Data Types and Functions

Python Module 3

Template by: Laurent Lessard

Basic Data Types

- Different Data Types
  - Built-In: Integer, Float, Boolean, Strings,...
  - User Defined: Classes

- Data Types also determine whether a certain operation makes sense for an object

```
5 + 7                   # Addition, returns 12
'Data' + 'Science'      # Concatenation, returns 'DataScience'
5 + 'Science'           # Error! Makes no sense!
'5' + 'Data'            # Concatenation, returns '5Data'
```

Integers

- Integers (`int`) are whole numbers like: 4, -23, 1782
- No limit in Python 3 on how large an integer can be (subject to the memory limits of the computer!)

```
a = -202
print(a)
```

```
-202
```

```
type(a)
```

```
int
```

Floats

- Floats (`float`) are numbers with a decimal point: `-202.25`
- Only about 16 significant digits of precision are stored.
- Use `e` for scientific notation. Ex: `1.4e24` ($= 1.4 \times 10^{24}$)

```
a = 2.1e3
print(a)
```

```
2100.0
```

```
type(a)
```

```
float
```

Booleans

- Booleans (`bool`) represent **`True`** or **`False`**.
- Widely used with logical operators like **`and`**, **`or`**, etc.

```
b1, b2 = True, False
type(b1)
```

```
bool
```

```
b1 and b2
```

```
False
```

```
x = -1
x == 1 or x > 2
```

```
False
```

Strings

- Strings (`str`) represent a sequence of characters.
- Can use either single or double quotation marks.

```
s = "Data12"
s = 'Data12'
type(s)          # gives the same answer for both!
```

```
str
```

- Length of a string can be found using the `len` function.

```
s = "Two words"
len(s)
```

```
9
```

Converting between data types

- The functions `int()`, `float()`, `bool()`, `str()` perform conversions between data types.
- The function `type()` returns the data type of an object.
- Not all conversions are possible!

Built-in Data Types Summarized

Data Type	Used to Represent	Examples
`int`	Integers: Whole numbers	`1, 44, -999, 0`
`float`	Floats: Numbers with a decimal point	`3.14159, -2.17, 0.0`
`str`	Strings: A series of characters	`"Hello World", "Data", 'This is a string'`
`bool`	Boolean: A logical (true or false) value	`True, False`

Functions

- A block of typically re-usable code to perform a task.
  - Example: A block of code that takes in your UW Campus ID and returns information like first name, last name, etc.

- Why Functions?
  - Removes redundancy in code.
  - Separates code into modules for easier readability and maintenance.

- “Should I write a function to do this?”
  - (the answer is probably yes!)

Types of functions

- Built-In
  - Part of Python packages/libraries. Examples: `print("Hello World!")`, `max(23,43,12)`, etc.
- User-Defined
  - Written by the user (you!) to achieve a specific purpose.
  - Can be created/modified as required, unlike built-in functions.

Using Built-in Functions

- Example: Printing to console - the `print` function

```
print("Hello World!")   # Name followed by arguments in brackets ( )
```

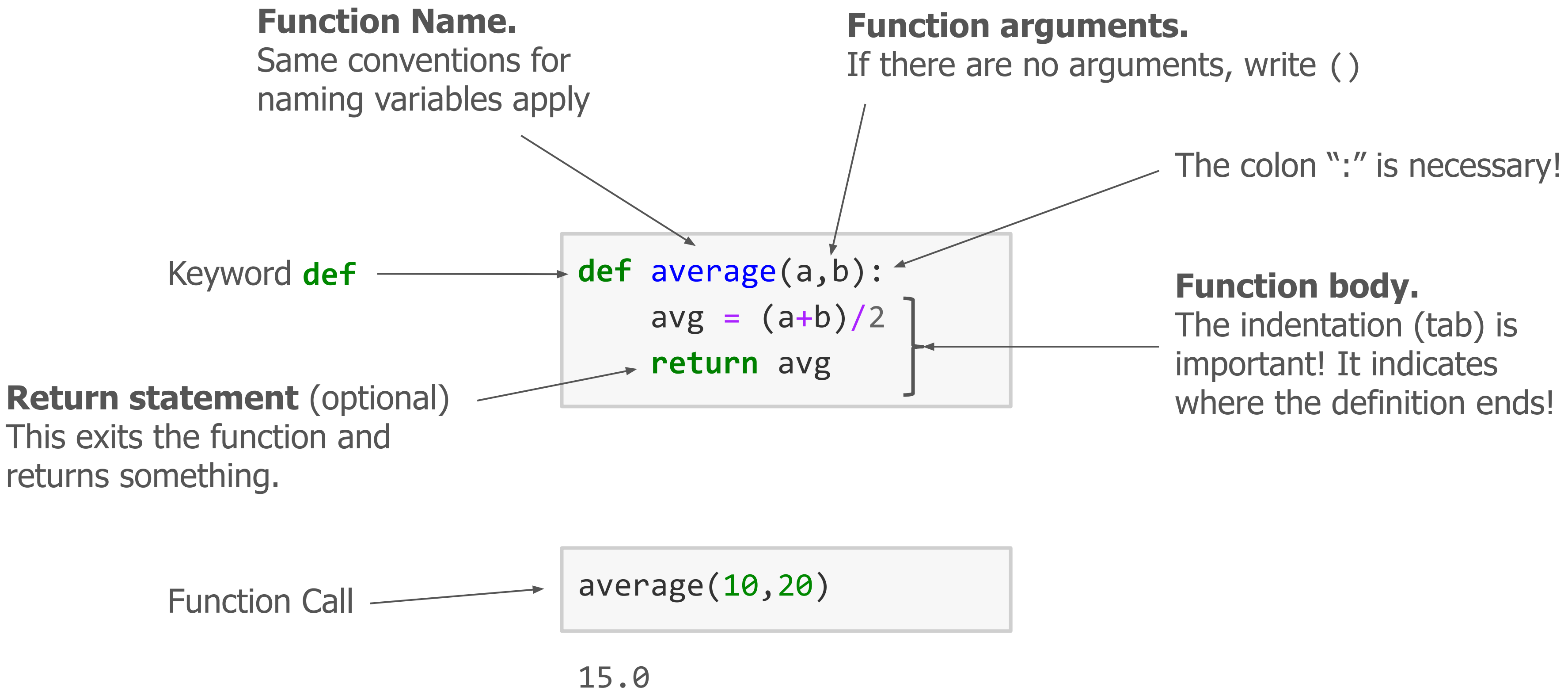
```
Hello World!
```

- Some functions are not available by default and must be imported via the appropriate package or library

```
from math import sqrt
sqrt(16)
```

```
4.0
```

Anatomy of a user-defined function



Function Name.
Same conventions for naming variables apply

Function arguments.
If there are no arguments, write ()

The colon ":" is necessary!

Keyword **def**

```
def average(a,b):
    avg = (a+b)/2
    return avg
```

Function body.
The indentation (tab) is important! It indicates where the definition ends!

Return statement (optional)
This exits the function and returns something.

Function Call

```
average(10,20)
```

```
15.0
```

Positional vs Named arguments

- Specify a default value to make an argument optional

```
def weather( day = 'Monday', forecast = 'cloudy' ):
    print('It will be', forecast, 'on', day)
```

- Arguments can be specified positionally or by name

```
weather()                                # It will be cloudy on Monday
weather('Tuesday','sunny')               # It will be sunny on Tuesday
weather('Wednesday')                     # It will be cloudy on Wednesday
weather('rainy')                         # It will be cloudy on rainy
weather(forecast='rainy')                # It will be rainy on Monday
weather(forecast='dry', day='Friday')    # It will be dry on Friday
```