

COMP1531

2.3 - Objects in python

Python with Rob



What are objects?

Objects

- Technically, a fairly simple idea
- Conceptually, a rich area of software design with complicated outcomes
- There's a whole course on Object-Oriented Design and Programming, so we'll only focus on basic stuff here

Objects in python

- Contain *attributes* and *methods*
- Attributes are values inside objects
- Methods are functions inside objects
- Methods can read or modify attributes of the object

A simple example

```
1 from datetime import date
2
3 today = date(2019, 9, 26)
4
5 # 'date' is its own type
6 print(type(today))
7
8 # Attributes of 'today'
9 print(today.year)
10 print(today.month)
11 print(today.day)
12
13 # Methods of 'today'
14 print(today.weekday())
15 print(today.ctime())
```

Everything* is an object

- Almost all values in python are objects
- For example:
 - lists have an append() method

```
1 animals = ["dog", "cat", "chicken"]  
2 animals.append("sheep") # Modifies the list 'animals'
```

- strings have a capitalize() method

```
1 greeting = "hi there!"  
2 print(greeting.capitalize()) # Returns a new string
```

Creating objects

- *Classes* are blueprints for objects

```
1 class Student:
2     def __init__(self, zid, name):
3         self.zid = zid
4         self.name = name
5         self.year = 1
6
7     def advance_year(self):
8         self.year += 1
9
10    def email_address(self):
11        return self.zid + "@unsw.edu.au"
12
13 rob = Student("z3254687", "Robert Leonard Clifton-Everest")
14 hayden = Student("z3418003", "Hayden Smith")
```


Details

- Methods can be *invoked* in different ways
 - `rob.advance_year()`
 - `Student.advance_year(rob)`
- The 'self' argument is implicitly assigned the object on which the method is being invoked
- The '`__init__()`' method is implicitly called when the class is *constructed*