Bu maqolani albatta saqlab qo'ying (ayniqsa Python ni o'zingiz o'rganayotgan bo'lsangiz)

Original maqola - WY Fok

Agar zerikarli karantin hayoti tufayli siz Pythonga sho'ng'ishga qaror qilsangiz, tabriklayman. Siz eng tez rivojlanayotgan dasturlash tiliga duch keldingiz. Ishonchim komilki, Python tilining soddaligi, o'rganish qulayligi va boshqalar kabi ba'zi afzalliklarini bilasiz. Umid qilamanki, siz Python tilini yanada samarali o'rganishingiz va ushbu jarayondan bahramand bo'lishingiz mumkin va buning uchun oddiy, ammo foydali "fokuslar" va maslahatlar ro'yxatini tayyorladim.

Tushuntirish: hozirda python 3.8 dan foydalanaman. Agar siz mening misollarni ishga tushirishda biron-bir muammoga duch kelsangiz, iltimos, bu Sizning Python versiyasi bilan bog'liq emasligini tekshiring.

Qiymatlarni taqqoslash

```
>>> a = 1
>>> b = 3
>>> a == 1
True
>>> a == 2
False
>>> a == b
False
>>> a > b
False
>>> a <= b
True
>>> if a <= b :
...     print('a is less than or equal to b')
...
a is less than or equal to b
```

Siz ikki ob'ekt qiymatini solishtirishingiz mumkin. Natijada **True** yoki **False** bo'ladi. Siz **if-else** operatorida, shart sifatida to'g'ridan-to'g'ri taqqoslashni ishlatishingiz mumkin.

Shartli qaytarish operatori

Ushbu kodning o'rniga,

```
>>> def compare(a,b):
...     if a> b:
...             return a
...     else:
...             return b
...
```

siz **return** operatoridan keyin bevosita shart ishlatishingiz mumkin:

```
>>> def compare(a, b):
...     return a if a > b else b
...
```

Bir qatorda, ro'yxat/kortej orqali bir nechta o'zgaruvchining belgilanishi

Buning o'rniga,

```
>>> arr_list = [1,4,7]
>>> a = arr_list[0]
>>> b = arr_list[1]
>>> c = arr_list[2]
```

siz xuddi shu ishni bir qatorda amalga oshiringiz mumkin:

```
>>> a, b, c = arr_list
>>> a
1
>>> b
4
>>> c
7
```

Ro'yxatlarni generatsiya orqali yaratish

Quqidagi kodning o'rniga,

```
>>> arr_list = [1,4,7]
>>> result = []
>>> for i in arr_list:
...     result.append(i*2)
...
>>> result
[2, 8, 14]
```

generatsiya orqali, bunday qilsangiz bo'ladi:

```
>>> result = [x*2 for x in arr_list]
>>> result
[2, 8, 14]
```

Ikki ro'yxat elementlarni solishtirish uchun zip ishlating

Bu kodning o'rniga,

```
>>> a = [1,5,8]
>>> b = [3,4,7]
>>> result = []
>>> for i in range(len(a)):
...     result.append(a[i] if a[i] < b[i] else b[i])
...
>>> result
[1, 4, 7]
```

quyidagi kod ancha qulayroq:

```
>>> result = [min(i) for i in zip(a,b)]
>>> result
[1, 4, 7]
```

Lambda-dan ikkinchi element uchun ichki ro'yxatni tartiblash uchun foydalaning

```
>>> arr_list= [[1,4], [3,3], [5,7]]
>>> arr_list.sort(key=lambda x: x[1])
>>> arr_list
[[3, 3], [1, 4], [5, 7]]
```

filter, map

Ushbu kod o'rniga,

```
>>> arr_list = [-1, 1, 3, 5]
>>> result = []
>>> for i in arr_list:
...     if i > 0:
...             result.append(i**2)
...
>>> result
[1, 9, 25]
```

pastdagi kodni ishlatsangiz bo'ladi:

```
>>> result = list(map(lambda x: x**2, filter(lambda x: x > 0, arr_list)))
>>> result
[1, 9, 25]
```

Aytgancha, quyida undan ham oson variantini topasiz:

```
>>> result = [i**2 for i in arr_list if i > 0]
>>> result
[1, 9, 25]
```

Ro'yxatdagi barcha elementlar noyobligini tekshirish

Ro'yxatdagi takrorlanadigan elementlarni olib tashlash uchun set funksiyasidan foydalaning va keyin ro'yxatning uzunligi va natijada paydo bo'lgan to'plamning tengligini tekshiring.

```
>>> arr_list = [1,4,4,6,9]
>>> len(arr_list) == len(set(arr_list))
False
```

Satrlarni formatlash

Satrlarni formatlashning afzalliklaridan biri -— asl qiymatning to'g'riligiga ta'sir qilmasdan yaxlitlash bilan raqamli qiymatni chiqarish qobiliyatidir.

```
>>> pi = 3.14159
# До Python3.6
>>> print('The value of pi is {:.2f}'.format(pi))
The value of pi is 3.14

>>> a, b, c = 1,5,9
>>> print('a is {}; b is {}; c is {}'.format(a,b,c))
a is 1; b is 5; c is 9

# С Python3.6+
>>> print(f'The value of pi is {pi:.2f}')
The value of pi is 3.14
>>> pi
3.14159

>>> print(f'a is {a}; b is {b}; c is {c}')
a is 1; b is 5; c is 9
```

#До Python3.6 — #Python3.6 gacha.

#С Python3.6+ — #Python3.6+ dan so'ng.

Formatlash bo'yicha to'liq informatsiya — PyFormat.

enumerate

Bu kodning o'rniga,

```
>>> arr_list = [1, 5, 9]
>>> for i in range(len(arr_list)):
...     print(f'Index: {i}; Value: {arr_list[i]}')
...
Index: 0; Value: 1
Index: 1; Value: 5
Index: 2; Value: 9
```

enumerate dan foydalaning:

```
>>> for i, j in enumerate(arr_list):
...     print(f'Index: {i}; Value: {j}')
...
Index: 0; Value: 1
Index: 1; Value: 5
Index: 2; Value: 9
```

Ro'yxatning ma'lum bir qismini olish

```
>>> arr_list = [1,4,6,8,10,11]
>>> a, *b, c = arr_list
>>> a
1
>>> b
[4, 6, 8, 10]
>>> c
11
```

Kombinatsiyalar va qayta joylashtirish

itertools imkoniyatlaridan foydalaning.

```
>>> str_list = ['A', 'C', 'F']
>>> list(itertools.combinations(str_list,2))
[('A', 'C'), ('A', 'F'), ('C', 'F')]
>>> list(itertools.permutations(str_list,2))
[('A', 'C'), ('A', 'F'), ('C', 'A'), ('C', 'F'), ('F', 'A'), ('F', 'C')]
```

Bir vaqtning o'zida bir nechta o'zgaruvchini yangilash

Ushbu kod o'rniga,

```
>>> a = 5
>>> b = 8
>>> temp = a
>>> a = b
>>> b = temp + a
>>> a
8
>>> b
13
```

har ikkala o'zgaruvchini bir qatorda hisoblashingiz mumkin. Bunday holda, vaqtinchalik o'zgaruvchini yaratish shart emas:

```
>>> a = 5
>>> b = 8
>>> a,b = b, a+b
>>> a
8
>>> b
13
```

(PS: Nima o'zi bu? Fibonachchi ketma-ketligi!)

Ro'yxatdan satr olish

```
>>> str_list = ['This', 'is', 'WYFok']
>>> ' '.join(str_list)
'This is WYFok'
```

Siz ro'yxatdagi barcha **string** formatida bo'lgan elementlarni **join** funksiyasi yordamida birlashtirishingiz mumkin. Faqat elementlar orasidagi ajratgichni qo'shish kerak.

```
>>> ans_list = [3,6,9]
>>> 'The answer is '+','.join(map(str,ans_list))
'The answer is 3,6,9'
```

map funksiyasi orqali sonlarni satrlarga aylantirish mumkin va **join** funksiyasi bilan birga, siz yangi **string** formatidagi o'zgaruvchilarni olasiz.

Qo'shimcha ma'lumot:str.join, map

```
>>> for i in range(3):
...     print('Hello')
...
Hello
Hello
Hello
```

Underscore (pastki chiziq)

Odatda, agar siz kod blokini takrorlashni istasangiz, quyida ko'rsatilganidek, **for loop** ishlatiladi:

```
>>> for i in range(3):
...     print('Hello')
...
Hello
Hello
Hello
```

Tepada ko'rib turganingizdek, **for** uchun yaratilgan **i** o'zgaruvchisi ishlatimayapti. Bunday holatda, **i** o'rniga _ (pastki chiziq) dan foydalanishingiz mumkin. (_ - faqat o'zgaruvchining nomi va Pythonda e'tiborsiz o'zgaruvchi sifatida qaraladi. Bu **for** o'zgaruvchisidan aslida foydalanmasligimizni anglatadi.)

```
>>> for _ in range(3):
...     print('Hello')
...
Hello
Hello
Hello
```

Dict.keys, Dict.values, Dict.items

```
>>> teacher_subject = {'Ben':'English','Maria':'Math','Steve':'Science'}
>>> teacher_subject.keys()
dict_keys(['Ben', 'Maria', 'Steve'])
>>> teacher_subject.values()
dict_values(['English', 'Math', 'Science'])
>>> teacher_subject.items()
dict_items([('Ben', 'English'), ('Maria', 'Math'), ('Steve', 'Science')])
```

Lug'at uchun kalitlarni va qiymatlarni mos ravishda olish uchun **keys** va **values** funksiyalaridan foydalanishingiz mumkin. **Items** funksiyasi yordamida siz kalitlarni va qiymatlarni bir vaqtning o'zida olishingiz mumkin. Kalitlar va qiymatlar o'rtasida almashinish kerak bo'lsa, bu foydali ancha bo'ladi.

```
>>> subject_teacher = {y:x for x,y in teacher_subject.items()}
>>> subject_teacher
{'English': 'Ben', 'Math': 'Maria', 'Science': 'Steve'}
```

Muhim: almashtirishda ikki nusxadagi qiymatlardan ehtiyot bo'ling, aks holda siz bo'sh bo'lgan elementlarni olasiz.

(**Qo'shimcha**: zip bilan siz ikkita ro'yxatning lug'atini yaratishingiz mumkin)

```
>>> subject = ['English','Math','Scienc']
>>> teacher = ['Ben','Maria','Steve']
>>> subject_teacher = {f:v for f,v in zip(subject,teacher)}
>>> subject_teacher
{'English': 'Ben', 'Math': 'Maria', 'Scienc': 'Steve'}
```

Ikki to'plamni solishtirish

```
>>> a = {1,2,3}
>>> b = {1,2,3,4,5}

# Is a a subset of b?
>>> a<=b
True

# Is a a superset of b?
>>> a>=b
False

# Union of a and b
>>> a.union(b)
{1, 2, 3, 4, 5}

# Intersection of a and b
>>> a.intersection(b)
{1, 2, 3}

# Difference

# Return elements in a but not in b
>>> a.difference(b)
set()

# Return elements in b but not in a
>>> b.difference(a)
{4, 5}
```

Is a a subset of b? — #a b ning ichki to'plamimi?

Is a a superset of b — #b a ning ichki to'plamimi?

#Union of a and b — #a va b ning birlashmasi

#Intersection of a and b — #a va b ning kesishmasi

#Difference — # Farq

#Return elements in a but not in b — #a da bo'lgan lekin b da bo'magan elementlarni qaytarish

#Return elements in b but not in a — #b da bo'lgan lekin a da bo'magan elementlarni qaytarish

collections.Counter

Ro'yxatdagi barcha elementlarning sonini hisoblash zarur bo'lsa, bu juda qulay. Ro'yxatdagi ushbu elementlarning tegishli soni bilan, ro'yxatning barcha elementlarini aks ettirishda **Counter** sinfining ob'ekti yordam beradi.

```
>>> import collections

>>> arr_list = [1,1,1,1,2,2,2,3,3,5,5,5,7]
>>> c = collections.Counter(arr_list)
>>> c
Counter({1: 4, 2: 3, 5: 3, 3: 2, 7: 1})
>>> type(c)
<class 'collections.Counter'>
>>> c[1]
4
>>> c[6]
0

# Convert back to a dictionary
>>> dict(c)
{1: 4, 2: 3, 3: 2, 5: 3, 7: 1}
```

#Convert back to a dictionary — #Qaytib lu'gatga aylantirish

Xulosa

Bu 'fokuslar' juda oddiy bo'lsa-da, ular vaqtni tejashga va kodingizni soddalashtirishga yordam beradi. Umid qilamanki, maqola sizga oddiy Python xususiyatlaridan qanday foydalanishni tushunishga yordam berdi. Ta'lim va kodlashda omad!

Tarjima: Yerzakov Jamshid