Values, types, and variables

Values

A value is the fundamental thing that a program manipulates or uses to perform operations. A value is data.

Here is a string value

```
"Hello, world!"
```

Here are some numbers

```
42
```

```
12.34
```

Here the only two Boolean values

```
True
```

```
False
```

Values have types...

Types

In Python there are several fundamental data types:

- `bool`: with values `True` and `False`
- `str`: for strings like `"Hello world"`
- `int`: for integers like `1`, `42`, and `-5`
- `float`: for floating point numbers like `96.8`

Python has a `type()` function to determine the type of a value.

```
print(type(55))
print(type(False))
print(type("Hi there."))
print(type(-101.4))
```

In many programming languages the terms `class` and `type` mean the same thing or are at least closely related.

Variables

- One of the most basic and powerful concepts in programming is that of a variable
- A variable associates a name to a value:

```
message = "Hello, world!"
n = 42
e = 2.71
# note we can print variables:
print(n) # yields 42
# note: everything after pound sign is a comment
```

- It is almost always preferred to use variables over values:
- Easier to update code
- Easier to understand code (useful naming)

- For example, What does the following code do:

```
print(4.2 * 3.5)
```

- If the values are assigned to variables with meaningful names, we might have something like:

```
length = 4.2
height = 3.5
area = length * height
print(area)
```

Now a person reading the code has a good idea of what the values represent and what the output of the code means.

Variable naming

- The name associated with a variable is referred to as an *identifier*
- Variables names must start with a letter or an underscore, such as
  - `_underscore`
  - `underscore_`
- The remainder of your variable name may consist of letters, numbers and underscores
  - `password1`
  - `n00b`
  - `un_der_scores`
- Names are case sensitive
  - `case_sensitive`, `CASE_SENSITIVE`, and `Case_Sensitive` are each a different variable.

Variable naming style

- too short: `a`, `b`, `c`
- too long: `number_of_particles_in_target_region`
- better: `num_target_particles`
- CamelCase: `numTargetParticles`

This is quite important for code readability. People think about this a lot. See: https://www.python.org/dev/peps/pep-0008/#naming-conventions

Important: don't override built-in names

```
print(abs(-7))
abs = 'must do sit-ups'
print(abs(-4))
```

Exercise: mortgage calculator

Let's write a simple mortgage calculator to compute the monthly payment for a mortgage.

Here are the parameters:

- L: initial principal or value of the loan
- r: yearly interest rate
- $c = r/12$: monthly interest rate
- n: term of the mortgage in terms of number of months (30 years = 360 months)

The formula to compute the fixed monthly payment is:

$$P = L\frac{c(1+c)^n}{(1+c)^n - 1}$$

Let's code this in Python:

```
# mortgage amount in dollars
L = 100000
# yearly interest rate of 4%
r = .04
c = r / 12
# duration of mortgage in months
n = 360
P = L * (c*(1+c)**n) / ((1+c)**n-1)
print("P = " + str(P))
```

Note that in Python the notation `x**y` is used to compute "`x` raised to the power `y`".

Exercise for you: reverse mortgage calculator

Now, write python code to compute the value of the loan given the monthly payment. The formula is:

$$L = P\frac{(1+c)^n - 1}{c(1+c)^n}$$