ACIT 2515

Python refresher

Numeric types: `int`, `float`, `complex`

Strings: `str`

Booleans: `bool`

Collections

- "Contain" multiple elements
- Elements can be of different types
- `list`, `tuple`, `set`
- `dict`

Properties of standard collections

Collection	Ordered (sortable)	Comments
`list`	Yes	Mutable.
`tuple`	Yes	**Not mutable**
`set`	No	Mutable. Elements are "unique".
`dict`	No	Key-value pairs (similar to hashmaps or arrays in other languages)

Lists and tuples

```
# Lists
my_list = list(1, 2, 3)
my_list.append("something")   # Add an element at the end of the list
my_list[0]                    # Access an element by its position in the list (starts at 0)
my_list[0] = 0                # Change values
another_list = [4, 5, 6]
my_list + another_list        # Concatenate lists with +

# Tuples
my_tuple = (1, 2, 3)
another_tuple = tuple("hello", "world")
my_tuple[1]                   # like lists
my_tuple[0] = 0               # ERROR! Tuples are immutable
```

Sets

```
my_set = set(1, 1, 1, 2, 3)   # Elements are unique
another_set = {"hello", "hello"}
s[0]                          # ERROR! Sets are not ordered or indexed
```

Dictionaries: extremely useful!

```
my_dict = dict(((1, "hello"), (2, "world")))
# {1: 'hello', 2: 'world'}
another_dict = {1: "goodbye", 2: "sunshine"}

my_dict[1]                    # 'hello'
my_dict["test"] = "something" # Add values to the dict
# {1: 'hello', 2: 'world', 'test': 'something'}
my_dict.keys()                # List of keys
my_dict.values()              # List of values
my_dict.items()               # List of key/value pairs
```

Strings

```
my_string = "hello world"
my_string[0]                    # 'h'
my_string.upper()               # 'HELLO WORLD'
my_string.split(' ')            # ['hello', 'world']
' '.join(['hello', 'world'])    # 'hello world'
'HELLO WORLD'.lower()           # 'hello world'
```

Conversion between types

```
round_number = 20
float_number = 42.4
string_number = "100"

int(float_number)              # 42
int(string_number)             # 100 (not '100' !)
float(round_number)            # 20.0
```

Functions

```
def register_student(student_id, name, international=False, scholarship=False):
    # Do something with the parameters
    return True
```

- Functions have parameters - and receive arguments.
- They return a value (of any type).

`register_student` takes four parameters (or arguments):

- `student_id`, and `name` must be provided in this order. They are *positional* arguments.
- `international` and `scholarship` are *keyword* arguments. They do not have to be provided, because they have a *default value*.

Keyword arguments can be provided in any order by using their names:

```
register_student("A01209697", "Tim", scholarship=True, international=True)
# This is a valid call, even if the keyword arguments are in a different order!
```

Return values

- All functions return "something" using the `return` statement.
- If the `return` statement is omitted, the function will `return None` (be careful!)
- A function returns ONE value - but the value can be a collection (list, set, tuple, dictionary, etc)