Data Structures and Algorithms

Live Class 2

Heikki Peura

h.peura@imperial.ac.uk

Today

1. **Recap**
2. Functions
3. A first algorithm
4. Homework 1

Announcements

- When you submit Session 1, `okpy.org` only shows you a text saying you have submitted. (OK transmits the command line exercises in a way that is not visible to you.)
- When you submit Session 2+, `okpy.org` will show you the functions you have completed in `ses01.py`. (Again not the command line exercises)
- HW deadlines are the ones in `syllabus.pdf` and on `okpy.org` – there's an error on the Hub.

Go to menti.com

What is the output?

```
x = 5
6 = y
print(x)
print(y)
```

A. 5, then 6

B. 5, then 5

C. 6, then 6

D. An error

E. I don’t know

What is the output?

```python
a = 2
b = a
a = 5
print(a)
print(b)
```

A. 2, then 2

B. 2, then 5

C. 5, then 2

D. 5, then 5

E. I don’t know

What is the output?

```python
x = 5
if x >= 0:
    print(1)
elif x < 20:
    print(2)
else:
    print(3)
print(4)
```

A. 1, then 2, then 4

B. 1, then 4

C. 4

D. 3, then 4

E. I don’t know

What is the output?

```python
x = 3
while x > 0:
    print(x)
    x = x - 1
```

A. 3, 2, 1

B. 3, 2, 1, 0

C. 3, 2

D. 2, 1, 0

E. I don’t know

What is the output?

```
x = 3
while x > 0:
    x = x - 1
print(x)
```

A. 3, 2, 1

B. 2, 1, 0

C. 1

D. 0

E. I don’t know

Today

1. Recap
2. **Functions**
3. A first algorithm
4. Homework 1

We have already been using functions

In Session 1, we used built-in functions:

```
>>> abs(-3)
3
>>> max(5, 3, 10)
10
>>> max(abs(-5), min(3, 9))
```

We say we **call** the function, specifying the arguments within parentheses.

What happens when we do this, and why are functions useful?

We use functions to organise tasks

A function is a named group of statements to perform a specific task.

- Input data → function → output data

We use functions to organise tasks

A function is a named group of statements to perform a specific task.

- Input data → function → output data

```python
# Let's define a function abs_value
def abs_value(a):
    if a < 0:
        return -a # The return statement stops function execution, outputs -a
    else:
        return a

# This function call runs the code block inside abs_value for a = -3
# The returned value is assigned to the variable y
y = abs_value(-3)
```

A function is like a **factory**: in goes input data (car parts), out comes output data (car).

A function may have multiple parameters separated by commas. It may return multiple values separated by commas.

Why functions?

1. **Abstraction**: user does not need to know what happens inside
2. Make **code easily re-usable** and modular
3. **Changing code becomes easier**: we don't have to copy same code in many places

Do we need to return something?

```python
def abs_value(a):
    if a < 0:
        return -a # We can use the value later if we return it
    else:
        return a

def print_abs_value(a):
    # We perform the same task and display the value to the user
    # But we cannot use it later
    if a < 0:
        print(-a) # only displays information for user
    else:
        print(a)

print_abs_value(4)
# Any function by default returns the special value None, a null value
x = print_abs_value(4)
```

Sometimes we don't need to use the results later. We may for instance just display them to the user. `return` is like getting a car from the factory. `print` is like peeking through the window to see what happens inside.

Local variables and scope

```python
def greeting(n):
    # Variables defined here are local in scope
    message = 'Hi ' + n
    return message

# Variables defined here are global in scope
name = 'Xue'

def print_message(msg):
    # There is no locally defined variable "name"
    # So it will be looked up from global scope
    print(msg + ', ' + name)

g = greeting(name)
print_message('Hello')
print(message) # An error because this is not globally accessible
```

The **scope** of a variable is the part of a program from which it can be accessed. Global variables can be accessed from anywhere, local ones only within the function (call).

Go to menti.com

What is the output?

```python
def hello():
    print("Hello!")
def goodbye():
    print("Goodbye!")

hello()
```

A. Hello

B. Goodbye

C. Both of them

D. Neither of them

E. I don’t know

What is the output?

```python
def add_one(x):
    y = x + 1

z = add_one(5)
print(z)
```

A. 5

B. 6

C. An error

D. None

E. I don’t know

What is the output?

```python
def add_three(x, y, z):
    return x + y + z

x = 5
y = 2
z = add_three(2, x, x)
print(x, y, z)
```

A. 5 5 5

B. 5 2 5

C. 2 2 12

D. 5 2 12

E. I don’t know

What is the output?

```python
def calc_1(x):
    a = x - 1
    return a

def calc_2(z):
    x = 5
    return calc_1(3)

calc_2(4)
```

A. 2

B. 3

C. 4

D. 5

E. I don’t know

Today

1. Recap
2. Functions
3. **A first algorithm**
4. Homework 1

Solving computational problems

Data = digitised information

Data structures describe ways to organise data

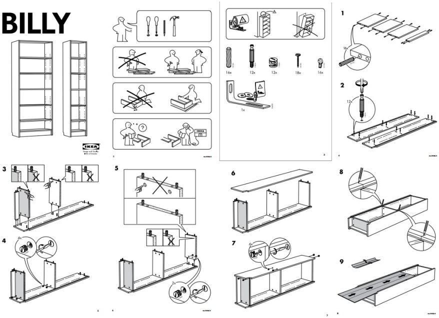
Algorithms describe how we process data:

- Step-by-step instructions
- Take input data and produce output data

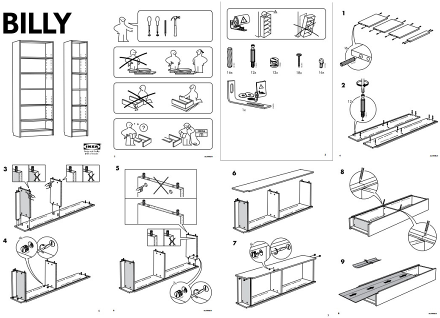
We write algorithms into **programs** (eg in Python)

Computers interpret and execute programs

An algorithm is a recipe



An algorithm is a recipe



Algorithm:

- Step-by-step instructions
- Takes input (data) and produces output (data)

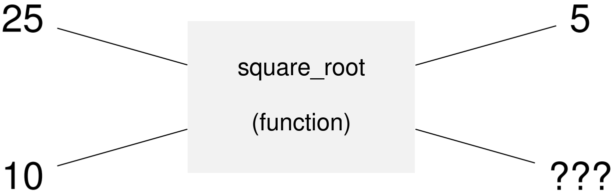
Pics: Hungry Gals, IKEA.

How do you calculate a square root?

The square root of a number x is a number y such that $y \times y = x$

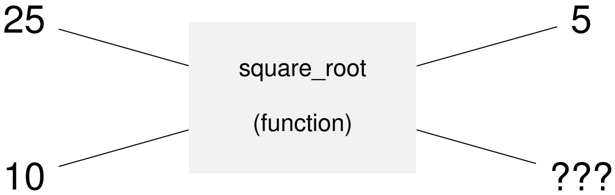
How do you calculate a square root?

The square root of a number x is a number y such that $y \times y = x$



How do you calculate a square root?

The square root of a number x is a number y such that $y \times y = x$



A function is like a factory

- In goes number, out comes square root
- Inside the factory, there's an **algorithm**

Square-root algorithm

The square root of x is y such that $y \times y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$

Square-root algorithm

The square root of x is y such that $y \times y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Repeat three times:

Square-root algorithm

The square root of x is y such that $y \times y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Repeat three times:
    - Divide the original number x by the guess to get a ratio

Square-root algorithm

The square root of x is y such that $y \times y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Repeat three times:
  - Divide the original number x by the guess to get a ratio
  - Find the average of the guess and the ratio

Square-root algorithm

The square root of x is y such that $y \times y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Repeat three times:
    - Divide the original number x by the guess to get a ratio
    - Find the average of the guess and the ratio
    - Use this average as the next guess

Let’s use Python

Square-root function

```python
def square_root(x):
    guess = x/2
    eps = 0.01
    while abs(guess*guess-x) >= eps:
        guess = (guess + x/guess)/2
    return guess

z = 20
y = square_root(z)
```

Square-root function

```python
def square_root(x):
    guess = x/2
    eps = 0.01
    while abs(guess*guess-x) >= eps:
        guess = (guess + x/guess)/2
    return guess

z = 20
y = square_root(z)
```

- Takes input *x* and outputs its square root
- Note: uses another function inside it: built-in function `abs`
- Abstraction, reusability, reliability

Today

1. Recap
2. Functions
3. A first algorithm
4. **Homework 1**

Homework 1

1. Download hw01.zip from the Hub and open the HTML file in the zip for instructions
2. 10 % of your grade, due 30 September
3. Grade is based on correctness: whether your code works for different inputs
4. **Individual** programming exercises - the College takes plagiarism very seriously
5. **Do not share your solution to any HW assignment with anyone.** If someone submits work similar to yours, it does not matter whether you wrote it first.
6. The tutors will not answer questions about the homework
7. Some questions require things we will learn next week

Review

Algorithms are recipes for solving problems

We divide programs into named functions:

- Reusable code
- User can “just use” the function

Up next:

- Session 2: Review exercises on working with functions
- Session 3: Working with for loops and lists