Finding Easter

An introduction to algorithms & Python

Stuart Allen

School of Computer Science & Informatics
Cardiff University

CM1103

Algorithms

Definition

An *algorithm* is a procedure for solving a problem that specifies the actions to be executed and the order in which they should be executed.

Definition

Pseudocode is an informal language similar to everyday English used to describe algorithms.

Example problem: Easter Sunday

- The canonical rule is that Easter Sunday is the first Sunday after the 14th day of the lunar month (the nominal full moon) that falls on or after 21 March (nominally the day of the vernal equinox)
  - In what proportion of years does Easter Sunday fall in March?
  - When will Easter Sunday next fall on March 22nd?
- There is an algorithm to find date of Easter Sunday given the year

Quotient & Remainder

21 // 4 = 5

e.g.: 21/4

5 -> quotient
4 -> remainder

Definition

Quotient is the **integer** part of the result of dividing two integers.

Remainder is the amount "left over" after the division of two integers which cannot be expressed with an integer quotient.

Algorithm to find Easter

Given the year y:

Gauss's algorithm to find Easter Sunday: Require: year y

Divide y by 19 and call the remainder a (ignore the quotient)
Divide y by 100 and call the quotient b and the remainder c
Divide b by 4 and call the quotient d and the remainder e
Divide $8b + 13$ by 25 and call the quotient g (ignore the remainder)
Divide $19a + b - d - g + 15$ by 30 and call the remainder h (ignore the quotient)
Divide c by 4 and call the quotient j and the remainder k
Divide $a + 11h$ by 319 and call the quotient m (ignore the remainder)
Divide $2e + 2j - k - h + m + 32$ by 7 and call the remainder r (ignore the quotient)
Divide $h - m + r + 90$ by 25 and call the quotient n (ignore the remainder)
Divide $h - m + r + n + 19$ by 32 and call the remainder p (ignore the quotient)

Easter falls on day p of month n

Variables & type

Definition

value One of the basic units of data, like a number or string, that a program manipulates.

variable A name that refers to a value.

type A category of values.

See Think Python! section 2.11

type(x)
-> returns the type of variable in Python

Python is a strongly typed language

Using Python to calculate Easter Sunday for a given year y

```
>>> y = 2010
>>> a = y % 19
>>> b = y // 100
>>> c = y % 100
>>> d = b // 4
>>> e = b % 4
>>> g = (8 * b + 13) // 25
>>> h = (19 * a + b - d - g + 15) % 30
>>> j = c // 4
>>> k = c % 4
>>> m = (a + 11 * h) // 319
>>> r = (2 * e + 2 * j - k - h + m + 32) % 7
>>> n = (h - m + r + 90) // 25
>>> p = (h - m + r + n + 19) % 32
>>> p
4
>>> n
4
```

Operator precedence

- Why `(8 * b + 13) // 25` and not `8 * b + 13 // 25`?
- http://docs.python.org/reference/expressions.html#operator-precedence gives the operator precedence in Python
- **Brackets** evaluated before ** before *, /, % before +, −

2**3 = 8

Using Python to calculate Easter Sunday for a given year y

```
>>> y = 2010
>>> a = y % 19
>>> b = y // 100
>>> c = y % 100
>>> d = b // 4
>>> e = b % 4
>>> g = (8 * b + 13) // 25
>>> h = (19 * a + b - d - g + 15) % 30
>>> j = c // 4
>>> k = c % 4
>>> m = (a + 11 * h) // 319
>>> r = (2 * e + 2 * j - k - h + m + 32) % 7
>>> n = (h - m + r + 90) // 25
>>> p = (h - m + r + n + 19) % 32
>>> p
4
>>> n
4
```

Defining a function

Definition

function: A named sequence of statements that performs some useful operation. Functions may or may not take arguments and may or may not produce a result.

function call: A statement that executes a function. It consists of the function name followed by an argument list.

Encapsulation: Hiding implementation in a function

Function to print date of Easter Sunday for given year y

```
def Easter(y):
    a = y % 19
    b = y // 100
    c = y % 100
    d = b // 4
    e = b % 4
    g = (8 * b + 13) // 25
    h = (19 * a + b - d - g + 15) % 30
    j = c // 4
    k = c % 4
    m = (a + 11 * h) // 319
    r = (2 * e + 2 * j - k - h + m + 32) % 7
    n = (h - m + r + 90) // 25
    p = (h - m + r + n + 19) % 32

    print(str(p) + "/" + str(n) + "/" + str(y))
```

Calling our functions from interactive mode

- Save function definition in a file e.g. `ImportantDates.py`
- `import` module (i.e. filename without `.py`)
- Call method using module name (e.g. `ImportantDates.Easter(2010)`)
- If we make changes to our module, save the file and `reload` from interactive mode

Defining functions

Definition

function definition: A statement that creates a new function, specifying its name, parameters, and the statements it executes.

header: The first line of a function definition.

body: The sequence of statements inside a function definition.

argument: A value provided to a function when the function is called. This value is assigned to the corresponding parameter in the function.

math.sqrt(4)
-> 4 is the argument

parameter: A name used inside a function to refer to the value passed as an argument.

def printAge(age):
-> age is parameter

Local variables are stored in functions, cannot be accessed by outside

Function to print date of Easter Sunday for given year y

```
def Easter(y):
    a = y % 19
    b = y // 100
    c = y % 100
    d = b // 4
    e = b % 4
    g = (8 * b + 13) // 25
    h = (19 * a + b - d - g + 15) % 30
    j = c // 4
    k = c % 4
    m = (a + 11 * h) // 319
    r = (2 * e + 2 * j - k - h + m + 32) % 7
    n = (h - m + r + 90) // 25
    p = (h - m + r + n + 19) % 32

    print(str(p) + "/" + str(n) + "/" + str(y))
```

Returning values from functions

Definition

local variable: A variable defined inside a function. A local variable can only be used inside its function.

return value: The result of a function. If a function call is used as an expression, the return value is the value of the expression.

Making the function more useful

- It is more useful to `return` a value from the function
- What type should we return?
- There is a built in module `datetime` that defines the `datetime` type
- The `datetime` type represents dates & times!!!
- Need `import datetime` to use

Using documentation

- `type(x)` – what type is `x`?
- `dir(x)` – what can we do to `x`?
- `help(x)` – (brief) help about `x`.
- `help(x.function)` – (brief) help about the function.

Function to **return** date of Easter Sunday for given year y

```
import datetime

def Easter(y):
    a = y % 19
    b = y // 100
    c = y % 100
    d = b // 4
    e = b % 4
    g = (8 * b + 13) // 25
    h = (19 * a + b - d - g + 15) % 30
    j = c // 4
    k = c % 4
    m = (a + 11 * h) // 319
    r = (2 * e + 2 * j - k - h + m + 32) % 7
    n = (h - m + r + 90) // 25
    p = (h - m + r + n + 19) % 32

    return datetime.date(year=y,month=n,day=p)
```

Testing

- How do we know our algorithm/implementation works as it should?
- We can **test** it using inputs where we know the correct answer

doctest

- `doctest` lets us test our code by embedding examples in the **comments** for a module
- The examples show the results of an interactive session using the function

Doctest example

```
def product(a, b):
    """
    >>> product(5, 6)
    30
    """
    return a * b
```

Running doctests

We can execute these tests:

- Inside the interpreter, by the code:

```
>>> import doctest
>>> import module_name
>>> doctest.testmod(module_name, verbose=True)
```

- At the command prompt, by entering the following command:

```
python -m doctest -v module_name.py
```

Distribution of Easter

- In what proportion of years does Easter Sunday fall in March?
- Calculate Easter Sunday **for** each year in a given range and check **if** the month is equal to 3 (for March)

Next Easter Sunday on April 1st

- When will Easter Sunday next fall on April 1st?
- Set *year* to be 2019, repeatedly calculate Easter Sunday and add one to year *until* Easter Sunday falls on April 1st
- I.e. repeat **while** the answer *is not* April 1st

Program control

- Fundamental operations of algorithms are statements, loops & decision
- The two main *looping* constructs in Python are `for` and `while` (see Think Python! sections 4.2 and 7.3)
- `if` is the main statement used in Python to select from alternative paths in the code (see Think Python! section 5.4)

Another example: finding square roots

- The square root of a number x is a number r such that $r^2 = x$ ($r \times r = x$)
- Every positive number has a positive and negative square root (the positive square root is called the *principal* and denoted by $\sqrt{x}$)
- E.g. $\sqrt{9} = 3$, $\sqrt{49} = 7$, $\sqrt{3697929} = 1923$, $\sqrt{2} = 1.414\ldots$

Algorithm

Babylonian Algorithm: Require: real number S

Choose a starting value x_0

Let x_{n+1} be the average (mean) of x_n and S/x_n

Repeat step 2 until desired accuracy is reached

or

Babylonian Algorithm: Require: real number S

Choose a starting value x_0

Set $x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right)$

Repeat step 2 until desired accuracy is reached

Summary

You should be able to:

- Use Python interactive interpreter for simple calculations
- Implement simple algorithms (i.e. without loops or branching) in Python
- Write simple functions in Python
- `import` and call functions defined in a Python file
- Understand the purpose of `for`, `while` and `if` statements
- Check whether a function satisfies the required `doctest`
- Be able to provide definitions for the terms listed on the following slide

Definitions covered (see Think Python!)

- value
- variable
- type
- function
- function call
- function definition
- header
- body
- argument
- parameter
- local variable
- return value

Optional further work

- `iCalendar` is a (fairly) widely used file format for describing calendar events, that can be used by a range of calendar applications (e.g. from Google or Apple)
- There is a Python module to parse & generate iCalendar files: https://pypi.python.org/pypi/icalendar
- Try installing this package and use the Gauss algorithm to generate an iCalendar file that gives the dates of Easter Sunday for the next 5, 10 or 50 years
- Try to add *Good Friday* (the Friday before Easter Sunday) and *Easter Monday* (the Monday after Easter Sunday) to this calendar (the `datetime` package may be useful here)