انتساب متغیر

قواعد انتخاب اسم

- اسامی با عدد آغاز نمی‌شوند
- اسم‌ها نمی‌توانند شامل فاصله باشند بجای فاصله از زیرخط یاکاراکتر زیر استفاده کنید: `_`
- اسم‌ها نمی‌توانند شامل یکی از علامت‌های زیر باشند:
  `:'",<>/?|\!@#%^&*~-+`
- بنابر استاندارد (PEP8) یکی از حالت‌های زیر مناسب است:
  `lowercase` with `underscores_` or `SeparateByCapitals`
- از واژگان کلیدی و یا کلماتی که در پایتون معنا دارند استفاده نکنید، مانند:
  `list` and `str`
- avoid using the single characters `l` (lowercase letter el), `O` (uppercase letter oh) and `I` (uppercase letter eye) as they can be confused with `1` and `0`

Dynamic Typing

وقتی می‌گوییم پایتون از *dynamic typing* استفاده می‌کند، یعنی شما می‌توانید یک متغیر از یک جنس را دوباره با یک مقدار از جنس دیگری مقدار دهی کنید.

```
In [1]: my_dogs = 2

In [2]: my_dogs
Out[2]: 2

In [3]: type(my_dogs)
Out[3]: int

In [4]: my_dogs = ['Sammy', 'Frankie']

In [5]: my_dogs
Out[5]: ['Sammy', 'Frankie']

In [6]: type(my_dogs)
Out[6]: list
```

Pros and Cons of Dynamic Typing

مزایا و معایب Dynamic Typing

مزایای Dynamic Typing:

- خیلی ساده می‌شود کار کرد
- سرعت توسعه خیلی زیاد می‌شود.

معایب Dynamic Typing

- ممکن هست حواستان نباشد و اشتباه کنید!
- باید از جنس متغیرهایی که استفاده می‌کنید آگاه شوید، مثلا دستور `type()`

Assigning Variables

Variable assignment follows `name = object`, where a single equals sign `=` is an *assignment operator*

```
In [5]: a = 5

In [6]: a
Out[6]: 5
```

Here we assigned the integer object `5` to the variable name `a`.
Let's assign `a` to something else:

```
In [7]: a = 10

In [8]: a
Out[8]: 10
```

You can now use `a` in place of the number `10`:

```
In [9]: a + a
Out[9]: 20
```

Reassigning Variables

Python lets you reassign variables with a reference to the same object.

```
In [10]: a = a + 10

In [11]: a
Out[11]: 20
```

There's actually a shortcut for this. Python lets you add, subtract, multiply and divide numbers with reassignment using `+=`, `-=`, `*=`, and `/=`.

```
In [12]: a += 10

In [13]: a
Out[13]: 30

In [14]: a *= 2

In [15]: a
Out[15]: 60
```

Determining variable type with `type()`

You can check what type of object is assigned to a variable using Python's built-in `type()` function. Common data types include:

- **int** (for integer)
- **float**
- **str** (for string)
- **list**
- **tuple**
- **dict** (for dictionary)
- **set**
- **bool** (for Boolean True/False)

```
In [16]: type(a)
Out[16]: int

In [17]: a = (1,2)

In [18]: type(a)
Out[18]: tuple
```

Simple Exercise

This shows how variables make calculations more readable and easier to follow.

```
In [19]: my_income = 100
         tax_rate = 0.1
         my_taxes = my_income * tax_rate

In [20]: my_taxes
Out[20]: 10.0
```

Great! You should now understand the basics of variable assignment and reassignment in Python.
Up next, we'll learn about strings!

In [8]:
```
a = 10
b = 20
c = a
a = b
b = c
print(a, b)
```

```
20 10
```

In [9]:
```
a = 10
b = 20
a, b = b, a
print(a,b)
```

```
20 10
```