# Lesson 4: More About Variables

## 1 Lists and arrays

In the last lesson we learned about lists and arrays, and how they behave differently. There are other aspects of lists and arrays that we need to explore. Consider this example:

```
import numpy as n
x = n.linspace(2,4,5)
print x
x[3] = 17.0
print x
```

The `linspace` command creates the array `[2.0, 2.5, 3.0, 3.5, 4.0]` and assigns it to the variable `x`. The line `x[3] = 17.0` replaces the number `3.5` with the number `17.0`. Note that `3.5` is actually the fourth element of the array. This illustrates a very important aspect of Python's behavior: *Indices for Python lists and arrays begin with* 0. That is, the first element of a list or array has index `0`, the second element of a list or array has index `1`, *etc.* Said another way, the first element of an array `x` is `x[0]`, the second element of an array `x` is `x[1]`, *etc.*

We can also use Python's indexing of lists and arrays to get a "slice" of the list or array. For example, consider the command `w = n.arange(6,15,1)`, which creates the list `[6,7,8,9,10,11,12,13,14]`. We can extract slices of `w` by writing `w[a:b]`. Then

$$\begin{aligned} \mathtt{x} = \mathtt{w}[3:6] &\quad \text{gives} \quad \mathtt{x} = [9, 10, 11] \\ \mathtt{y} = \mathtt{w}[:5] &\quad \text{gives} \quad \mathtt{y} = [6, 7, 8, 9, 10] \\ \mathtt{z} = \mathtt{w}[5:] &\quad \text{gives} \quad \mathtt{z} = [11, 12, 13, 14] \end{aligned}$$

Observe that `w[a:b]` includes the array elements with index values beginning with `a` and ending with `b - 1`. Similarly, `w[:b]` includes all of the array elements beginning with `0` and ending with `b - 1`. Finally, `w[a:]` includes all elements beginning with `a` and ending with the final element.

Keep two things in mind: (i) Python starts counting at 0, and (ii) the index after the colon is the first element that is *excluded* from the slice.

Finally, Python allows indexing from the end of the list or array, starting with $-1$ and proceeding backward. That is, the final element of the list or array has index $-1$; the second to last element has index $-2$, *etc.*

Exercise 1: Create the arrays listed in this section and verify all of the results.

## 2 Variables

### 2.1 Assignment

Let's take a closer look at how Python works with variables. When we write a line in Python like

```
x = 5
```

Python stores the integer 5 in a memory space and then uses the variable $x$ to reference the integer in that space. If we then write

```
y = 4
```

Python stores the integer 4 in another memory space and uses the variable $y$ to reference that integer. We can change the object that $y$ references by redefining $y$:

```
y = 6
```

The integer 4 no longer has a variable referencing it, and the memory space is available to be used by Python for some other object. Next, let's redefine $y$ by writing:

```
y = x
```

The integer 6 no longer has a variable referenceing it. Now Python uses both $x$ and $y$ to reference the same integer object 5. We can think of this sequence of commands as shown in Figure 1.

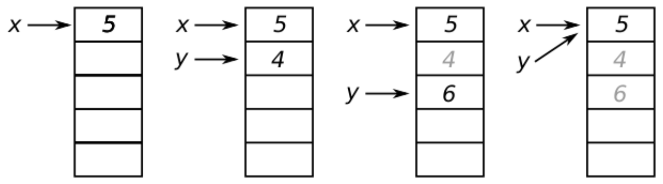


Figure 1: Example of variable assignment and reassignment.

We need to be careful about using more than one variable to reference the same object. Consider the following lines of code, which begin by assigning the array object [1,2,3,4,5] to the variable x:

```
x = [1,2,3,4,5]
y = x
print y
x[1] = 7
print y
```

Exercise 2: Write a code with the above lines. Before running the code, try to predict what the output for $y$ will be. Do you understand why the output come out this way?

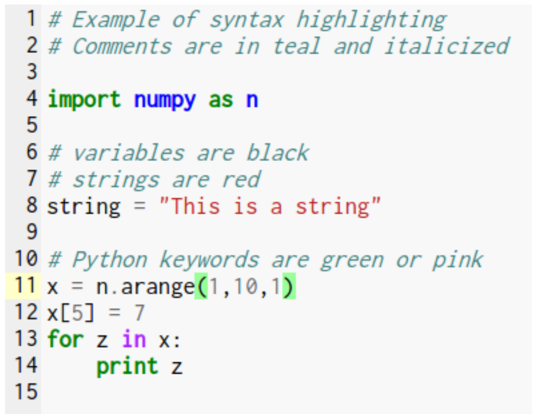
## 2.2 Naming

There are a few rules to be aware of when naming variables in Python.

1. variable names can be arbitrarily long
2. variable names can contain letters, numbers, and the underscore character (_)
3. variable names may *not* begin with a number
4. variable names are case sensitive, so time and Time are separate variables
5. variable names cannot be one of the Python keywords:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| and | del | from | as | elif | global | assert | else |
| if | break | except | import | class | exec | in | continue |
| finally | is | def | for | lambda | not | while | or |
| with | pass | yield | print | raise | return | try | |

There are other guidelines for variable naming presented in the Style Guide for Python Code known as PEP8. We don't need to worry about accidentally using one of Python's keywords as a variable name. Many text editors (Canopy included) have a feature called syntax highlighting which automatically changes the color and/or font of the text depending on the category of the text. For example, see Figure 2. With syntax highlighting, you should



```python
# Example of syntax highlighting
# Comments are in teal and italicized

import numpy as n

# variables are black
# strings are red
string = "This is a string"

# Python keywords are green or pink
x = n.arange(1,10,1)
x[5] = 7
for z in x:
    print z

```

Figure 2: Example of syntax highlighting using the Canopy text editor for Python.

know if a variable name is a keyword.

## 2.3 Types

Python has a number of variable types, including:

1. **Integer**

   The maximum integer for a 64-bit system is $2^{63} - 1$.

2. **Long**

   Integers that are $2^{63}$ or larger are automatically stored as long integers. The distinction between **integer** and **long** is transparent to the user.

3. **Float**

   This is a number containing a decimal point. Floats require more memory than integers, and are generally slower in calculations.

4. **Complex**

   Complex numbers are numbers with a real and an imaginary part. Python uses $j = \sqrt{-1}$, so it's fine to write $x = 5 - 3j$. Python will handle complex numbers correctly.

5. **String**

   A string is a sequence of characters.

6. **List**

   Indicated by square brackets: [ ]. A list is a comma separated collection of items. These items can be any type of variable, including other lists.

7. **Tuple**

   Indicated by parentheses: ( ). Tuples are like lists, except tuples are immutable—once a tuple is defined, its contents cannot be changed.

8. **Dictionary**

   Indicated by curly braces: { }. Instead of numeric indices, dictionaries use keys, which are string labels.

Exercise 3: Use python to compute $\exp(3.0j)$, $\exp(3.1j)$, *etc*. What does the answer approach as the numbers 3.0, 3.1, *etc* approach $\pi$?

Exercise 4: Use python to compute $x = 3^5$ and $y = 3^{100}$ and print the results. Use the command `print type(x), type(y)` to display the types for $x$ and $y$.

## 2.4 String formating

We've already seen how to print variables as strings using printf formating; for example:

```
print 'The position at t = {} s is y = {} m' .format(t,y)
```

In the above statement, the variable `t` is converted to a string and inserted into the first slot {}. The variable `y` is converted to a string and inserted into the second slot {}. The format can be controlled in greater detail. The statement

```
print 'The position at t = {:.3f} s is y = {:10.4f} m' .format(t,y)
```

will write `t` as a fixed point real number (a float) with three digits after the decimal point. It will write `y` as a fixed point real number with a minimum width of 10 (including the decimal point) and 4 digits after the decimal point. In addition to `f`, here are a few more common "presentation types":

```
f  fixed point real number
e  exponential notation
g  general format (f or e, depending on size of number)
d  decimal integer
```

Exercise 5: Write this code and observe the output:

```
x = 5.738108
y = 13.24913
print 'x = {} and y = {}' .format(x,y)
```

Try different formatting options within the curly brackets {}. Change `x` to an integer and test the output for different formats.

Homework: Answer these questions:

1. What are the elements of the array `arange(-2,12,2)`?
2. If `a = [19,4,-7,13,55,-12,16,8]`, what is `a[3]`? What is `a[3:]`? What is `a[:3]`?
3. What variable type is $x = 5^3$? How about $x = 5.0^3$ and $x = 5^{3.0}$?
4. If $x = 7$ and $y = 4.2$, what variable types are $x$, $y$, $x + y$ and $x * y$?
5. If $x = 3.4$ and $y = 2 + 8j$, what variable types are $x$, $y$, $x + y$ and $x * y$?
6. If $x = 10.0/3.0$, what is the output of `print "x = {:.2f}".format(x)`?
7. Write a single python code that will do the following.
   - Ask the user for their first name and last name. The code should print: `Hello, `*`name`*`.  Your initials are `*`initials`*`.`
   - Ask the user for the initial height $y_0$ and initial velocity $v_0$ of an object moving vertically in Earth's gravity. The code should compute the object's time to reach the ground. (Assume ground level is $y = 0$ with the $y$ axis pointing up, and $y_0 > 0$). The code should print: `The total time of flight for the object is t = `*`time`*` s.` Give *`time`* to three decimal places.

Your homework should include answers to each of the questions, and present the formula used for the time of flight of an object moving vertically. It should also include a copy of your python code.