Python for Beginners

Pepe García

2020-04-20

Python for Beginners, session 1

About

The Professor

Pepe García

About

The Professor

Pepe García
jgarciah@faculty.ie.edu

About

The Professor

Pepe García
jgarciah@faculty.ie.edu
Tech Lead @ 47deg

About

The Professor

Pepe García
jgarciah@faculty.ie.edu
Tech Lead @ 47deg
Ask me anything

About

The Course

- 7 sync sessions

About

The Course

- 7 sync sessions
- 3 async sessions

Learning Objectives

- Learn What's programming

Learning Objectives

- Learn What's programming
- Understand how computers execute programs

Learning Objectives

- Learn What's programming
- Understand how computers execute programs
- Learn the basics of Python

Plan for this session

- Learn about software

Plan for this session

- Learn about software
- Understand what are algorithms

Plan for this session

- Learn about software
- Understand what are algorithms
- Understand what are data structures

Language

Throughout this course we will use Python as our programming language, but there are many more!

Language

There are several ways for categorizing programming languages.

Language classification

Language	Paradigm	Execution	Purpose
Python	imperative	interpreted	general
Java	object oriented	compiled	general
Javascript	imperative	interpreted	general
Haskell	functional	compiled	general
SQL	declarative	interpreted	specific
HTML	declarative	interpreted	specific

Language

Python

Python is one of the most used languages right now. Its applications range from Data Science to Web servers

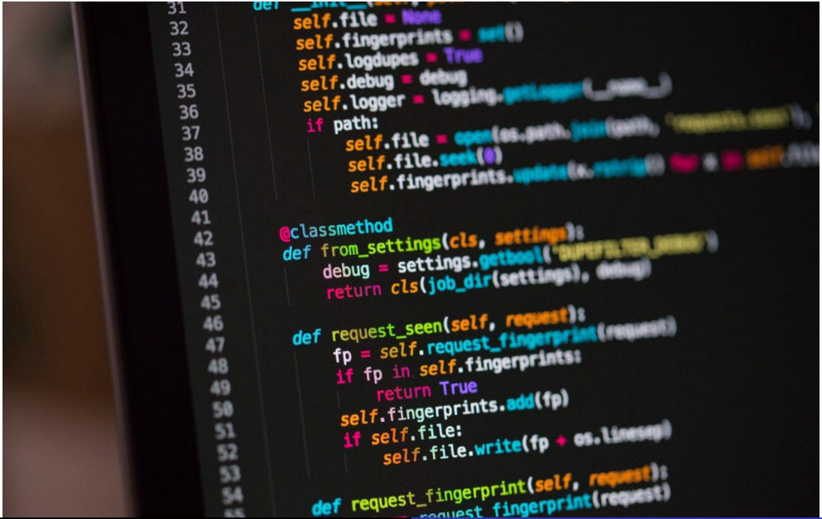
How do we write code?

How do we write code?

Coding is basically putting words together following a programming language specification.

How do we write code?

We can put these words directly in a text file and then execute it as a program.



```
        self.file = None
        self.fingerprints = set()
        self.logdupes = True
        self.debug = debug
        self.logger = logging.getLogger(__name__)
        if path:
            self.file = open(os.path.join(path, 'requests.seen'), 'a+')
            self.file.seek(0)
            self.fingerprints.update(x.rstrip() for x in self.file)

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('DUPEFILTER_DEBUG')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

How do we write code?

Or we can feed these words directly into the programming language console.

Demo

Spyder

Spyder is the editor we will use for writing Python code.
Let's open it and try some expressions in the console.

Programs

Programs

What is a program?

A program is a piece of software with a specific task.

Programs

What is a program?

A program is a piece of software with a specific task.
This task can be something BIG, like handling a nuclear reactor, or something small like Ctrl-c/Ctrl-v.

Programs

What is a program?

A program is a piece of software with a specific task.
This task can be something BIG, like handling a nuclear reactor, or something small like Ctrl-c/Ctrl-v.
There are two main components of programs, **algorithms** & **data structures.**

Algorithms

Algorithms

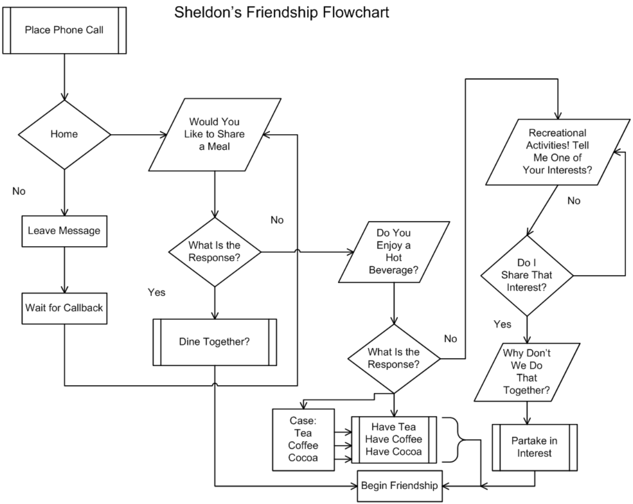
What is an algorithm?

Algorithms

What is an algorithm?

An algorithm is a sequence of steps that guide the computer in how to solve a problem

Algorithms


Sheldon's Friendship Flowchart

Place Phone Call
Home
No
Leave Message
Wait for Callback
Would You Like to Share a Meal
What Is the Response?
No
Yes
Dine Together?
Do You Enjoy a Hot Beverage?
What Is the Response?
No
Case: Tea Coffee Cocoa
Have Tea Have Coffee Have Cocoa
Begin Friendship
Recreational Activities! Tell Me One of Your Interests?
No
Do I Share That Interest?
Yes
Why Don't We Do That Together?
Partake in Interest

link to the video

Algorithms

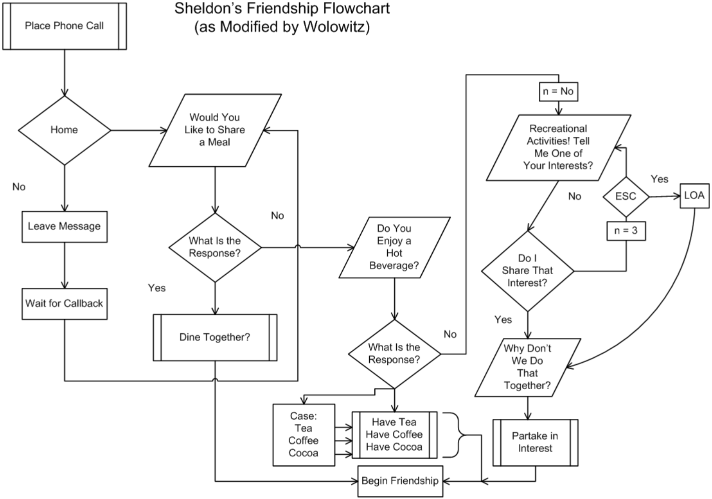
What's wrong with this algorithm? why did Wolowitz needed to fix it?

Algorithms

What's wrong with this algorithm? why did Wolowitz needed to fix it?

There was an infinite loop, a **bug**

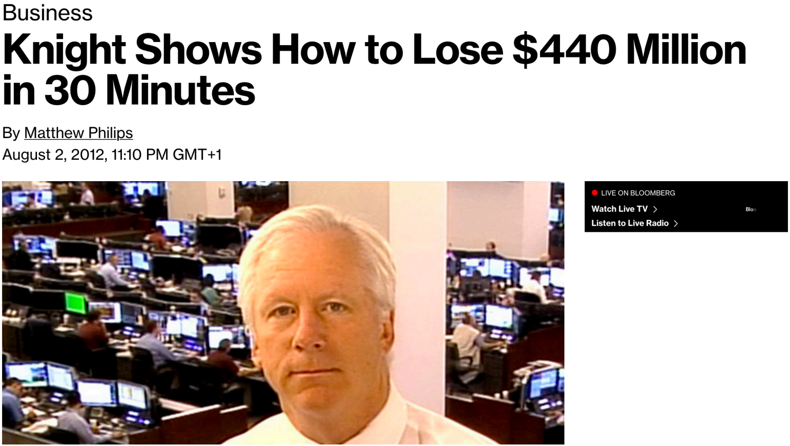
Algorithms


Sheldon's Friendship Flowchart
(as Modified by Wolowitz)
Place Phone Call
Home
No
Leave Message
Wait for Callback
Would You Like to Share a Meal
No
What Is the Response?
Yes
Dine Together?
Do You Enjoy a Hot Beverage?
What Is the Response?
No
Case: Tea Coffee Cocoa
Have Tea Have Coffee Have Cocoa
Begin Friendship
n = No
Recreational Activities! Tell Me One of Your Interests?
No
ESC
Yes
LOA
n = 3
Do I Share That Interest?
Yes
Why Don't We Do That Together?
Partake in Interest

Algorithms

What other cases of bugs do we know?

Algorithms



Business

Knight Shows How to Lose $440 Million in 30 Minutes

By Matthew Philips
August 2, 2012, 11:10 PM GMT+1

LIVE ON BLOOMBERG
Watch Live TV >
Listen to Live Radio >

https://www.bloomberg.com/news/articles/2012-08-02/knight-shows-how-to-lose-440-million-in-30-minutes

Checkpoint

Are there any questions/comments so far?

10 minutes break

Python for Beginners, session 2

Python for Beginners, session 2

Plan for this session

- Python basic datatypes

Python for Beginners, session 2

Plan for this session

- Python basic datatypes
- variables

Python for Beginners, session 2

Plan for this session

- Python basic datatypes
- variables
- operators

Datatypes

Datatypes tell Python how we want to use the data. There are several primitive data types in Python such as bool, int, str, float.

Datatypes

Integers

Integers (or ints) represent whole numbers without decimal parts. We create them by using their numeric representation directly

```
1
234
432432
```

Datatypes

Integers

Integers (or ints) represent whole numbers without decimal parts. We create them by using their numeric representation directly

```
1
234
432432
```

Demo

Datatypes

Floating point numbers

floats represent numbers that have a fractional part. We use a dot to separate the integer and fractional parts

```
3.14
1.0
33.33
```

Datatypes

Floating point numbers

floats represent numbers that have a fractional part. We use a dot to separate the integer and fractional parts

```
3.14
1.0
33.33
```

Demo

Datatypes

Strings

Strings are used for textual representation. They can be created using either double or simple quotes.

```
'this is a string'
"this is another string"
```

Datatypes

Strings

Strings are used for textual representation. They can be created using either double or simple quotes.

```
'this is a string'
"this is another string"
```

Demo

Datatypes

Booleans

Booleans represent truthiness. There are only two values in for the bool type in Python: True and False

```
True
False
```

Datatypes

Booleans

Booleans represent truthiness. There are only two values in for the bool type in Python: True and False

```
True
False
```

Demo

Getting the type of a value

We can always ask ask a value for its type using the **type(value)** function

```
type("patata")
```

Getting the type of a value

Inside Spyder, check what's the type of the following expressions:

```
"there is some text here"
1
True
44.4
'true'
'False'
2
'33.3'
```

Does someone want to do it?

Operators

Operators are symbols in the language that perform different kinds of computations on values

They're binary

Arithmetic Operators

symbol	meaning
+	sum
-	substraction
*	multiplication
/	division
**	exponentiation
//	floored division
%	modulus

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division
- Sum/Substraction
- when operators have the same precedence, evaluate left to right

Arithmetic Operators

Rules of precedence

- Parentheses
- Exponentiation
- Multiplication/Division
- Sum/Substraction
- when operators have the same precedence, evaluate left to right

Demo

String operators

sum and multiplication operators work on strings too. They're used to concatenate and multiply strings, respectively.

String operators

sum and multiplication operators work on strings too. They're used to concatenate and multiply strings, respectively.

Demo

Variables

Variables are names that point to values in Python.

Variables

Naming variables

It's important to be as descriptive as possible when naming variables

There are some naming rules we should obey

Variables

Naming rules

- variable names can’t start with a number

Variables

Naming rules

- variable names can't start with a number
- variable names can't contain special characters such as `!`, `@`, `.`

Variables

Naming rules

- variable names can't start with a number
- variable names can't contain special characters such as `!`, `@`, `.`
- Can't be one of the reserved words

Variables

Reserved words

and
as
assert
break
class
continue
def
del
elif
else
except
False
finally
for
from
global
if
import
in
is
lambda
None
nonlocal
not
or
pass
raise
return
True
try
while
with
yield

Variables

Mutability

In Python variables are mutable. This means that we can change their value at any time

```python
name = "Pepe"
print(name)

name = "Jose"
print(name)
```

Converting values

There are some times when we need to convert a value from one type to another.

We use the **int()**, **bool()**, **str()**, and **float()** functions for that

Converting values

There are some times when we need to convert a value from one type to another.

We use the **int()**, **bool()**, **str()**, and **float()** functions for that

```
int('23')
bool(1)
bool(0)
str(True)
float("3.2")
```

Printing output

One can print output using the **print()** function

User input

There is a handy function **input()** that allows us to capture input from the user

```
name = input("Tell me your name: ")

print("hello, " + name)
```

Recap

- Datatypes (int, float, bool, str)

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)
- Converting values

Recap

- Datatypes (int, float, bool, str)
- Variables (naming, mutability)
- Operators (arithmetic, precedence, string operators)
- Converting values
- User input

Exercises for the Async session

1. Create a program that calculates the total number of seconds in an hour and prints it afterwards.

Exercises for the Async session

1. Create a program that calculates the total number of seconds in an hour and prints it afterwards.
2. How does the following expression evaluate? 2 + (3 + 4) + (5 * 33 ** 34)

Exercises for the Async session

1. Create a program that calculates the total number of seconds in an hour and prints it afterwards.
2. How does the following expression evaluate? 2 + (3 + 4) + (5 * 33 ** 34)
3. Create a program that asks the user for their age and their mother's age and calculate the age difference

Exercises for the Async session

1. Create a program that calculates the total number of seconds in an hour and prints it afterwards.
2. How does the following expression evaluate? $2 + (3 + 4) + (5 * 33 ** 34)$
3. Create a program that asks the user for their age and their mother's age and calculate the age difference
4. Make the following expressions work (use Python console for this one)

```
3 + "3"
'there are ' + 4 ' dogs barking'
```

Reading for the async session

What Is Code is a great essay by Paul Ford. It's from 2015 but has aged (and will age) very well.

https://www.bloomberg.com/graphics/2015-paul-ford-what-is-code/