INFO5002: Intro to Python for Info Sys

Week 2

Slides created by: Zachary Doucet

Northeastern University

Week 2

I. Recap

II. Operators

III. Functions

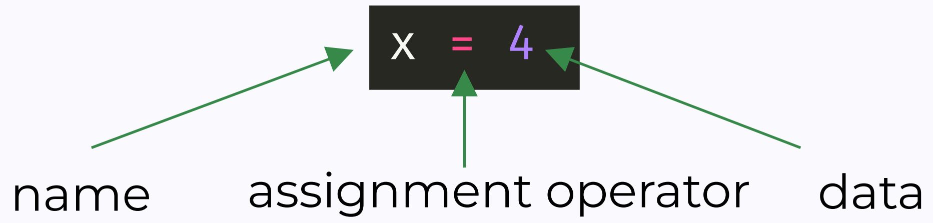
IV. Conditionals

V. Recursion

Recap

Variables

- Variables act as labels that **reference** a saved data.



```
x = 4
```

- 4 basic data types of: integers, floats, booleans, and strings.

Can define an integer differently

- Base 10 `170`
- Binary or base 2 `0b10101010`
- Octet or base 8 `0o252`
- Hexademical of base 16 `0xAA`

Operators

[1], PCC 26-27

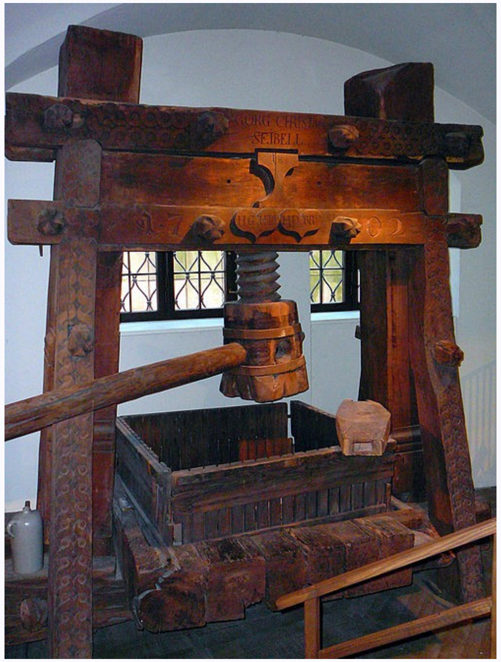
Operators as action

- A process that performs an operation is an operator.

$$OP = \{o \mid o : X \to Y\}$$

- Null operators:

$$\emptyset = \{o \in OP \mid o : X \to X\}$$


Source: Wikimedia

Arithmetic Operators

```
x = 4

# Addition

x = x + 1

# Subtraction
x = x - 2
```

```
# Multiplication

x = x * 4

# Division
x = x / 6
# Modulo
x = x % 2
```

What is x after each operation?

Arithmetic Operators (Continued)

```
x = 3

# Exponential

x = x ** 3

# Floor division (int div)
x = x // 10
```

```
# Negation

x = -x
```

What is x after each operation?

Bitwise Operators

```
x = 0b0101
y = 0b1001

# And

z = x & y

# Or
z = x | y
```

```
# Exclusive Or

z = x ^ y

# Inversion
z = ~x
```

[2]

```
# Left and right shift
z = x << 2
z = y >> 3
```

What is z after each operation?

Comparison Operators

```python
x = 10
y = 12

# Equal
z = x == y


# Difference
z = x != y
```

```python
# Greater than

z = x > y

# Less than
z = x < y

# Ordering and equal
z = x >= y
z = x <= y
```

What is z after each operation?

Logical Operators

```
x = 10
y = 12
z = 10

# And
a = x == y and x == z


# Or
a = x == y or x == z
```

```
# Not
a = not x == y and x == z
```

What is a after each operation?

Don’t forget operator precedence!

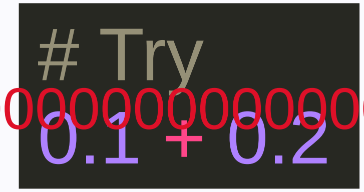
- The general rules of operator precedence from math applies to python. Thus, use parentheses to be **explicit**.

```
1 + 6 / 2  !=  (1+6) / 2
```

- Can be a common source of bugs!

Don’t forget that floats are representational!

- Performing operations on floats may not yield the expected output.



```
# Try
0.1 + 0.2
```

0.30000000000000004

- Can be a common source of bugs!

Operator shorthand

Most operators support a shorthand for operations performed on the assigned variable.

```
x = x + 1

x = x - 1

x = x * 2

x = x & 0b1
```

Can be $\xrightarrow{\hspace{2em}}$ turned

```
x += 1

x -= 1

x *= 2

x &= 0b1
```

String Operators

```
x = "Be yourself"
y = "everyone else is taken"


# Concatenation
z = x + "; " + y

# Contains
z = "else" in z
```

```
# Repetition
z = (x + ", ") * 2
```

What is z after each operation?

Let’s practice

I. Let *x* be the addition of 2 and 5 together.

II. Let *y* be 4 multiplied by 2 to the power of 3.

III. Let *z* be taken as the modulo of 1 added by 5 and 7 subtracted from 3.

IV. Let *bit* be the bitwise AND of 0b1010101010 with the bitwise inversion of 0b0101010101.

V. Let *string* be the string of “hello world” repeated 6 times while writing “hello world” only once in its instantiation.

And some more

I. Let *a* be if the integer 4 is equal to the string 4.

II. Let *b* be if 3 is equal to 3.0.

III. Let *c* be if 2 to the power of 10 is less than 10 to the power of 3.

IV. Let d be if “y i” is in the string “today is friday”.

V. Let e be if 5 * 3 is not greater than 2 subtracted from 12.

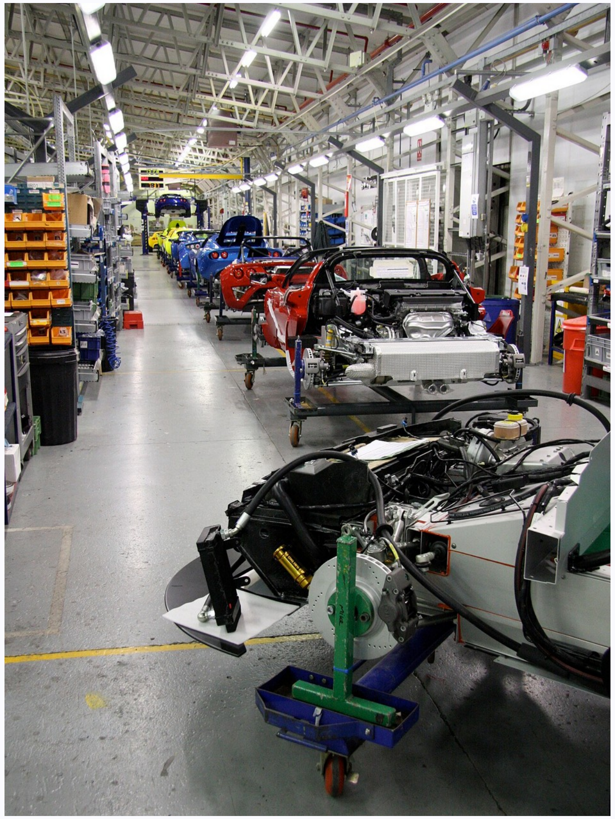
Functions

PCC 129-155

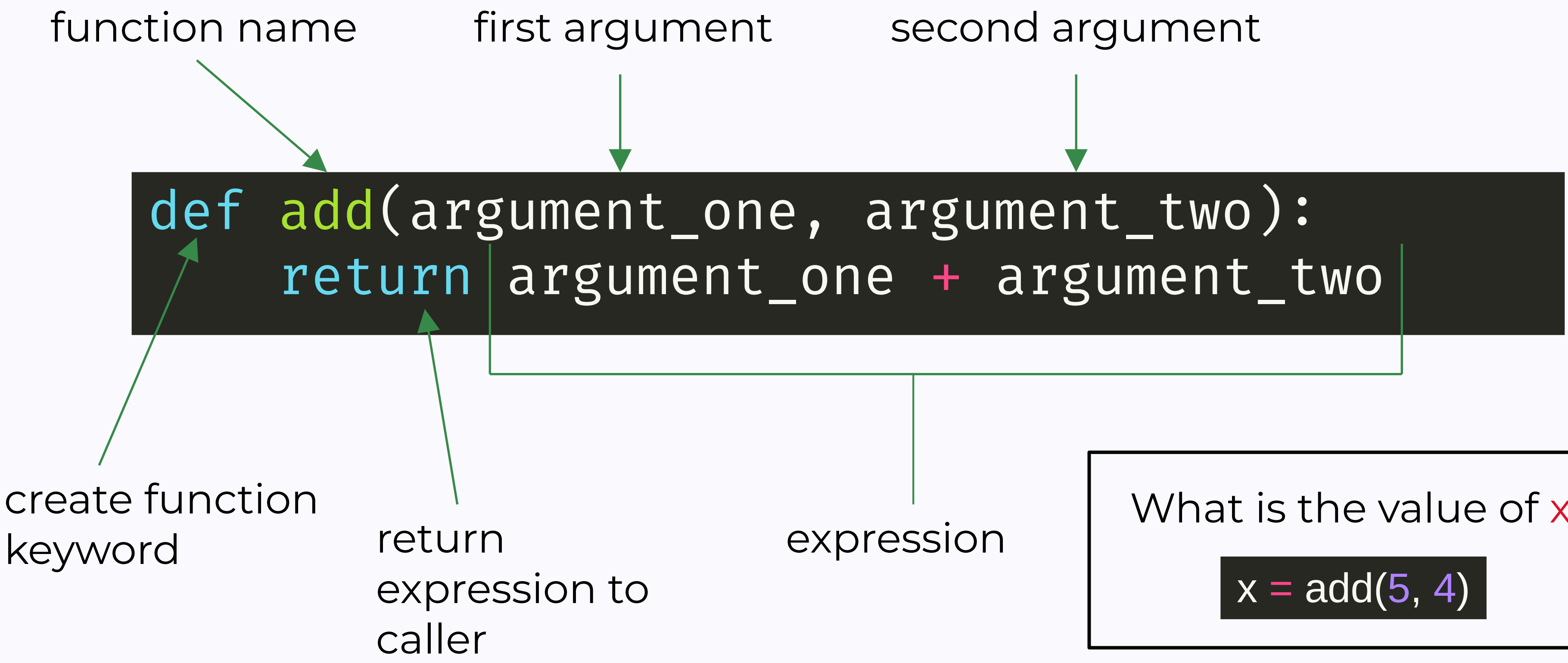
Functions as factories

Source: Wikimedia

A way to group operators together that can be executed on **different data**.


Source: Wikimedia

Creating Functions



```
def add(argument_one, argument_two):
    return argument_one + argument_two
```

What is the value of x?

```
x = add(5, 4)
```

Let’s practice

- Create the following functions:

  I. get_greeting which returns “welcome to my store”.

  II. print_greeting which prints “welcome to my store”.

  III. sub that takes two numbers and returns the subtraction of the second from the first.

  IV. multiply_all that takes five numbers and returns the multiple of them all.

Best Practices

Make code readable

- Keep each line short in length. Never more than 100 chars.
- Group similar lines together and leave a line break between logically different lines.
- Use comments to help explain confusing code.

```
# This is a comment
x = 2
```

```
"""This is a multi-line comment
    that I can run as long
    as I want """
x = 1
```

Make code readable (Continued)

- Use descriptive names for variables and functions and not embeddings, mapping, or encodings.

Single Responsibility Principle

- One thing should do one thing; and do that thing very well.

Tries to be
- Fridge
- Media Player
- Entertainment System
- Calendar

- Each function should be responsible of a single logical idea.
- Break big functions into smaller reusable functions.

Citations

[1] https://docs.python.org/3.13/library/operator.html

[2] https://en.wikipedia.org/wiki/Two's_complement