session_one___01_variable_assignment

September 18, 2019

1 Variable Assignment

1.1 Rules for variable names

- names cannot start with a number
- underscore ' _ ' is the only special charater that can be included in a variable name
- it's considered best practice (PEP8) that names are lowercase with underscores
- avoid using Python built-in keywords like `int` 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`
- some valid variable names are: `apple`, `_apple`, `app_le`, `apple11`, `apple1apple`
- some invalid variable names are: `9apple`, `int`, `float`

1.2 Assignment Operator

Variable assignment is done using the *assignment operator*. A single equals sign `=` is the *assignment operator* in Python.

[1]:
```
# Here, number_of_apples is assigned a value of 10
number_of_apples = 2
```

Now, we can refer to `number_of_apples` anywhere down the line.

[2]:
```
# Here, cost_of_apple is assigned a value of 20
cost_of_apple = 20
```

Now, we can refer to `cost_of_apple` anywhere down the line.

[3]:
```
# Here, we refer to number_of_apples and cost_of_apples and compute the total␣
↪cost
# and store it in the variable total_cost
total_cost = number_of_apples * cost_of_apple

print(total_cost)
```

40

1.3 Determining variable type with `type()`

You can check what type of data 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)

Don't be intimidated by the words `function`, `list`, `set`, `dict`, etc. We are going to cover all that soon.

Just have a look at the examples below, you will understand what `type()` is all about.

[4]:
```
type(2)
```

[4]:
```
int
```

[5]:
```
type(3.4)
```

[5]:
```
float
```

[6]:
```
type(cost_of_apple)
```

[6]:
```
int
```

[7]:
```
cost_of_mango = 5.50
```

[8]:
```
type(cost_of_mango)
```

[8]:
```
float
```

1.4 Reassigning Variables

Python allows you to reassign variables to a reference of itself.

[9]:
```
a = 1
```

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

[11]:
```
a
```

[11]:
```
11
```

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

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

[13]:
```
a
```

[13]:
```
21
```

[14]:
```
a *= 10
```

[15]:
```
a
```

[15]:
```
210
```

[16]:
```
a -= 200
```

[17]:
```
a
```

[17]:
```
10
```

[18]:
```
a /= 10
```

[19]:
```
a
```

[19]:
```
1.0
```

1.5 Dynamic Typing

Python uses *dynamic typing*, meaning you can reassign variables to different data types. This makes Python very flexible in assigning data types.

`cost_of_mango` is float, but due to the availability of dynamic typing, we can reassign `cost_of_mango` to an integer value.

[20]:
```
type(cost_of_mango)
```

[20]:
```
float
```

[21]:
```
cost_of_mango = 20
```

[22]:
```
type(cost_of_mango)
```

[22]:
```
int
```

Though dynamic typing increases productivity, it may result in unexpected bugs!