01-Variable Assignment

October 26, 2020

Coursework delivered by: Alison Mukoma

1 Variable Assignment

1.1 Rules for variable names

- names can not start with a number
- names can not contain spaces, use _ intead
- names can not contain any of these symbols:

  `:'",<>/?|\!@#%^&*~-+`
- it's considered best practice (PEP8) that names are lowercase with underscores
- avoid using Python built-in keywords like `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`

1.2 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*.

[21]:
```
my_dogs = 2
```

[22]:
```
my_dogs
```

[22]:
```
2
```

[23]:
```
my_dogs = ['Sammy', 'Frankie']
```

[24]:
```
my_dogs
```

[24]:
```
['Sammy', 'Frankie']
```

1.2.1 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!
- you need to be aware of `type()`

1.3 Assigning Variables

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

[25]:
```
a = 5
```

[26]:
```
a
```

[26]: 5

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

[27]:
```
a = 10
```

[28]:
```
a
```

[28]: 10

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

[29]:
```
a + a
```

[29]: 20

1.4 Reassigning Variables

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

[30]:
```
a = a + 10
```

[31]:
```
a
```

[31]: 20

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

[32]:
```
a += 10
```

[33]:
```
a
```

[33]:
```
30
```

[34]:
```
a *= 2
```

[35]:
```
a
```

[35]:
```
60
```

1.5 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)

[36]:
```
type(a)
```

[36]:
```
int
```

[37]:
```
a = (1,2)
```

[38]:
```
type(a)
```

[38]:
```
tuple
```

1.6 Simple Exercise

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

[39]:
```
my_income = 100
tax_rate = 0.1
my_taxes = my_income * tax_rate
```

[40]:
```
my_taxes
```

[40]:
```
10.0
```

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