INFO5002: Intro to Python for Info Sys

Week 3

Slides created by: Zachary Doucet

LVX VERITAS VIRTVS

Northeastern University

Week 3

I. Functions+

II. Recursion

III. Conditionals

IV. Loops

V. Advanced Data Types

VI. Popular Functions

Recap

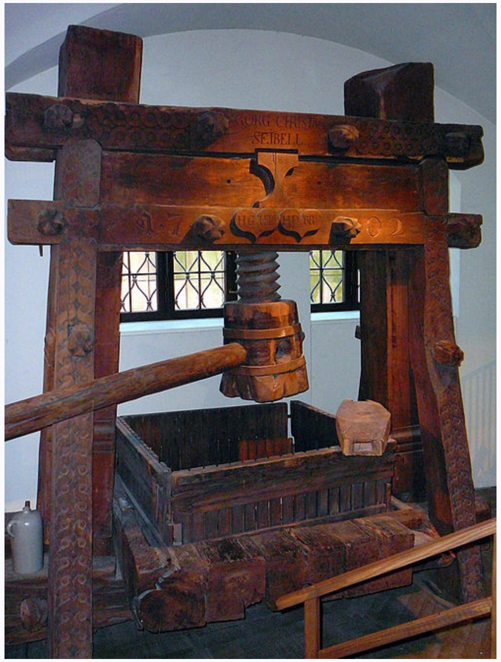
Operators as action

- A process that performs an operation is an operator.

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

- Null operators:

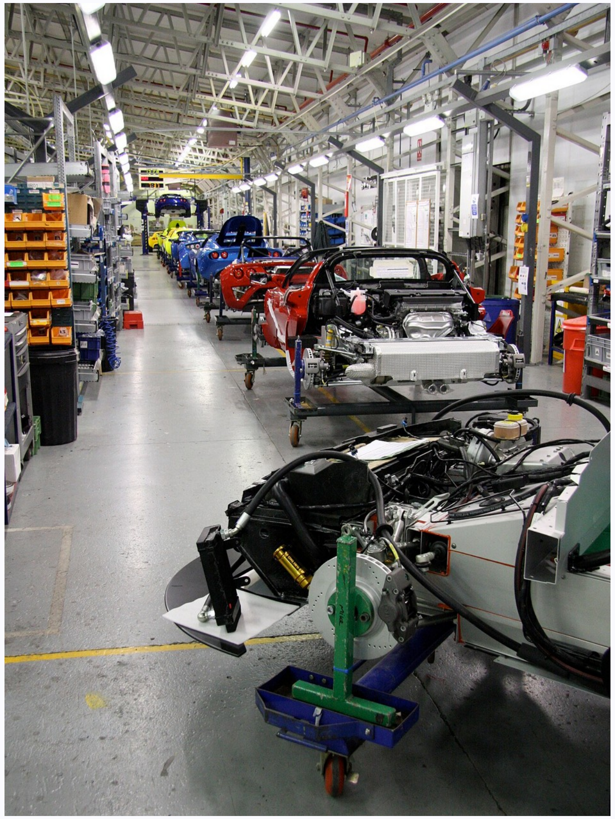
$$\emptyset = \{o \in OP \mid o : X \rightarrow X\}$$


Source: Wikimedia

Functions as factories

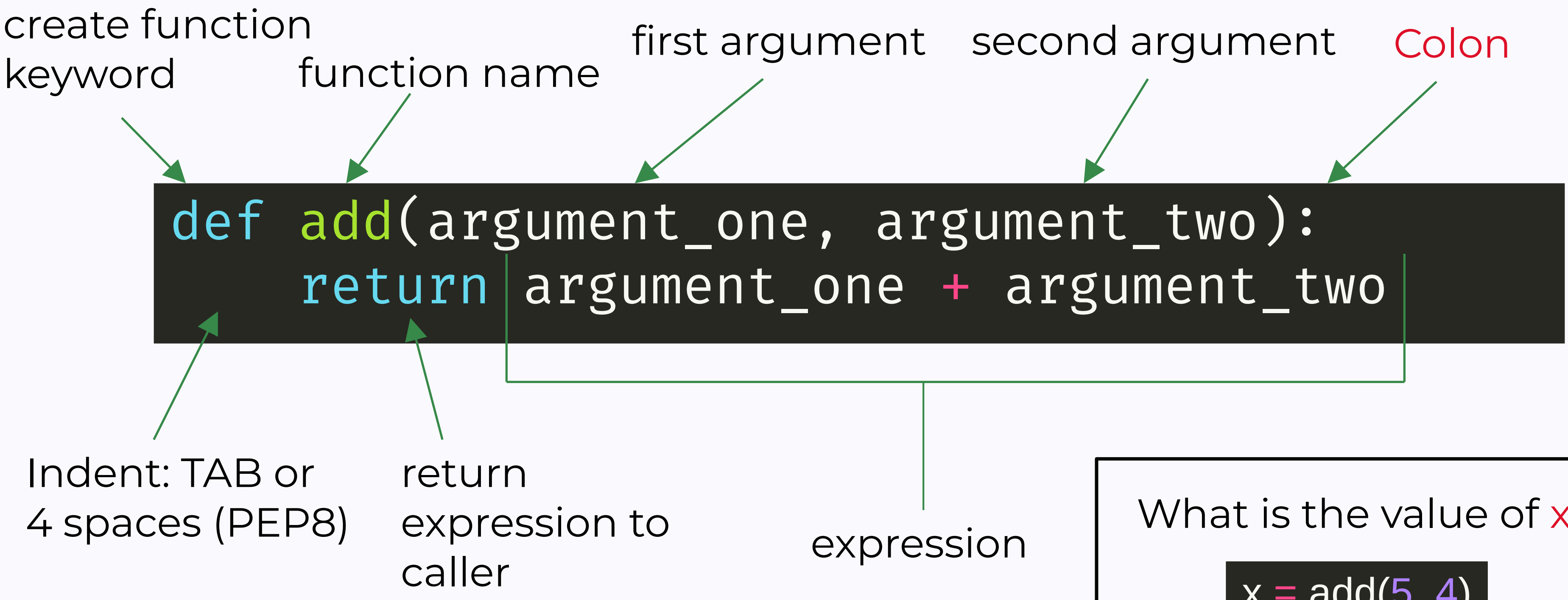
Source: Wikimedia

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



Source: Wikimedia

Creating Functions



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

What is the value of x?

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

As a recap, let's write functions.

- Create the following functions:
  1. can_vote takes in a person's age and return if they are eligible to vote in Canada (18).
  2. calculate_interest that takes in last month's balance (arg1) and returns the amount of interest based on the interest (assume already decimal) (arg2).
  3. hypotenuse that takes two lengths of a right triangle and returns the length of the hypotenuse.

Functions+

PCC 133-135

Additional functional features

- You can define a function's argument(s) as optional by providing a default value.

```python
def increment(input, by=1):
    return input + by
```

WARNING: When using default values make sure that **all non default values appear before** in function's signature.

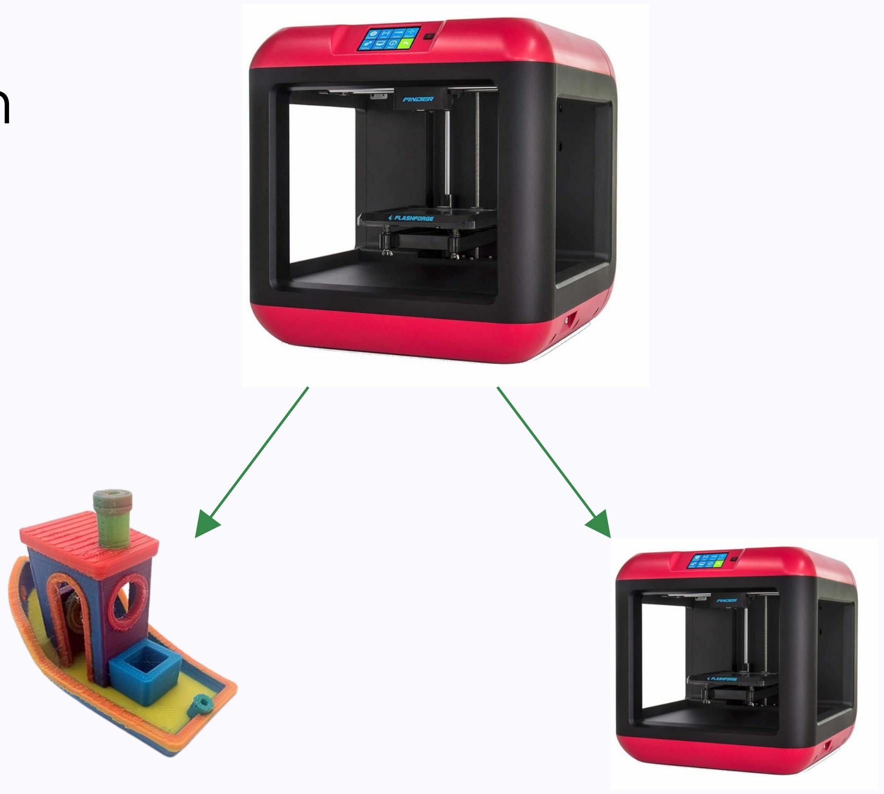
- If we want to not rely on argument positions when calling a function we can use keyword arguments.

```python
def calculate_mortgage_payment(principal, downpayment, interest,
        is_fixed, term, amortisation):
    ...
```

```python
calculate_mortgage_payment(principal=1_000_000, interest=0.052,
    downpayment=45_000, term=3, amortisation=35, is_fixed=True)
```

Higher Order Functions

- Functions usually return data.
- What if we return a function?



Create and return a function

```python
def create_greeting(person_name):
    def tell_person(message):
        print("Hey " + person_name + ". " + message)
    return tell_person
```

```python
x = create_greeting("Bobbie")
x("Want to join tomorrow?")
# Hey Bobbie. Want to join tomorrow?
x("Don't forget class Friday!")
# Hey Bobbie. Don't forget class Friday!
```

Recursion

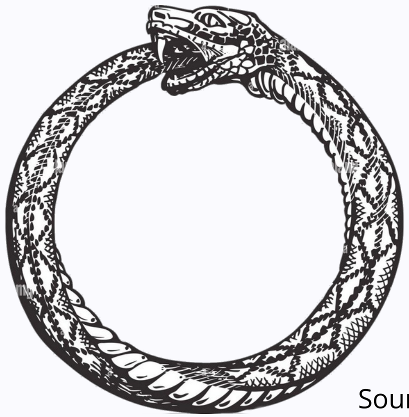
What happens if I call myself?

```python
def countdown(t):
  print(t)
  countdown(t-1)
```

```python
countdown(10)
```

```
10
9
...
-980
```

```
File "<python-input-2>", line 3, in countdown
  countdown(t-1)
  ~~~~~~~~~^^^^^
 [Previous line repeated 988 more times]
RecursionError: maximum recursion depth exceeded
```

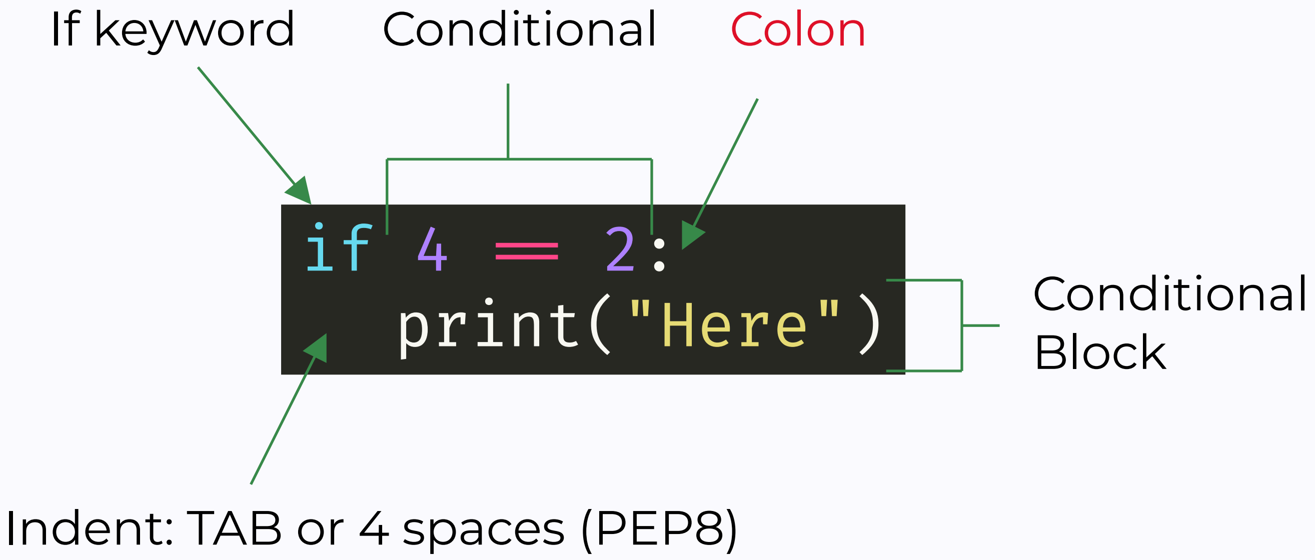


Source: Sergey Pykhonin

Conditionals

PCC 71-85

Conditionals protect the stack



```python
if 4 == 2:
    print("Here")
```

3 Different Conditional Statements

- if which executes its conditional block if its condition evaluates to True.
- elif acts like if but creates another execution path.
- else acts as a *catch-all* and executes its conditional block if it is reached.

```python
x = 4
if x < 4:
   print("Less than four")
elif x < 6:
   print("Less than six")
elif x <= 12:
   print("Less than or equal to 12")
elif x >= 13:
   print("Greater than or equal to 13")
else:
   print("I am something else")
```

Let’s practice conditionals

- Create the following function:
    I. age_group which takes in an age and if less than 2 returns “baby”, if between 2 and 4 return “toddler”, if greater than 4 and less than 12 return “kid”, if greater than or equal to 12 and less than 18 return “teen”, if greater than or equal to 18 and less than 65 return “adult”, if greater than or equal to 65 return “senior”.

And some more

- Create the following function:

I. even which takes a number and prints “is even” if it is even, otherwise “is odd”.

II. evenfy which takes a number and makes it even if it is odd by multiplying by 2.

III. pair which takes in two numbers and prints “Paired” if both are even or if both are odd; otherwise, “Failed to Pair”.

Let’s practice recursion

- Create the following functions:

  I. countdown which will countdown from a given number down to 0.

  II. factorial which returns the factorial of a given number.

  III. fibonacci which returns the i’th fibonacci number.