Variable Assignment

Dynamic Typing

Python uses dynamic typing, meaning you can reassign variables to different data types. This makes Python very flexible in assigning data types; it differs from other languages that are statically typed.

In [1]:

```
my_dogs = 2
```

In [2]:

```
my_dogs
```

Out[2]:

```
2
```

In [3]:

```
my_dogs = ['Bruno', 'Sheru']
```

In [4]:

```
my_dogs
```

Out[4]:

```
['Bruno', 'Sheru']
```

Pros and Cons of Dynamic Typing

Pros of Dynamic Typing

very easy to work with

faster development time

Cons of Dynamic Typing

may result in unexpected bugs!

Assigning Variables

Variable assignment follows name = object, where a single equals sign = is an assignment operato

In [5]:

```
a = 5
```

In [6]:

```
a
```

Out[6]:

```
5
```

You can now use a in place of the number 5:

In [7]:

```
a + a
```

Out[7]:

```
10
```

Reassigning Variables

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

In [8]:

```
a = a + 10
```

In [9]:

```
a
```

Out[9]:

```
15
```

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

In [10]:

```
a += 10
```

In [11]:

```
a
```

Out[11]:

```
25
```

In [12]:

```
a *= 2
```

In [13]:

```
a
```

Out[13]:

```
50
```

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 [14]:

```
type(a)
```

Out[14]:

```
int
```

In [15]:

```
a = (1,2)
```

In [16]:

```
type(a)
```

Out[16]:

```
tuple
```

In [ ]:

```

```