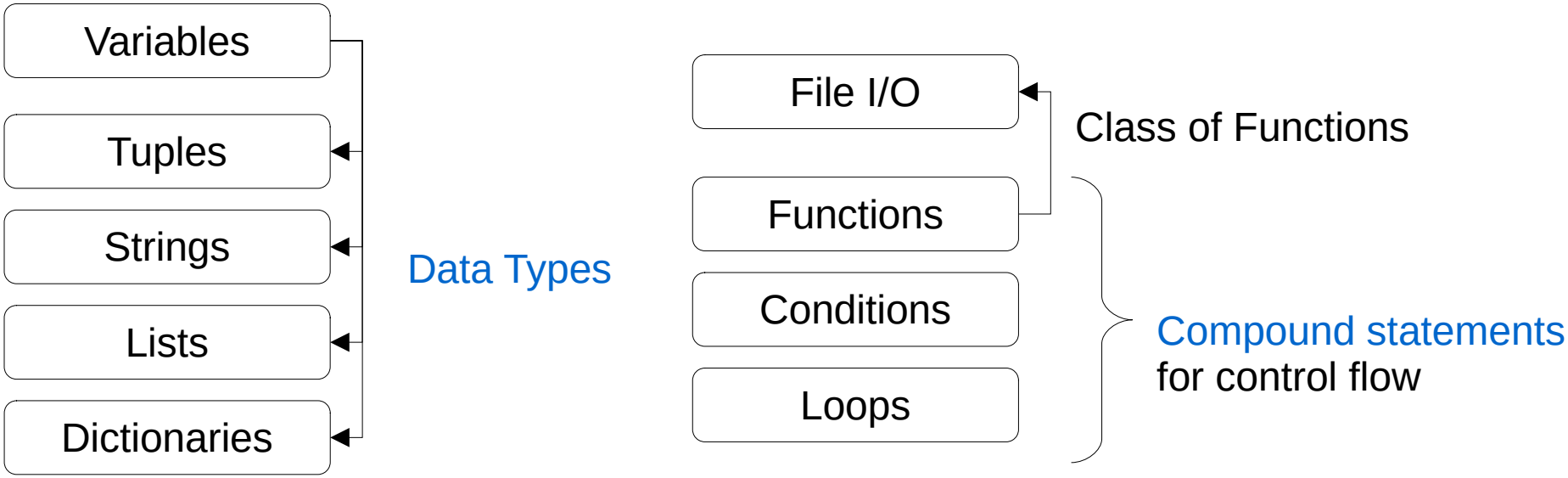
Course Topics

Variables	Tuples	Lists
Functions	Strings	Dictionaries
Conditions	Loops	File I/O

Course Topics



Variables

Objective: assign the (integer) value of “3” to a variable named “my_variable”

Variable name

Value

$$\ggg \; \mathrm{my\_variable} = 3$$

Assignment operator
(NOT EQUALS)

Note: Assignment statements are read from right to left
i.e., we are assigning the integer value “3” to the variable “my_variable”

Numeric Variables

```
>>> int_one = 1
>>> type(int_one)
<class 'int'>
```

```
>>> float_one = 1.0
>>> type(float_one)
<class 'float'>
```

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division

Operator	Operation
%	Modulo (remainder)
**	Exponent
//	Floor divison

Revision:

What is the data type for each operation, e.g. `type(1.0 + 2)`

What is the order of operations, e.g. `print(1.0 + 2 * 2)`

String Variables

```
>>> str_one = "1.0"
>>> type(str_one)
<class 'str'>
```

Iterating over Strings

```
>>> for ch in str_one:
>>>     print(ch)
1
.
0
```

String Slicing

```
>>> print(str_one[:-1])
1.
```

String Concatenation

```
>>> print("1" + ".")
1.
```

Tuples

Immutable, ordered sequence of variables

```
>>> tup_of_ones = (1, 1.0, 'one', '1.0')
>>> type(tup_of_ones)
<class 'tuple'>
```

Zero-Indexed

```
>>> tup_of_ones[0]
1
```

Tuple Slicing

```
>>> print(tup_of_ones)[:-1]
(1, 1.0, 'one')
```

Lists

Mutable, ordered sequence of variables

```
>>> list_of_ones = [1, 1.0, 'one', '1.0']
>>> type(list_of_ones)
<class 'list'>
```

Zero-Indexed

```
>>> list_of_ones[0]
1
```

List Slicing

```
>>> print(list_of_ones)[:-1]
[1, 1.0, 'one']
```

Mutable

```
>>> list_of_ones[0] = 2
>>> print(list_of_ones[0:2])
[2, 1.0]
```

Dictionary

Pairs of <u>unique</u> keys and their values

```
>>> dict_of_ones = {"int_one": 1
                    "float_one": 1.0,
                    "str_one": "1.0"}
```

Look-up

```
>>> print(dict_of_ones["str_one"])
"1.0"
```

Mutable

```
>>> dict_of_ones[2] = "two"
>>> print(dict_of_ones[2])
"two"
```

Revision:

How do you get all the key-value pairs from a dictionary?
How do you check if a key is in a dictionary?

Functions

Function name Argument(s)

class **type**(*object*)
class **type**(*name*, *bases*, *dict*, ***kwds*)

Signatures

With one argument, return the type of an *object*. The return value is a type object and generally the same object as returned by `object.__class__`.

- Same function name can have multiple “signatures”
  - Also known as “Overloading”
  - Changes behaviour depending on argument(s) provided

Control Flow

```python
print('first')
print('second')
print('third')
```

```
first
second
third
```

Commands in .py files are executed in order (i.e., from top to bottom)

How to avoid repeating commands?

For Loop

```python
elements = ['first', 'second', 'third']
for i in elements:
    print(i)
```

```
first
second
third
```

- Keyword: “for”
- Implicit assignment of element to `i`

What if we want to count from 1 to 100?

While Loop

```
1   i = 0
2   while (i < 100):
3       i = i + 1
4       print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

- Keyword: “while”
- Repeats indented code block (lines 3-4) until condition (i < 100) is not true

If Else

```python
i = 0
while (i < 100):
    if i == 0:
        print('is zero!')
    elif (i % 2) == 0.0:
        print('is even!')
    else:
        print('is odd!')
    i = i + 1
```

```
is zero!
is odd!
is even!
is odd!
is even!
is odd!
is even!
```

- Keyword: “if”, “elif”, “else”
- Executes code block if condition is met
- “else”: all other cases

Note: % is the <u>modulo</u> operator
Returns the remainder of a division operation

File I/O

Writing a file

```
>>> with open('file.txt', 'w') as file_object:
>>>     file_object.write('first line' + '\n')
>>>     file_object.write('second line')
```

Reading from a file

```
>>> with open('file.txt', 'r') as file_object:
>>>     for line in file_object:
>>>         line = line.rstrip('\n')
>>>         columns = line.split(',')
```