What is the difference between a “function” and a “procedure”?

Generally speaking, we all hear about the *functions* or *procedures* in programming languages. However, I just found out that I use these terms almost interchangeably (which is probably very wrong).

So, my question is:

> What is the difference in terms of their functionality, their purpose and use?

An example would be appreciated.

function terminology procedure

edited Jan 4 at 0:54 — nbro — 4,207 4 18 54

asked Apr 6 '09 at 11:48 — rpr — 1,473 2 14 19

BTW You forgot sub-routine – Rob Wells Apr 6 '09 at 12:18

See also: stackoverflow.com/q/10388393/974555 – gerrit Jan 29 '13 at 22:28

3 I think SICP gets this right. Functions exist only in math, and they represent *what is* knowledge. Procedures exist in programming languages (including functional ones), and they represent *how to* knowledge. **Function**: sqrt(x) = the y such that y^2=x. **Procedure**: `(define (sqrt x) (newtons-method (lambda (y) (- (square y) x)) 1.0))` . – mk12 May 28 '14 at 19:47

16 Answers

A function returns a value and a procedure just executes commands.

The name function comes from math. It is used to calculate a value based on input.

A procedure is a set of command which can be executed in order.

In most programming languages, even functions can have a set of commands. Hence the difference is only in the returning a value part.

But if you like to keep a function clean, (just look at functional languages), you need to make sure a function does not have a side effect.

edited Apr 6 '09 at 11:57

answered Apr 6 '09 at 11:52 — Toon Krijthe — 42.7k 20 113 180

Thats what I was going to say... :) – Nick Haslam Apr 6 '09 at 11:53

How can you assure no side effects either in an imperative (java, c) or declarative language (scala, scheme)? – orlybg Oct 10 '13 at 18:05

1 @orlybg, in declarative languages, the consistency comes from the implementation of the language. Their scope restrictions prevent them from having side effects. On the other hand, imperative languages exploit their side effects explicitly. Side effects are not always bad. – Tharindu Rusira Oct 28 '13 at 4:37

I'm reading the following Ada tutorial (goanna.cs.rmit.edu.au/~dale/ada/aln/8_subprograms.html), where the second paragraph of that page starts with "Procedures in Ada are similar to those in Pascal. A procedure can contain return statements.". Is this an error in the text? Or does it mean that it can have return statements but don't return any values? – jviotti Jan 14 '15 at 23:24

In pascal, procedures do not have return statements, only functions do. Must be an error in the text. However, a procedure can have an "exit" statement, which could act as a "return" statement without arguments, meaning no return values. – Eric Fortier Mar 2 '16 at 20:11

This depends on the context.

In Pascal-like languages, functions and procedures are distinct entities, differing in whether they do or don't return a value. They behave differently wrt. the language syntax (eg.

procedure calls form statements; you cannot use a procedure call inside an expression vs. function calls don't form statements, you must use them in other statements). Therefore, Pascal-bred programmers differentiate between those.

In C-like languages, and many other contemporary languages, this distinction is gone; in statically typed languages, procedures are just functions with a funny return type. This is probably why they are used interchangeably.

In functional languages, there is typically no such thing as a procedure - everything is a function.

edited Feb 12 '12 at 3:39

mikera
79.9k 12 195 352

answered Apr 6 '09 at 12:00

jpalecek
37.2k 5 70 116

and documentation of programming languages can call functions and procedures whatever it likes, because people will accept any name since the background behind those names has been washed out long ago. – Arne Babenhauserheide May 11 '15 at 9:25

Example in C:

```
// function
int square( int n ) {
   return n * n;
}

// procedure
void display( int n ) {
   printf( "The value is %d", n );
}
```

Although you should note that the C Standard doesn't talk about procedures, only functions.

answered Apr 6 '09 at 11:57
anon

1 *...the C Standard doesn't talk about procedures, only functions.* That's because it only has functions. A function that returns nothing is a `void function` . Kernighan & Ritchie Ch 1.7: "In C, a function is equivalent to a subroutine or function in Fortran, or a procedure or function in Pascal." In other words... this answer is wrong. – Mogsdad Aug 12 '15 at 15:41

5 The answer is not wrong, and it is a good example of the difference between pure functions and procedures. K&R called every subroutine a "function" to keep things simple, but a subroutine with side effects is in fact a "procedure", not a "function" in the canonical sense from mathematics. C might be a better language if it distinguished real functions from procedures, this would help with static analysis, performance optimization, and parallelization. – Sam Watkins Sep 18 '15 at 2:59

In general, a procedure is a sequence of instructions.
A function can be the same, but it usually returns a result.

answered Apr 6 '09 at 11:51

HS.
2,139 2 14 26

There's a term *subroutine* or *subprogram* which stands for a parameterized piece of code that can be called from different places.

Functions and procedures are implementations of those. Usually functions return values and procedures don't return anything.

answered Apr 6 '09 at 11:53

sharptooth
112k 52 333 734

More strictly, a function f obeys the property that f(x) = f(y) if x = y, i.e. it computes the **same result** each time it is called with the same argument (and thus it does not change the state of the system.)

Thus, rand() or print("Hello"), etc. are not functions but procedures. While sqrt(2.0) should be a function: there is no observable effect or state change no matter how often one calls it and it returns always 1.41 and some.

answered Apr 6 '09 at 12:01

Ingo
29.1k 4 34 82

2 This usage is relavent in the context of "functional" programming. Be aware that many (often imperative) languages which call their subprograms "functions" do not require this property. – dmckee Apr 6 '09 at 14:29

1 I didn't suggest that programming languages require this property. Anyway, one can write strict functions in any language, and I feel it is good habit to program as much as possible in clean functions, then glue the pieces together with some main procedure. – Ingo Apr 6 '09 at 14:43

Basic Difference Function must return a value but in Stored Procedure it is optional(Procedure can return zero or n values). Functions can have only input parameters for it whereas Procedures can have input/output parameters . Function takes one input parameter it is mandatory but Stored Procedure may take o to n input parameters.. Functions can be called from Procedure whereas Procedures cannot be called from Function.

Advance Difference Procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it. Procedures can not be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be. Functions that return tables can be treated as another rowset. This can be used in JOINs with other tables. Inline Function can be though of as views that take parameters and can be used in JOINs and other Rowset operations. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function. We can go for Transaction Management in Procedure whereas we can't go in Function.

answered Dec 18 '13 at 9:27

Mudassar Shahbaz
39 2

1 This answer is very language-specific, while the question was language-agnostic. The statements here are not all true in the general case, but it would be helpful if you clarified the language or environment you're asserting them for. – Mogsdad Aug 12 '15 at 15:25

Inside procedure we can use DML (Insert /Update/Delete) statements, But Inside function we can not use DML statements.

Procedure can have both input\output parameters, But Function can have only input parameter.

We can use Try-Catch Block in Stored Procedure, But In Function We can not use Try-Catch block.

We can not use Stored Procedure in Select statement, But In Function We can use in Select statement.

Stored Procedure can return 0 or n values (max 1024), But Function can return only 1 value which is mandatory.

Stored Procedure can not be call from Function, But We can call function from Stored Procedure.

We can use transaction in Stored Procedure, But In function we can not use transaction.

We can not use Stored Procedure in Sql statement anywhere in the Where/Having/select section, But In function we can use.

We can not join Stored Procedure, But we can join function.

for more.. click here...http://dotnet-developers-cafe.blogspot.in/2013/08/difference-between-stored-procedure-and.html

answered Dec 31 '13 at 6:10

Mukesh Kumar
1,157 11 22

This answer is very language-specific, while the question was language-agnostic. The statements here are not all true in the general case, but it would be helpful if you clarified the language or environment you're asserting them for. – Mogsdad Aug 12 '15 at 15:25

This answer is completely incorrect for the vast majority of programming languages. Procedures only have input parameters, and functions have both input and output. – cybermonkey Jan 28 '16 at 10:17

In most contexts: a function returns a value, while a procedure doesn't. Both are pieces of code grouped together to do the same thing.

In functional programming context (where all functions return values), a function is an abstract object:

```
f(x)=(1+x)
g(x)=.5*(2+x/2)
```

Here, f is the same function as g, but is a different procedure.

answered Apr 6 '09 at 11:56

xtofl
28.4k 6 66 141

If we're language-agnostic here, *procedure* usually specifies a series of acts required to reliably and idempotently achieve certain result. That is, a procedure is basically an algorithm.

Functions, on the other hand, is a somewhat independent piece of code within a larger program. In other words, function is the implementation of a procedure.

answered Apr 6 '09 at 11:56

Anton Gogolev
80.5k 28 154 251

A function returns a value and a procedure just executes commands.

The name function comes from math. It is used to calculate a value based on input.

A procedure is a set of command which can be executed in order.

In most programming languages, even functions can have a set of commands. Hence the difference is only in the returning a value part.

But if you like to keep a function clean, (just look at functional languages), you need to make sure a function does not have a side effect.

answered May 22 '12 at 4:17

OWOEYE GBENGA
21 1

Function can be used within a sql statement whereas procedure cannot be used within a sql statement.

Insert, Update and Create statements cannot be included in function but a procedure can have these statements.

Procedure supports transactions but functions do not support transactions.

Function has to return one and only one value (another can be returned by OUT variable) but procedure returns as many data sets and return values.

Execution plans of both functions and procedures are cached, so the performance is same in both the cases.

answered Jul 18 '12 at 10:44

pulak
19 1

2 The question isn't tagged with sql... – Shmiddty Jan 29 '13 at 22:30

I object with something I keep seeing over and over in most of these answers, that what makes a function a function is that it returns a value.

A function is not just any old method that returns a value. Not so: In order for a method to be a real function it must return the same value always given a specific input. An example of a method that is not a function is the `random` method in most languages, because although it does return a value the value is not always the same.

A function therefore is more akin to a map (e.g. where `x -> x'` for a one dimensional function). This is a very important distinction between regular methods and functions because when dealing with real functions the timing and the order in which they are evaluated should never matter where as this is not always the case with non functions.

Here's another example of a method that is not a function but will otherwise still return a value.

```
// The following is pseudo code:
g(x) = {
  if (morning()) {
     g = 2 * x;
  }
  else {
   g = x;
  }
  return g;
}
```

I further object to the notion that procedures do not return values. A procedure is just a specific way of talking about a function or method. So that means if the underlying method that your procedure defines or implements returns a value then, guess what that procedure returns a value. Take for example the following snippet from the SICP:

```
// We can immediately translate this definition into a recursive procedure
// for computing Fibonacci numbers:

(define (fib n)
  (cond ((= n 0) 0)
        ((= n 1) 1)
        (else (+ (fib (- n 1))
                 (fib (- n 2))))))
```

Have you heard of recursive procedures much lately? They are talking about a recursive function (a real function) and it's returning a value and they are using the word "procedure". So what's the difference, then?

Well another way of thinking of a function (besides the meaning mentioned above) is as an abstract representation of an ideal like the numeral 1. A procedure is that actual implementation of that thing. I personally think they are interchangeable.

(Note, if you read that chapter from the link I provide you may find that a harder concept to grasp is not the difference between a function and a procedure, but a process and a procedure. Did you know that a recursive procedure can have an iterative process?)

An analog for procedures are recipes. For example; suppose you have a machine called `make-pies` this machine takes in ingredients of `(fruit, milk, flower, eggs, sugar, heat)` and this machine returns a `pie` .

A representation of this machine might look like

```
make-pies (fruit, milk, flower, eggs, sugar, heat) = {
   return (heat (add fruit (mix eggs flower milk)))
}
```

Of course that's not the only way to make a pie.

In this case we can see that:

```
A       function     is to a     machine
as a    procedure    is to a     recipe
as      attributes   are to      ingredients
as      output       is to       product
```

That analogy is OK but it breaks down when you take into account that when you are dealing with a computer program everything is an abstraction. So unlike in the case of a recipe to a machine we are comparing two things that are themselves abstractions; two things that might as well be the same thing. And I hold that they are (for all intent and purposes) the same thing.

edited Feb 24 '14 at 15:24

answered Feb 24 '14 at 1:35

dkinzer

13.6k 7 42 69

1 A function that always returns the same value for given arguments is sometimes called a "pure function". In most languages that distinguish between procedures and functions, functions are not required to be pure, and the term "function" is correctly used to refer to subroutines that can have side effects and that can return different results on successive calls with the same arguments. (And in C-like languages, even subroutines that don't return values are properly called "functions".) – Keith Thompson May 15 '14 at 15:59

Agreed, which is why I end bay saying the the words are interchangeable. – dkinzer May 15 '14 at 19:56

Yes, but you *begin* by saying that "A function is not just any old method that returns a value", whereas in many languages that's *exactly* what a function is. – Keith Thompson May 15 '14 at 20:37

In the context of **db**: Stored procedure is **precompiled** execution plan where as functions are not.

answered Nov 26 '16 at 21:36

Awais
1,057 10 18

Procedures: 1.Procedures are the collections of statements that defines parameterized computations. 2.Procedures cannot return values.

3.Procedures cannot be called from function.

Functions 1.Functions structurally resemble procedures but are semantically modeled on mathematical functions. 2.It can return values 3.Function can be called from procedures.

answered Dec 21 '14 at 9:48

Safi ur Rehman
11

3.Procedures cannot be called from function. In what language is this true? None that I have experience in have this restriction. – Mogsdad Aug 12 '15 at 15:17

It is true. If you call a procedure from a function, then it is not a function. As to what language enforces this, that is a good question, to which I do not know the answer. May be a functional one, but even then I am not sure: pure list is functional (there is no set: no side affects), but as it has lambdas it is possible to implement set. Could you write a compiler that enforces no use of set, it would have to detect all implementations of it. You could remove lambdas from the language, but that would be worse. – richard Mar 13 '16 at 17:37

Ohh I just thought of a language C++: a const method can not call a not const method (though you will need the correct compiler checks turned on, and not try to hard to get around it.) – richard Mar 13 '16 at 17:38

Procedures and functions are both subroutines the **only** difference between them is that a procedure returns **multiple** (or at least can do) values whereas a function can only return **one** value (this is why function notation is used in maths as usually only one value is found at one given time) although some programming languages do not follow these rules this is their true definitions

answered Mar 30 '15 at 18:50

user2766296
8 4

Um, no. A procedure does not `return` anything. You're talking about side-effects, which are possible with both (if allowed by the language). – Mogsdad Aug 12 '15 at 15:15

A procedure can return any amount of values, that amount may be zero – user2766296 Feb 14 '16 at 13:28

A side-effect would be if a had an array and passed it into a function or procedure that found the biggest value, the array would be passed by reference and after the sub-routine has run the array is sorted, the fact that it is sorted is a side-effect, the value returned is the biggest value in the array – user2766296 Feb 14 '16 at 13:44