Chapter 1 outline:

- Introduction, sets and elements (last week Wednesday)
- Python expressions (last week Friday)
- Python functions; denoting sets (**Today**)
- Set operations; visual verification of set propositions (Friday)
- Cardinality, Cartesian products, powersets (next week Monday)

Today:

- Review/finishing Friday's material
  - The Jupyter notebook environment
  - Expressions and types
  - Variables and functions
- Review of sets
- Set-builder notation
- Sets in Python
- Functions on sets

Which of the following is **not** true about sets?

- A set can contain the same element more than once. ✓
- A set is unordered.
- A set can contain elements other than numbers.
- A set can be empty.

- An **expression** is a programming construct that evaluates to a value.
- A **literal** is the simplest expression that evaluates to a specific value.
- A **type** is a set of values associated because of how they are stored in computer memory and what operations are available for them.
- A **subexpression** is an expression that is part of a larger expression.
- An **operator** is a symbol that can be applied to one or more expressions to make a larger expression.

Type	**Kind of information**
`int`	whole numbers and their opposites
`float`	real numbers
`str`	text
`bool`	true or false
`type`	types

Which of the following is **not** a `str` operator?

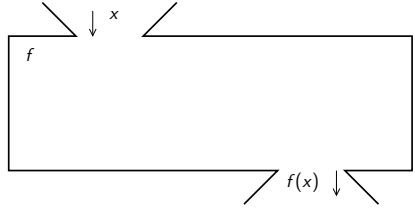
- `//` ✓
- `+`
- `*`
- `in`

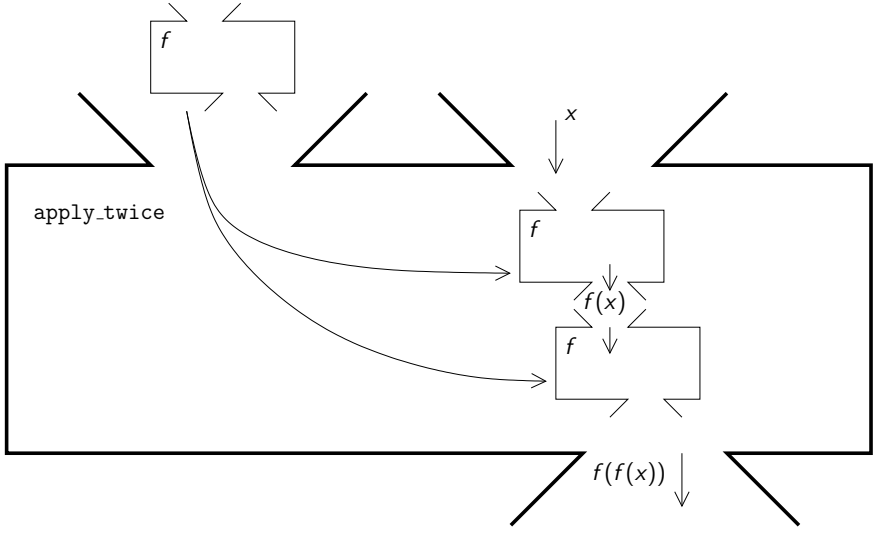
Which of the following is **not** true about types?

- *Literal* is a kind of type. ✓
- `type` is itself a type.
- Types are themselves values.
- In some cases, you can convert a value from one type to another.

A function is...

- a parameterized expression.
- a named piece of code that can be invoked many times in different contexts.
- an extension to the programming language.
- an abstract machine.
- a *value*.




f
x
apply_twice
f
f(x)
f
f(f(x))

- A **variable** is a symbol that can stand in place of a value.
- An **identifier** is a programmer-defined name
- A **keyword** is a symbol that could be an identifier except that it has a predefined meaning in the programming language.
- A **function** is a named peice of code that can be invoked from different contexts.
- A **formal parameter** is a variable used to stand for the input to a function.
- An **actual parameter** is a value passed as input to a function.
- An **application** is an expression that induces the interpeter to evaluate the body of a function, with the provided actual parameters bound to the function's formal parameters.

What is the right way to assign a value to a variable in Python?

- `x = 5` ✓
- `int x = 5;`
- `def x : 5`
- `let x == 5`

Which of the following is **not** a type of value that can be passed to a function?

1. keyword ✓
2. function
3. set
4. str

Without using the Python interpreter or a Jupyter notebook, predict the result of the Python expression

```
{ x + 1 for x in range(10,15) if x % 2 == 0}
```

1. `{11, 13, 15}` ✓
2. (Other options omitted...)

For next time:

Pg 23: Exercises 1.3.(1, 3, 6, 7)
Pg 30: Exercises 1.4.(2, 3, 5, 7)

Note that Exercises 1.3.(3,6,7) and 1.4.(3,5,7) are programming problems to do in and turn in as a Jupyter notebook on Canvas; Exercises 1.3.1 and 1.4.2 should be done and turned in on paper.

Read Sections 1.(5 & 6)

Take quiz