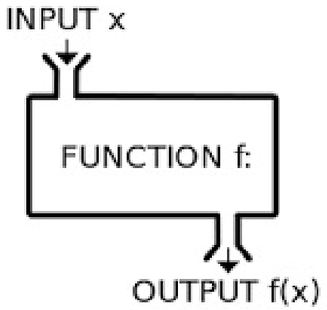
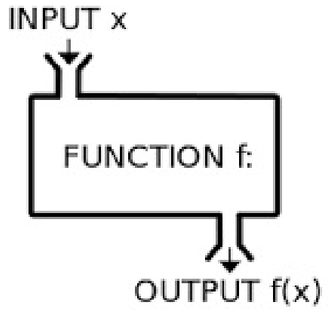
Functional Python



Python Grenoble Meetup, 18 Jan 2017

Mike Bright, @mjbright

Functional Python - Tour de Table



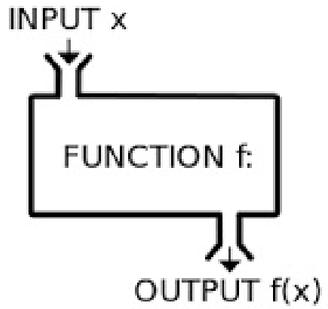
- Who are you?
- Do you know or use Functional Programming?
  - in what languages?

Please Challenge me !!

I'm not an FP expert

Definitions of FP are discutable

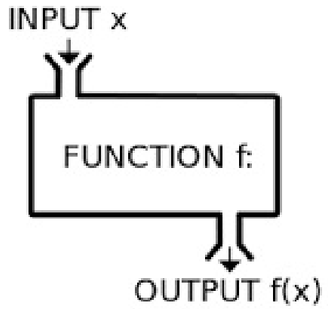
What is Functional Programming?



Wikipedia Functional Programming

There is no absolute definition (but common principles).

What is Functional Programming?

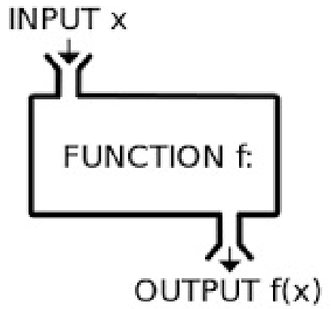


Wikipedia Functional Programming

There is no absolute definition (but common principles).

Basic principle is functions without side effects or state (variables) - unlike Imperative Programming.

What is Functional Programming?



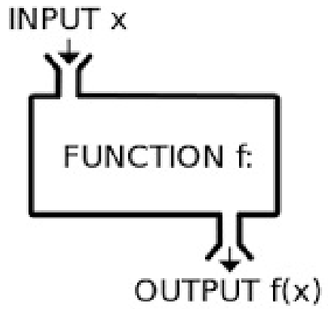
Wikipedia Functional Programming

There is no absolute definition (but common principles).

Basic principle is functions without side effects or state (variables) - unlike Imperative Programming.

A pure function takes a set of arguments (x) and returns a set of values (y). Calling the function with the same arguments will always produce the same result.

What is Functional Programming?



Wikipedia Functional Programming

There is no absolute definition (but common principles).

Basic principle is functions without side effects or state (variables) - unlike Imperative Programming.

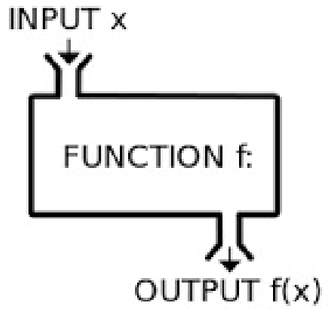
A pure function takes a set of arguments (x) and returns a set of values (y). Calling the function with the same arguments will always produce the same result.

It has no side effects on it's environment

- no reassignment of variables (not variable!)
- no prints, or writes to disk !!

This doesn't sound very useful though.

What is Functional Programming? - 2



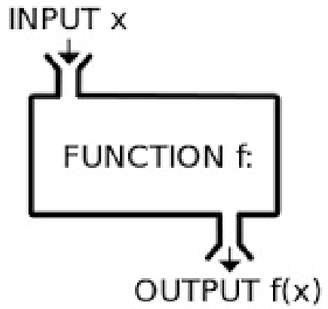
Wikipedia Functional Programming

Looking at FP concepts as per wikipedia page

Functional languages provide support for

- 1st-class and higher-order functions
  - functions are objects, can be passed as arguments
  - can return new functions from a function
- Pure functions
  - no side effects (memory or I/O)
  - can be reasoned about
    - provable results
    - optimizations possible
- Recursion
  - "simulate looping" by accumulating results
  - Tail recursion optimization possible (compiler)
  - Higher order functions can factor out recursion

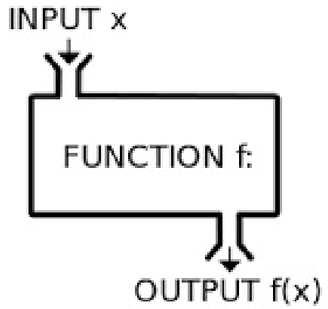
What is Functional Programming? - 3



Wikipedia Functional Programming

- Strict (eager) versus non-strict (lazy) evaluation
  - Should `length([2+1, 3*2, 1/0, 5-4])` complete?
  - lazy is default in some languages (e.g. Haskell)
- Type systems
  - tend to be strongly typed (may be implicit)
    - static (ML, Scala, Haskell) or
    - dynamic (JS, Lisp, Python)
  - Static typing can facilitate mathematical proofs
  - "case classes"
- Referential transparency
  - No state is modified: functions are transparent
  - `x = x * 10` is not legal
  - Functions return/accumulate new values

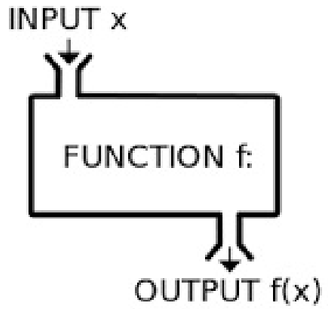
What is Functional Programming? - 4



Wikipedia Functional Programming

- Data Structures
  - Linked-lists rather than arrays (random access)
  - Operations on lists such as
    - filter, map, reduce, flatMap
- Monads (Collections with certain operations)
  - filter, map, flatMap
  - used by some languages to handle (isolate) stateful operations such as i/o

Functional Programming Languages



In practice languages are more or less "functional".

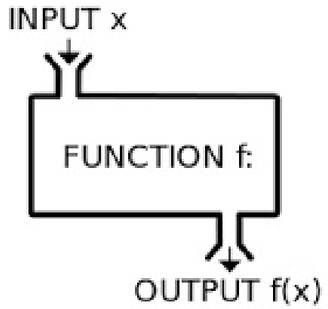
Some languages such as Haskell are closer to the "ideal" of a pure functional language.

Pure languages have the advantage of being provably correct.

Other languages mix paradigms.

- Scala: OOP, FP
- Python: OOP, FP
- JavaScript: "OOP", FP
- Java: OOP, FP

Functional Programming Languages



In practice languages are more or less "functional".

Some languages such as Haskell are closer to the "ideal" of a pure functional language.

Pure languages have the advantage of being provably correct.

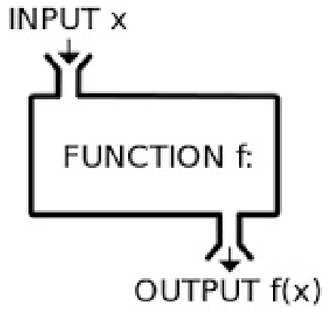
Other languages mix paradigms.

- Scala: OOP, FP
- Python: OOP, FP
- JavaScript: "OOP", FP
- Java: OOP, FP

Let's compare several languages.

Wikipedia Functional Prog Lanugaues comparison

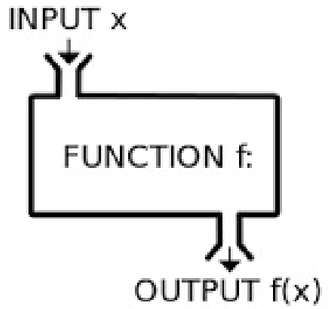
Functional Programming Languages - 2



The Functional paradigm can be mixed with others whilst encouraging good programming practices.

It facilitates understanding and debugging of code.

Functional Programming Languages - 2



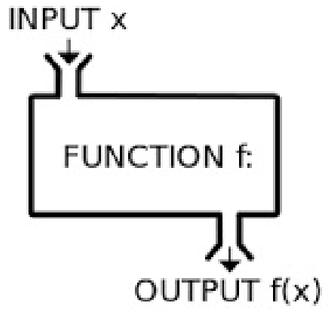
The Functional paradigm can be mixed with others whilst encouraging good programming practices.

It facilitates understanding and debugging of code.

Note: Some say that Object-Oriented Programming is contradictory to Functional Programming because of it's focus on instance variables and the use of getters and setters.

Others see this as a hierarchy of abstractions FP above OOP above libraries (procedures).

Functional Programming Languages - 3



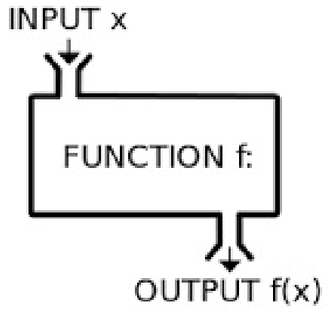
StackOverflow: Functional programming vs Object Oriented programming

- OO: good when fixed set of operations on things, and you primarily add new things.
- FP: good when fixed set of things, and you primarily add new operations on existing things.

But when you need to add new operations to OO, or new things to FP we encounter the "expression problem".

Scala provides a very usable mix of both paradigms and mixins help address the "expression problem".

Functional Python



Python mixes several programming styles such as Imperative, OOP, Functional

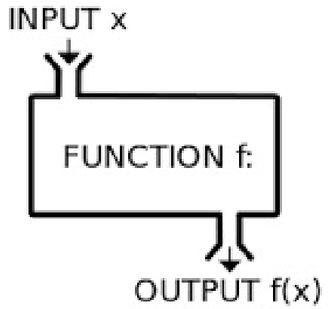
Python had first class functions from its' inception.

Support later added for "lambda", "map", "reduce", and "filter" in 1994, as well as closures in Python 2.2.

- lambda functions are convenient unnamed functions
  - declared in a single line (or argument)
  - are pure functions, e.g.
    - `incr = lambda x: return x+1`
- `map(fn, iterable, ...)`
  - Apply fn to each item, return result elements
- `reduce(fn, iterable, [init])`
  - Apply fn to each item, combining result elements
- `filter(fn, iterable)`
  - Use fn to select or not each item, return result elements

Python 3 relegated "reduce" to the functools standard library module.

Functional Programming in Python



Python encourages functional programming by

- Tuples, Namedtuples immutable data types.
- List and Dictionary Comprehensions
  - `[ x*2 for x in range(3)]` ==> `[ 0 1 4 ]`
  - `{ x: x*2 for x in range(3)}` ==> `[ 0 1 4 ]`
- Generator Comprehensions (lazy evaluation)
  - `( x*2/0 for x in range(3) )`
- Monads
  - PyMonad module

Functional Python - PacktPub Index

REMOVE

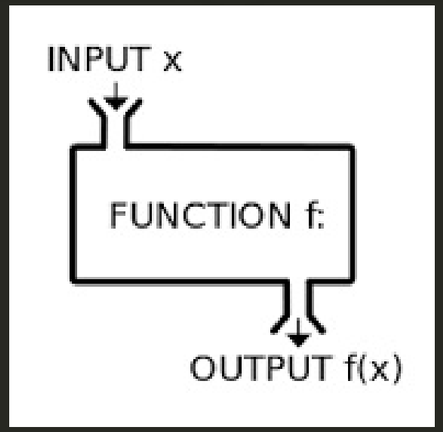
- Functional Programming
- Functional Programming Features
  - 1st-class functions
  - Immutable data
  - Strict and non-strict evaluation
  - Recursion
  - Functional type systems
  - Advanced concepts
- Functions, Iterators and Generators
- Collections
- Higher-order Functions
- Recursions and Reductions
- Additional Tuple Techniques
- Itertools Module
- Functools Module
- Decorator Design Techniques
- Multiprocessing and Threading Modules
- Conditional Expressions and Operator Modules
- The PyMonad Library
- A Functional approach to Web Services
- Optimizations and Improvements

Functional Python - OReilly Index

REMOVE

- What is Functional Programming ?
- Avoiding Flow Control
  - Encapsulation
  - Comprehensions
  - Recursion
  - Eliminating Loops
- Callables
  - Named Functions and Lambdas
  - Closures and Callable Instances
  - Methods of Classes
  - Multiple Dispatch
- Lazy Evaluation
  - The Iterator Protocol
  - Module: Itertools
- Higher-order Functions
  - Utility Higher-order Functions
  - The operator module
  - The functools module
  - Decorators

Functional Python Demo

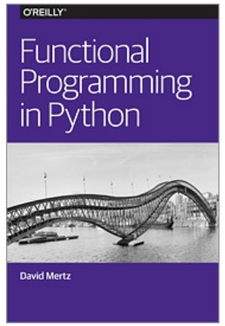


Notebook

Resources

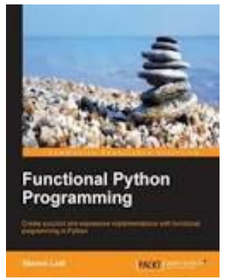
python.com
Functional Programming HOWTO

HOWTO



OReilly
"Functional Programming in Python"

Free download



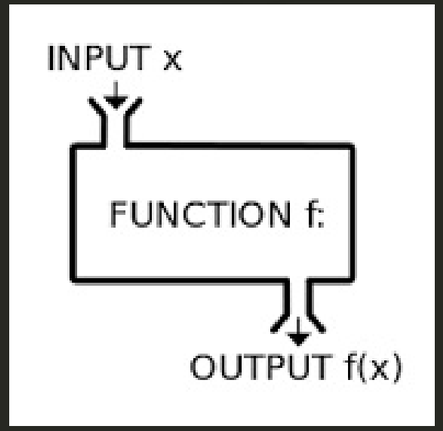
PacktPub
"Functional Python Programming"

Info

github.com
Awesome Functional Python

Sources

Thank you !



Questions?