DAY61

PYTHON.BOY

5 WEIRD PYTHON FEATURES

PYTHON

1. There is an "else" statement, but no "if"

Many programming languages have the "if-else" structure to handle conditional statements.

However, in Python, you can even use an "else" statement without an "if".

```
leaders = ["Elon", "Tim", "Warren"]

for i in leaders:
    if i == "Yang":
        print("Yang is a leader!")
        break
else:
    print("Not found Yang!")

# Not found Yang!
```

For example, the above code has no "if" statements. But the code block under the "else" statement was executed successfully!

This is the "for-else" syntax in Python.

PYTHON

It is a weird feature and we should be careful to use it in production. But the idea of it is unexpectedly simple:

The "else" block will execute when there is no break in the for loop.

To prove it, let's change the list a bit:

```
leaders = ["Yang", "Elon", "Tim", "Warren"]

for i in leaders:
    if i == "Yang":
        print("Yang is a leader!")
        break
else:
    print("Not found Yang!")

# Yang is a leader!
```

As the above program shows, the leaders list contains "Yang" at this time. So the for loop was broken and the else statement wasn't executed.

2. An immutable tuple has been changed

- As we know, tuples are immutable Python objects.
- But the following tuple can be changed:

```
tp = ([1, 2, 3], 4, 5)
tp[0].append(4)
print(tp)
# ([1, 2, 3, 4], 4, 5)
```

- This is because the mutability of nested Python objects depends on each object itself. The tp is an immutable tuple, but the first item of tp is a list which is mutable.

3. 256 is 256, but 257 is not 257

- If you compare numbers in Python, sometimes the results are surprising:

```python
tp = ([1, 2, 3], 4, 5)
tp[0].append(4)
print(tp)
# ([1, 2, 3, 4], 4, 5)
```

- This is because the mutability of nested Python objects depends on each object itself. The tp is an immutable tuple, but the first item of tp is a list which is mutable.

```python
a=256
b=256
print(a is b) # True

x=257
y=257
print(x is y) # False
```

- Under the hood, Python pre-loads all the small integers in the range of [-5, 256] to save time and memory costs. Therefore, when an integer in this range is declared, Python just references the cached integer to it and won't create any new object.
- In a word, the a and b are the same object, but the x and y are two different objects.
- We can print the id of every variable to prove this:

```
print(id(a)) # 9772544
print(id(b)) # 9772544

print(id(x)) # 139673526812528
print(id(y)) # 139673526812496
```

- This mechanism is named integer interning or integer caching.

4. String interning in Python

- Similar to the integer interning mechanism, Python also caches small-size strings to save computing resources.
- Let's see an example:

```python
a = "Yang"
b = "Yang"
print(a is b) # True

c = "Yang Zhou"
d = "Yang Zhou"
print(c is d) # False
```

- The string interning also depends on the implementation of Python.
- For this example, I use CPython and its caching algorithm is AST optimizer, which can cache up to 4096 characters, but any strings that include spaces will not be interned.

5. "+=" is faster than "="

- To concatenate strings in Python, the += and + operators can give us the same result but at different costs:

```python
import timeit

print(timeit.timeit(
  "s1 = s1 + s2 + s3",
  setup="s1 = ' ' * 100000; s2 = ' ' * 100000; s3 = ' ' * 100000",
  number=100
))
# 0.20684236199849693
print(timeit.timeit(
  "s1 += s2 + s3",
  setup="s1 = ' ' * 100000; s2 = ' ' * 100000; s3 = ' ' * 100000",
  number=100
))
# 0.0148679150006501
```

- In Python, the += operator is faster than + when concatenating more than two strings. Because the += is an in-place operation and the time for creating a new object will be saved compared to the + operation.

- This feature seems weird at the first glance but it can help us speed up our programs significantly, especially for complex string-handling scenarios.

LIKE | COMMENT | SHARE | FOLLOW

Thanks for reading :)

if you found this post useful, then giving a like and following this page would mean the world to me <3

PYTHON.BOY