Pykidz!

A 'Kideveloper' initiative

Functions, Modules & Packages: Modularity in Python

I do I learn!

Functions in Python

The Importance of Python Functions

- Abstraction & Reusability
- Modularity
- Namespace Separation

Modularity

```
# Main program

# code to collect input from the user
<statement>
<statement>
<statement>
<statement>

# do the calculations
<statement>
<statement>
<statement>
<statement>
<statement>

# print the result
<statement>
<statement>
<statement>
<statement>
```

Modularity

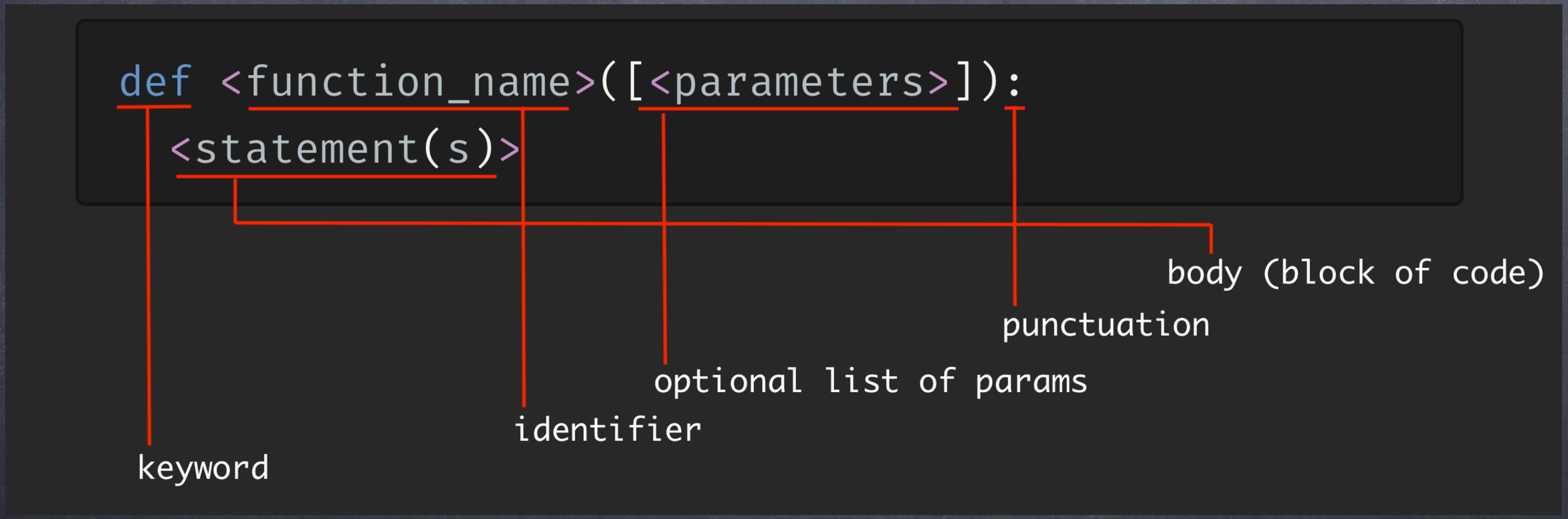
```
def take_user_input():
  <statement>
  <statement>
  <statement>
  <statement>

def do_calculations():
  <statement>
  <statement>
  <statement>
  <statement>
  <statement>

def print_results():
  <statement>
  <statement>
  <statement>
  <statement>

# Main program
take_user_input()
do_calculations()
print_results()
```

```
# Main program

# code to collect input from the user
<statement>
<statement>
<statement>
<statement>

# do the calculations
<statement>
<statement>
<statement>
<statement>
<statement>

# print the result
<statement>
<statement>
<statement>
<statement>
```

Function Calls & Definition

function definition

```
def <function_name>([<parameters>]):
    <statement(s)>
```



Argument Passing

- Positional Arguments
- Keyword Arguments
- Default Parameters
- Pass-by-value vs Pass-by-reference in python
- Side Effects

The return Statement

- Exiting a Function
- Returning Data to the Caller
- Revisiting Side Effects

Modules & Packages - An intro

Modular Programming

- Simplifying code
- Maintainability
- Reuse
- Scoping

3 ways

- Module written in python
- Written in 'C'
- A built-in module
    - import

Modules

- The Module Search Path
- The `import` Statement
- Executing a Module as a Script
- Reloading a Module

Python Packages

Conclusion

I do I learn!