if i < 10 or j > 5 or k == 4:
   pass

# Pythonic way
if all([i < 10, j > 0, k == 5]):
   pass

if any([i < 10, j > 0, k == 5]):
   pass
```

2 Lists

2.1 Negative Indexing

Python supports negative indexing. We can use -1 to refer to the last element in the sequence and -2 for the second last and so on.

```
# Negative Indexing
values = [10, 20, 30, 40, 50, 60, 70, 80]

print (values[-1]) # 80
print (values[-3]) # 60
print (values[-8]) # 10
```

2.2 Sorting

Python has built-in method for sorting a list and it performs the sorting in O(n * log(n)) Time. It modifies the original list and does not create or return new list. We can even the order (descending) using the parameter $\boxed{reverse}$

```
nums = [14, 70, 45, 20, 40, 89, 10]
nums.sort()
print(nums) # [10, 14, 20, 40, 45, 70, 89]

nums.sort(reverse=True)
print(nums) # [89, 70, 45, 40, 20, 14, 10]

chars = ['X', 'A', 'E', '1', '2']
chars.sort()
print(chars) # ['1', '2', 'A', 'E', 'X']

strings = ["py", "Py", "print", "Virtual", "venv", "pprint"]
strings.sort()
print(strings)
# ["py", "Py", "print", "Virtual", "venv", "pprint"]
```

2.3 Check if list/string contains unique elements

The following trick will help to check if the elements form a list are all unique.

```
def all_unique(s):
   if len(s) == len(set(s)):
      return True
   return False

string = "venv"
print(all_unique(string)) # False

lst = [10, 15, 20, 25, 30]
print(all_unique(lst)) # True
```

3 Misc

3.1 Generate random number

Generate random number between the range

```
import random
print(random.randint(1, 5)) # number between 1 and 5 inclusive
print(random.randint(0, 1)) # print 0 or 1 randomly
```