4 Objects

Key terms: expression `print bool int float str`

Reading: Severance 2

Exercise: Write a program that indicates, true or false, whether the current year is an even number.

Most programs handle data, organized as objects with specific names. Every object represents one or more pieces of some kind of data. For example, an object called `math.pi` represents a single floating-point figure of about 3.141592653589793.

It's important to choose good names for objects. Best practice is to use names which describe both the measure and the units used; this can be surprisingly tricky. Object names can't include spaces, so often they are lowercase_with_underscores or in CamelCase. Avoid repeating class names like `list` and `str`, or module names like `sys` and `math`, because doing so clobbers the existing references to those resources. Keywords such as `return` and `not` are also reserved and unavailable for use in naming.

Python's built-in object types, or classes, are similar to those of other languages. `str`s (strings) contain text in any language, encoded in the variable width scheme called UTF-8. For convenience, literal strings can be delimited by single quotes, `'`, double quotes, `"`, or triple quotes, `'''` or `"""`.

```
# Determine the percentage of U.S. counties that are in California:

counties_state = 58
counties_country = 3144
county_share = counties_state / counties_country
county_share_percent = 100 * county_share

print("This percentage of U.S. counties are in California:", county_share_percent)
```

`bool`s (Booleans) each have either the value `True` or `False`; `int`s (integers) are of unbounded size; these objects grow as large as needed on demand, and cannot be overflowed; `float`s (IEEE 754 floating point numbers) have huge flexibility and variable precision.[10] For convenience working with large numbers, underscores can serve as thousands separators: `10_500_000`.

All these types of objects are immutable objects, so they are not "variables." Numbers support the combined assignment operators `+=`, `*=`, etc.; the famous `++` was pointedly left out because `+=1` is more semantically consistent. These types' immutability means that if we "change" their value, we are really throwing away the old object and then using the same name for a new one.

Literal objects like `4` or `''hello''` are anonymous, or nameless. Named objects (or instances) can be created at any time by assigning to a new name: `z_meters = 4`. They can also be created on the fly by calling their class name: `int(4)`. This construct does not "cast" or "typecast" but simply creates new anonymous objects as needed.

With more than one object at hand, we might seek to combine them in a compound expression with its own value. For example, we can add and subtract numbers (`cash_balance-expenses`) or compare them (`hours_out!=hours_back`, `speed_mph>35`). These expressions call functions behind the scenes that return some desired value. Comparison operators raise `TypeError` when they can't handle the types passed (inquire whether `4>'4'`). Nonetheless, strings can be multiplied by integers: `5*'z'` is exactly what you'd think it is.

Next, we will address mathematical resources available to your programs.

[10] `https://docs.python.org/3.6/tutorial/floatingpoint.html`