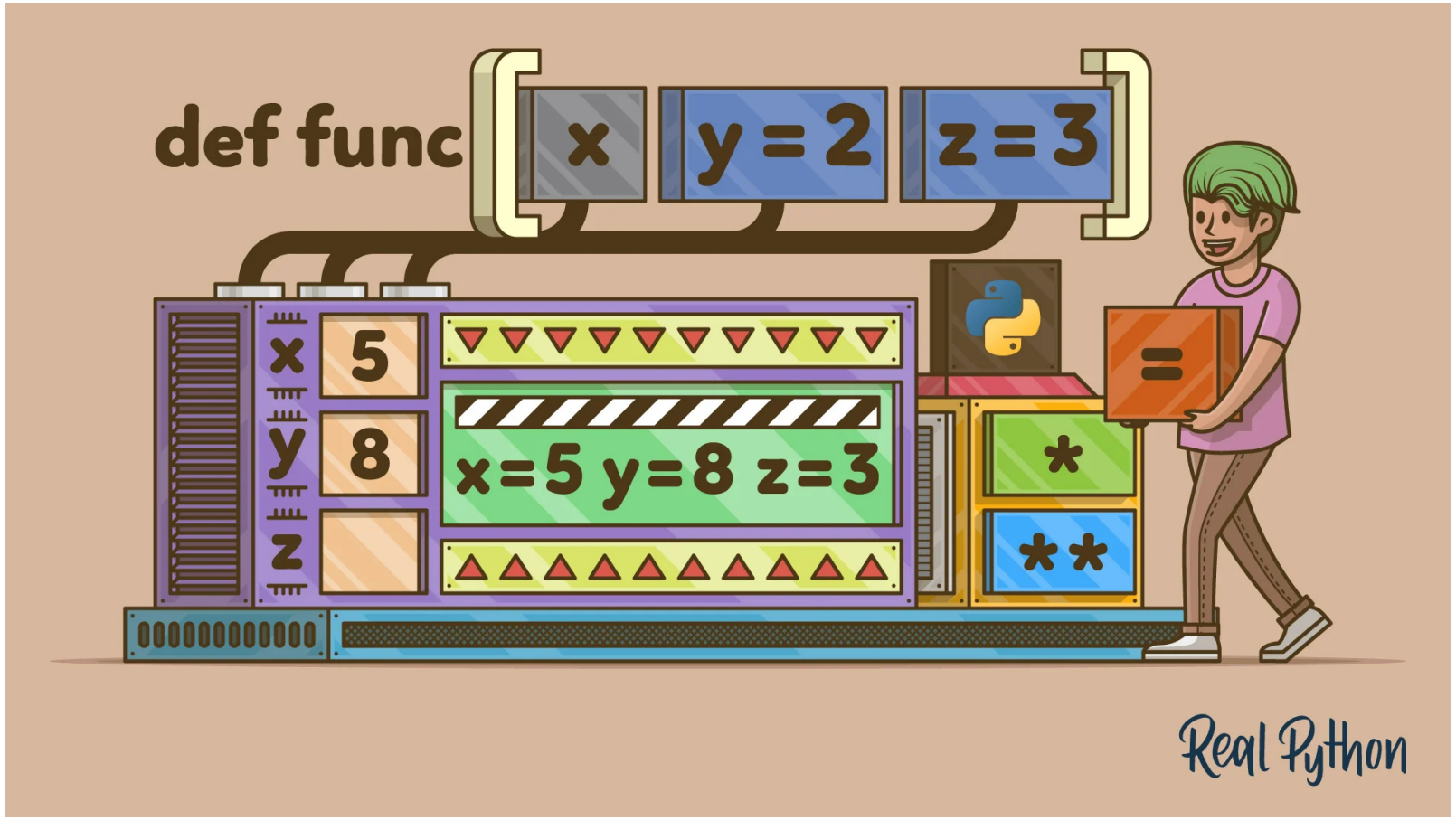
Using Python Optional Arguments When Defining Functions

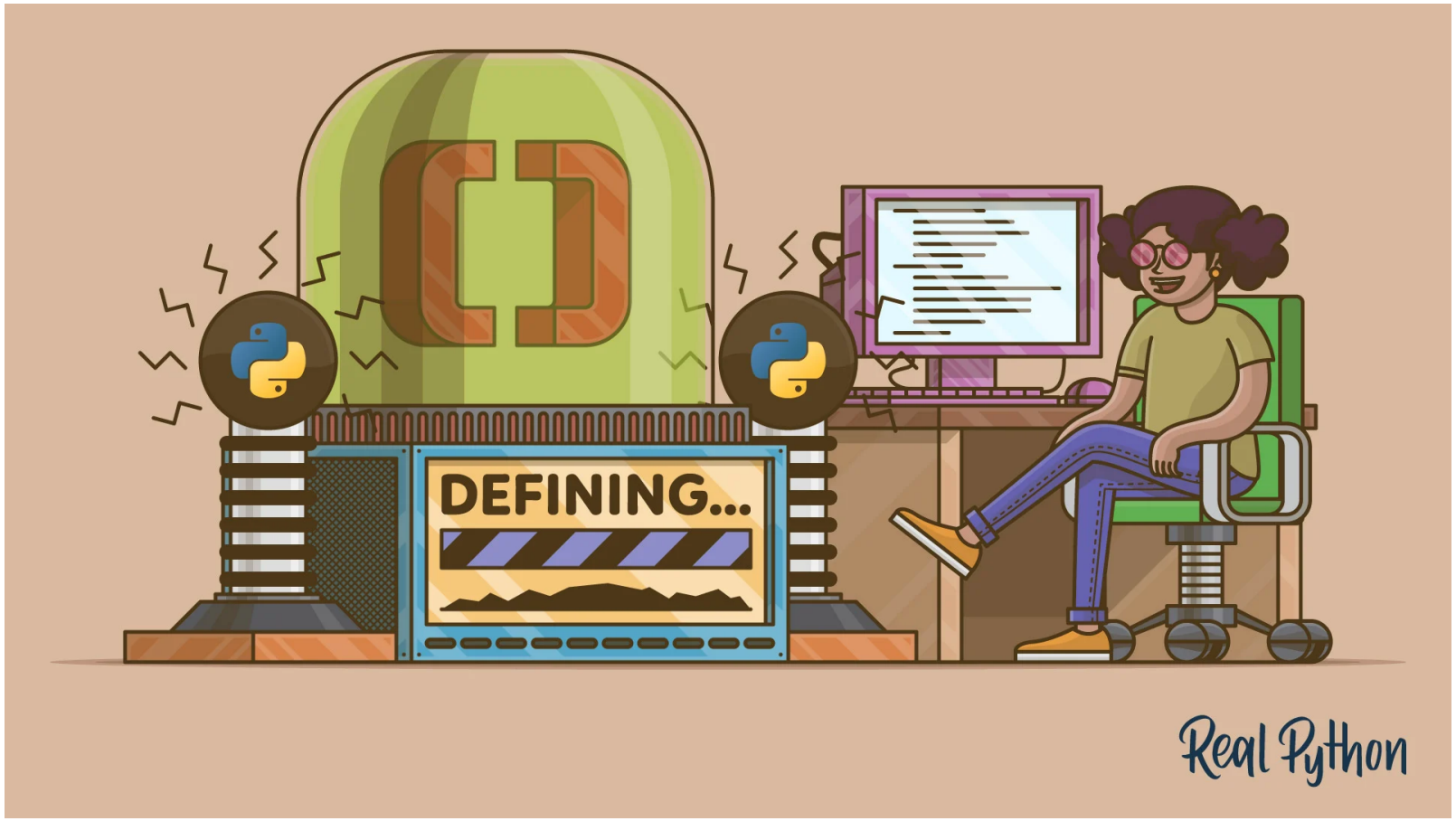


Using Python Optional Arguments When Defining Functions

Using Python Optional Arguments When Defining Functions

- The Difference Between Parameters and Arguments
- Functions With Optional Arguments and Default Values
- How To Define Functions Using Args and Kwargs
- Dealing With Error Messages

Defining Your Own Python Function



https://realpython.com/defining-your-own-python-function/

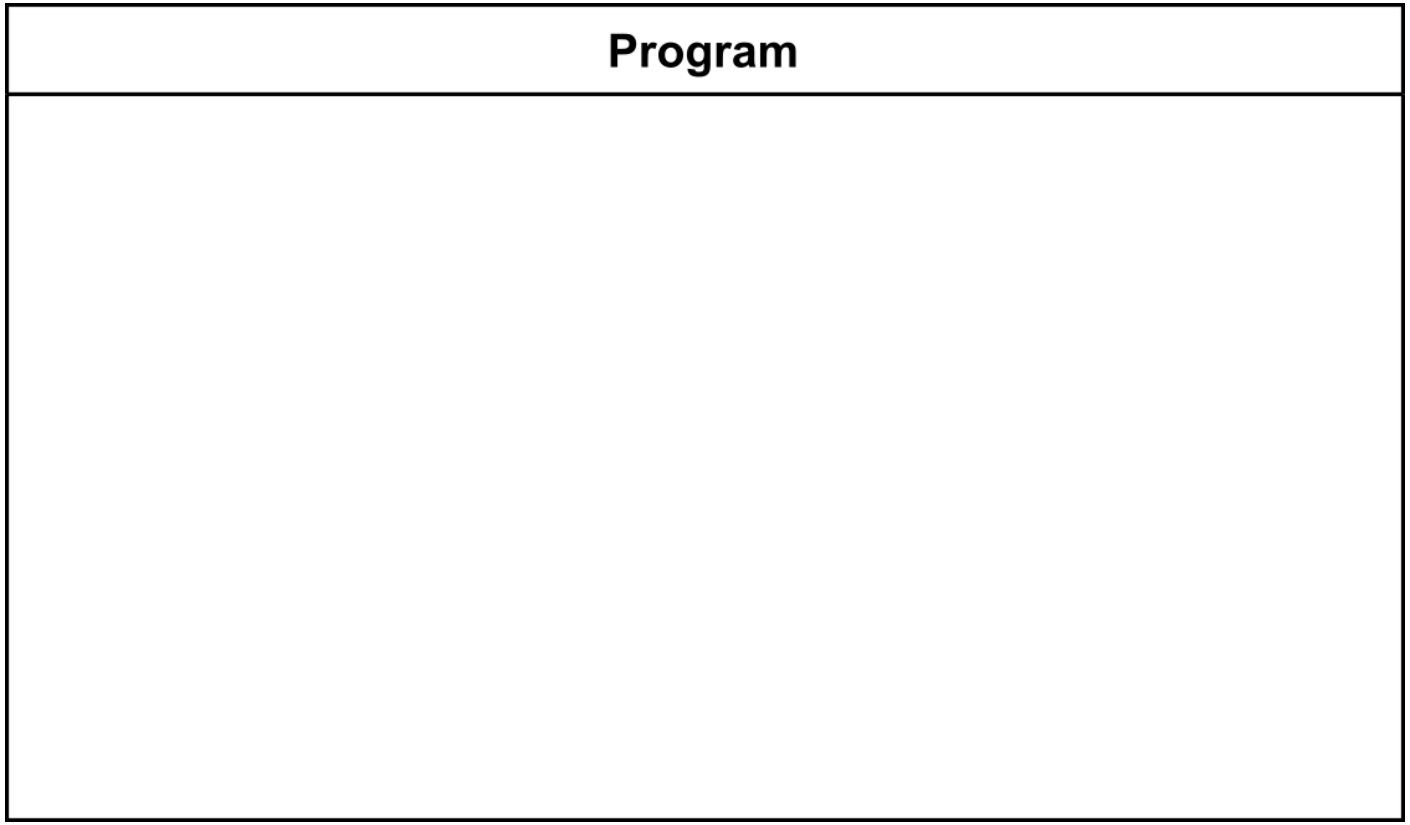
Let’s Get Started

Using Python Optional Arguments When Defining Functions

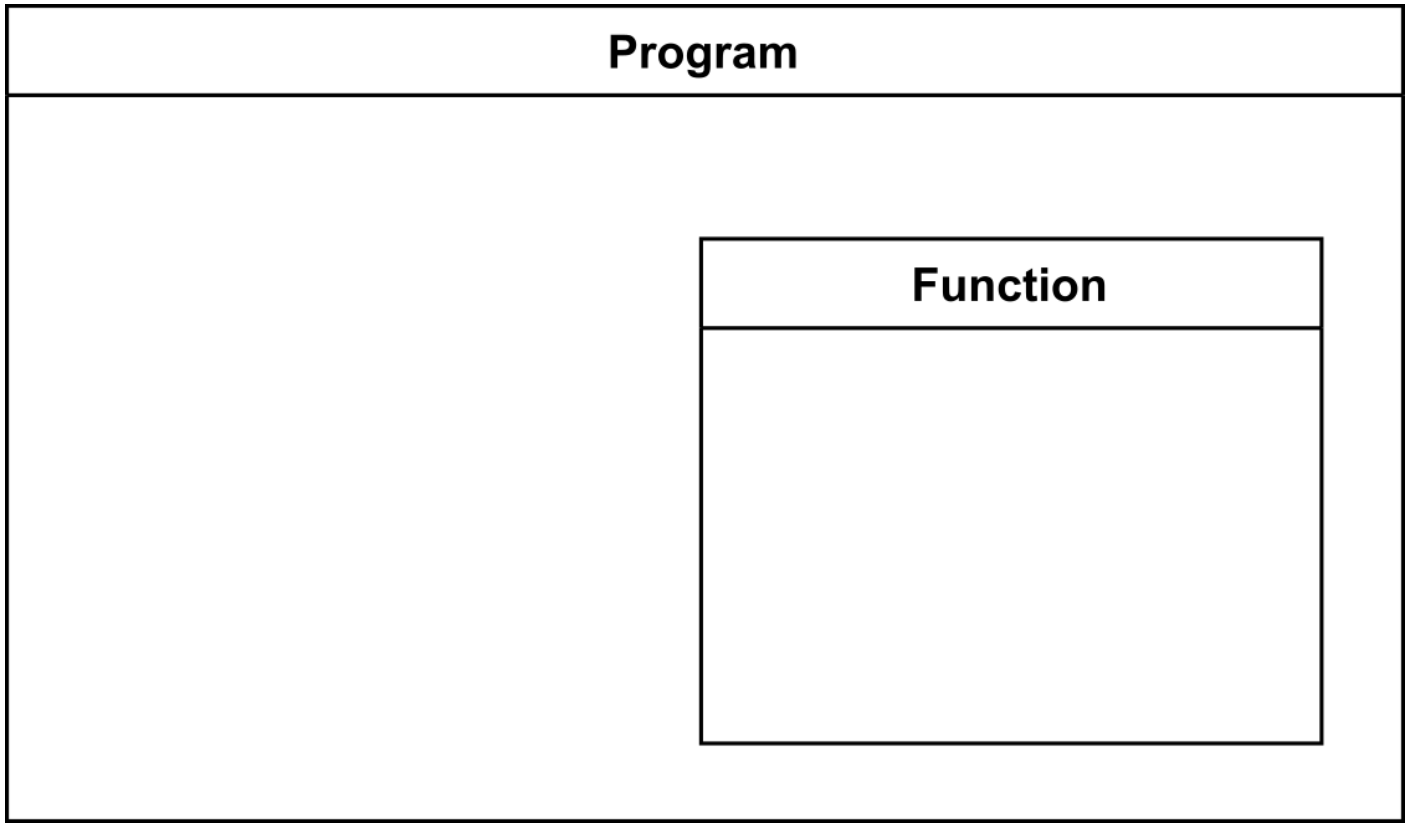
1. **Creating Functions in Python for Reusing Code**
2. Using Python Optional Arguments With Default Values
3. Using `args` and `kwargs`

Creating Functions in Python for Reusing Code

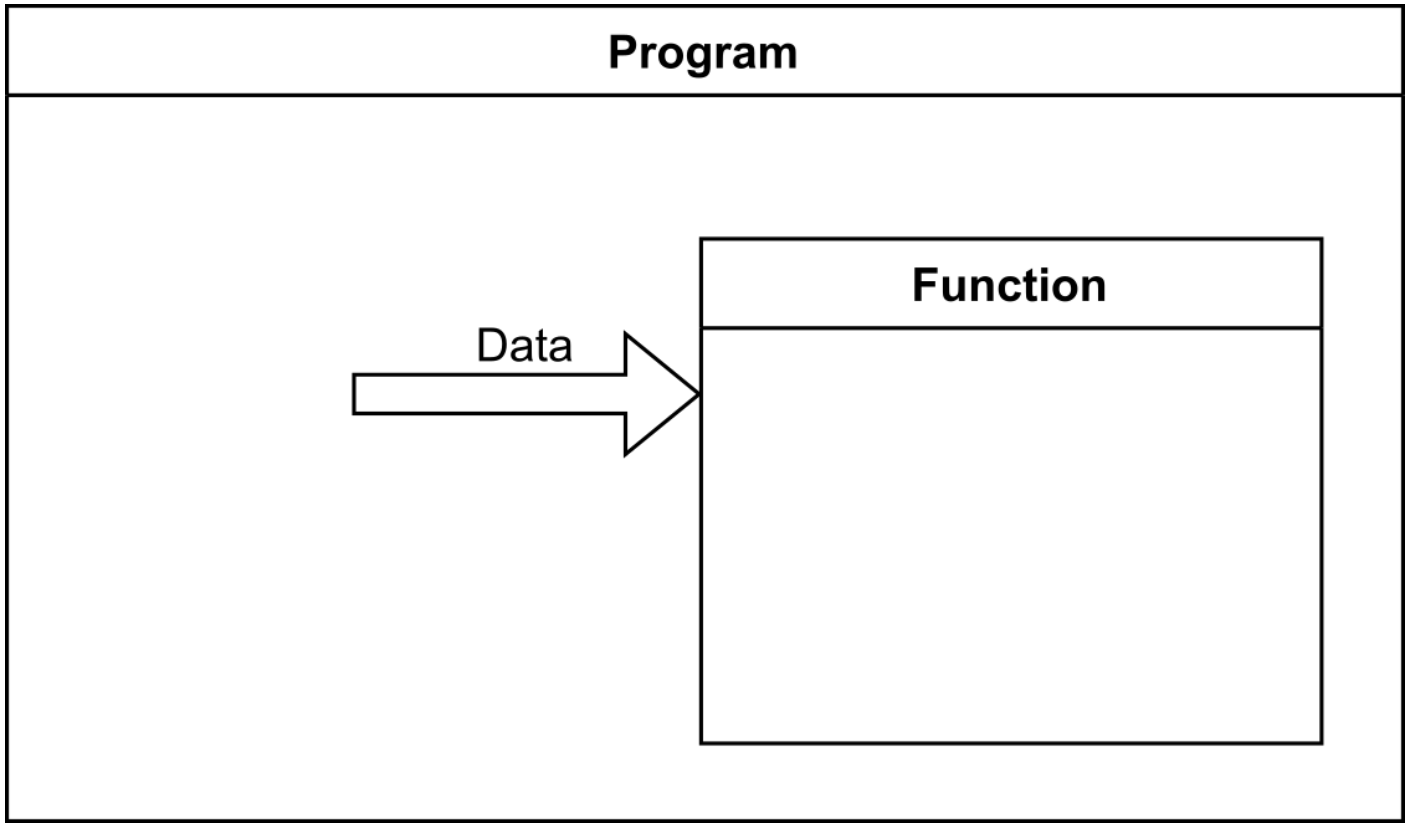
Functions in Action



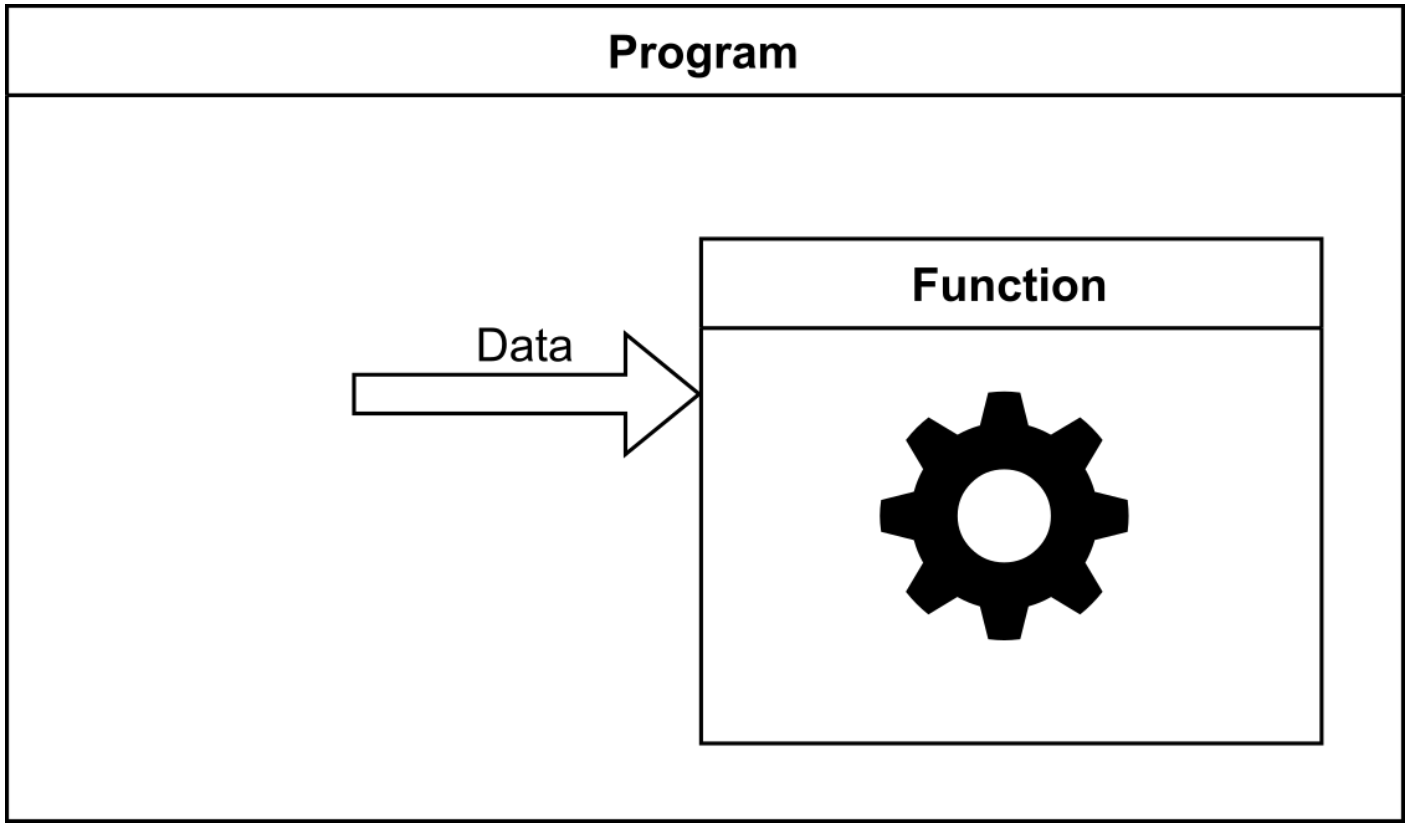
Functions in Action



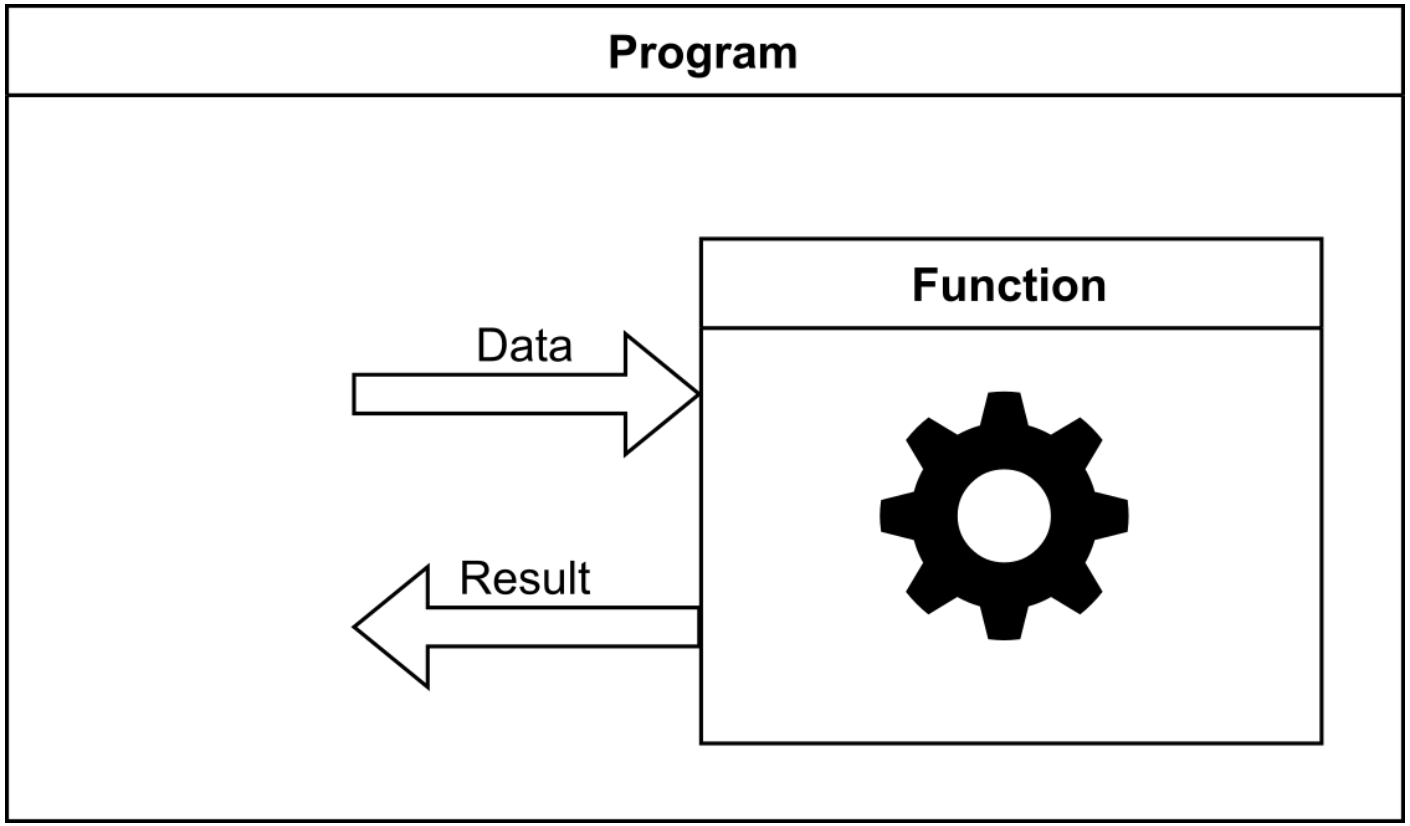
Functions in Action



Functions in Action



Functions in Action



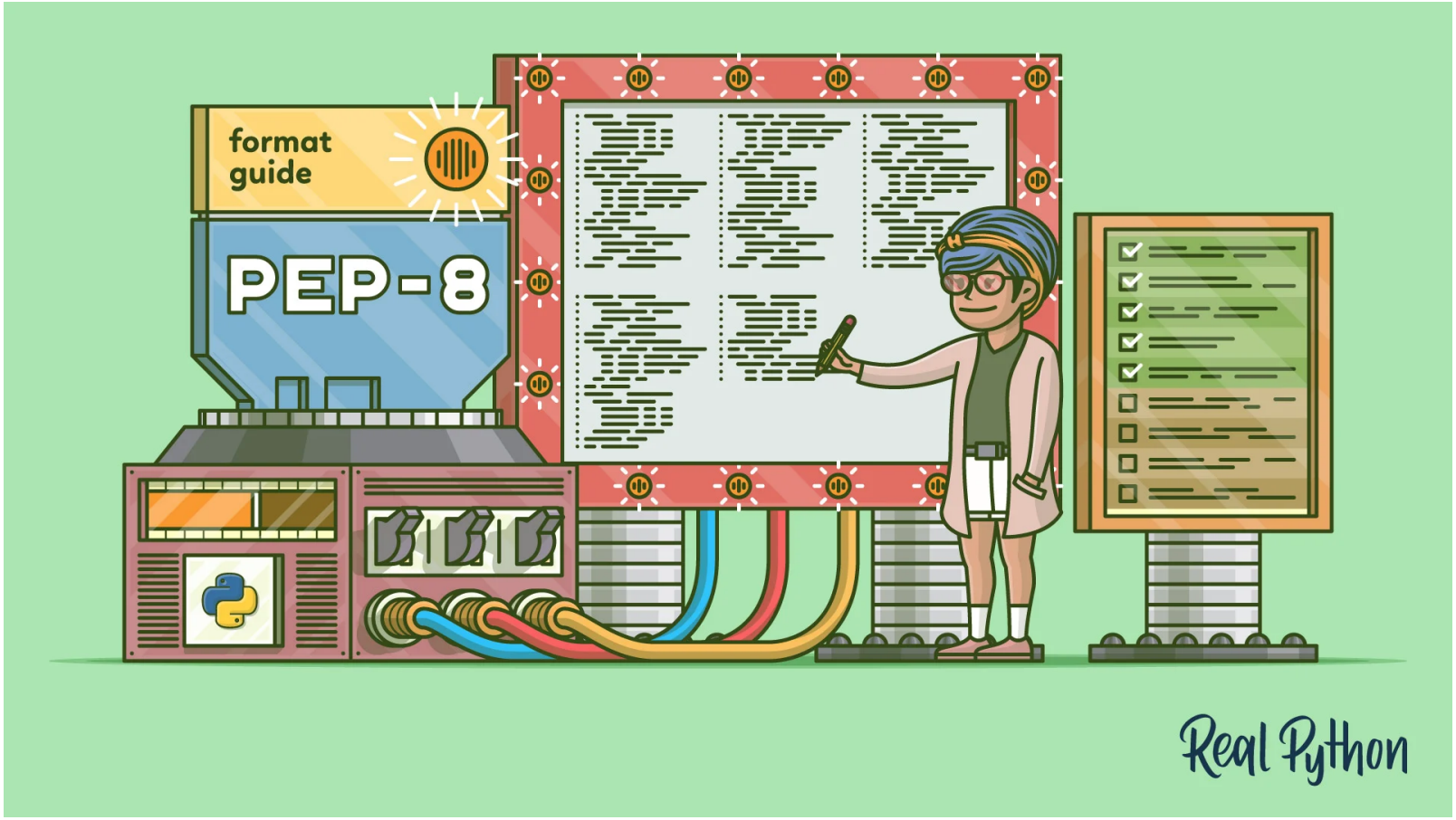
Functions

- Allow Code Re-use
- Extend Python Vocabulary
- Allow Succinct Solutions

Function Naming Convention

Name	PEP8
`do_something()`	
`dosomething()`	
`DoSomething()`	

How to Write Beautiful Python Code With PEP 8



https://realpython.com/python-pep8/

Calling A Function

Function Call	OK?
`do_something()`	
`do_something`	

Next: Defining Functions With No Input Parameters

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
   - 1.1 **Defining Functions With No Input Parameters**
   - 1.2 Defining Functions With Required Input Arguments
2. Using Python Optional Arguments With Default Values
3. Using `args` and `kwargs`

Defining Functions With No Input Parameters

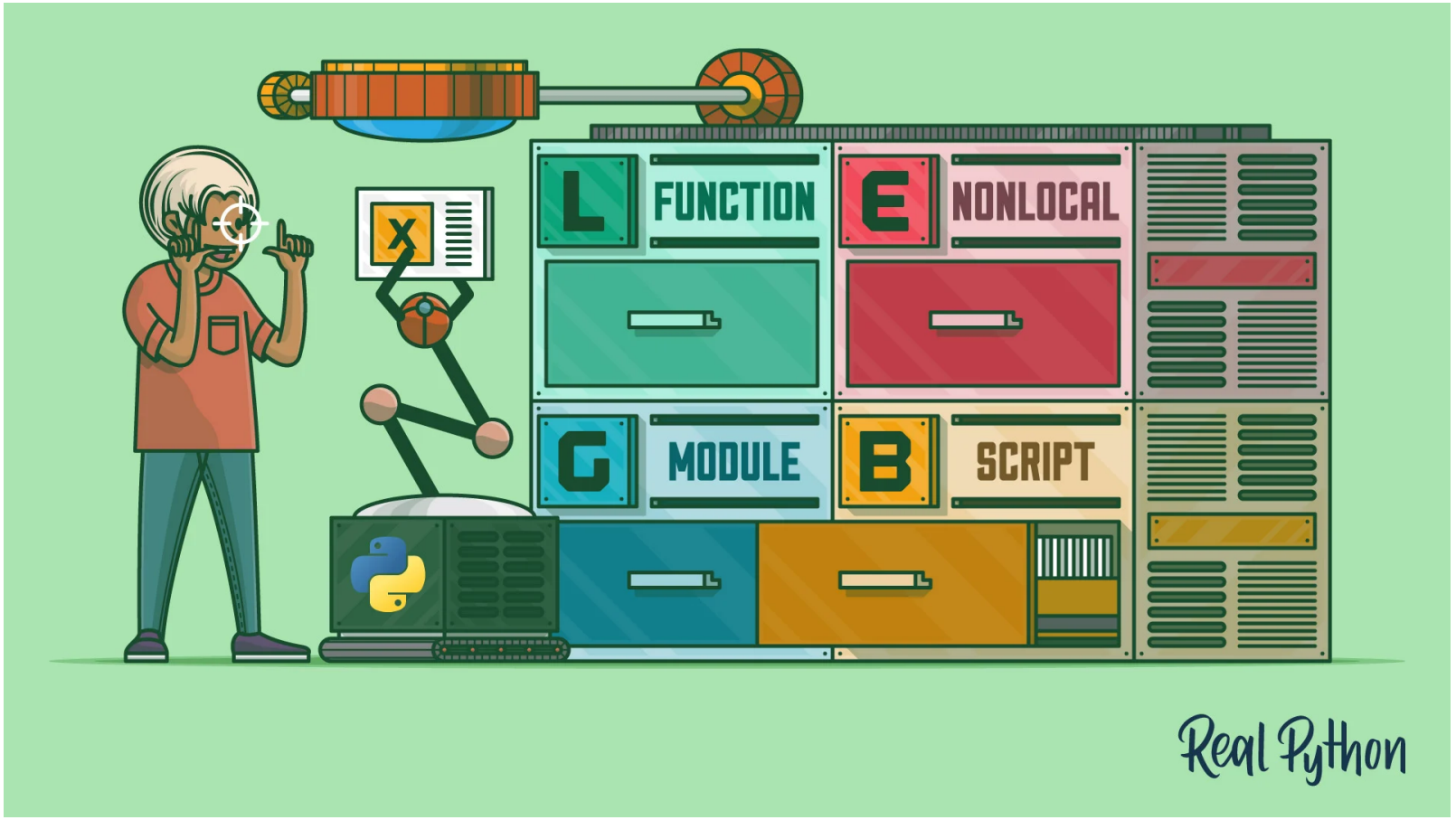
Function Signature

```
def show_list():
```

`shopping_list` Is a Global Variable

- Accessible From Everywhere in the Program
- This Is the Global Scope

Python Scope & the LEGB Rule: Resolving Names in Your Code



https://realpython.com/python-scope-legb-rule/

Global Variable Pitfalls

- Not Good Practice
- Multiple Functions Accessing Same Data Structure
- Can Lead to Hard-to-Find Bugs

Next: Defining Functions With Required Input Arguments

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
   1.1 Defining Functions With No Input Parameters
   1.2 **Defining Functions With Required Input Arguments**
2. Using Python Optional Arguments With Default Values
3. Using `args` and `kwargs`

Defining Functions With Required Input Arguments

Function Signature

```
def add_item(item_name, quantity):
```

Parameters Are Used Within the Function

```
def add_item(item_name, quantity):
    if item_name in shopping_list.keys():
        shopping_list[item_name] += quantity
    else:
        shopping_list[item_name] = quantity
```

Arguments Are Values Passed to the Function

```
add_item("Bread", 1)
add_item("Pies", 4)
```

Arguments and Parameters

- Parameters - Formal Parameters
- Arguments - Actual Parameters

Error Messages Will Be Covered Later On

Next: Using Python Optional Arguments With Default Values

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. **Using Python Optional Arguments With Default Values**
3. Using `args` and `kwargs`

Using Python Optional Arguments With Default Values

- More Flexible Functions
- Can Be Called With or Without Argument
- Use Default Value If No Argument Provided

Next: Default Values Assigned to Input Parameters

Using Python Optional Arguments When Defining Functions

Default Values Assigned to Input Parameters

Function Signature

```
def add_item(item_name, quantity=1):
```

Default Values In Action

Default Values In Action

Function Call	`quantity` Value	Default Used?
`add_item("Milk", 2)`	2	

Default Values In Action

Function Call	`quantity` Value	Default Used?
`add_item("Milk", 2)`	2	✗
`add_item("Bread")`	1	✓

Parameter Order Is Important

Required And Optional Arguments

Required And Optional Arguments

```
add_item("Bread")
```

Required And Optional Arguments

```
add_item("Bread")
```

`item_name` = `"Bread"`, `quantity` = `1` (optional, default value used)

Required And Optional Arguments

```
add_item("Bread")
```

`item_name` = `"Bread"`, `quantity` = `1` (optional, default value used)

```
add_item("Milk", 2)
```

Required And Optional Arguments

```
add_item("Bread")
```

`item_name` = `"Bread"`, `quantity` = `1` (optional, default value used)

```
add_item("Milk", 2)
```

`item_name` = `"Milk"`, `quantity` = `2`

Keyword Arguments

```
add_item(item_name = "Milk", quantity = 2)
```

Avoid Chameleon-Like Functions

- Functions Should Have A Single Responsibility
- Consider A Separate Function For Alternate Functionality

Next: Common Default Argument Values

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. Using Python Optional Arguments With Default Values
   - 2.1 Default Values Assigned to Input Parameters
   - 2.2 **Common Default Argument Values**
   - 2.3 Data Types That Shouldn't Be Used as Default Arguments
   - 2.4 Error Messages Related to Input Arguments
3. Using `args` and `kwargs`

Common Default Argument Values

- `1`
- `True`
- Dependent on Function

Common Integer Default Values

- `0` or `1`
- Useful Fallback Values
- Sensible Default for `add_item()`
- Other Options May Be Appropriate

`""` Equates To `False`

`""` Equates To `False`

Python Shell:

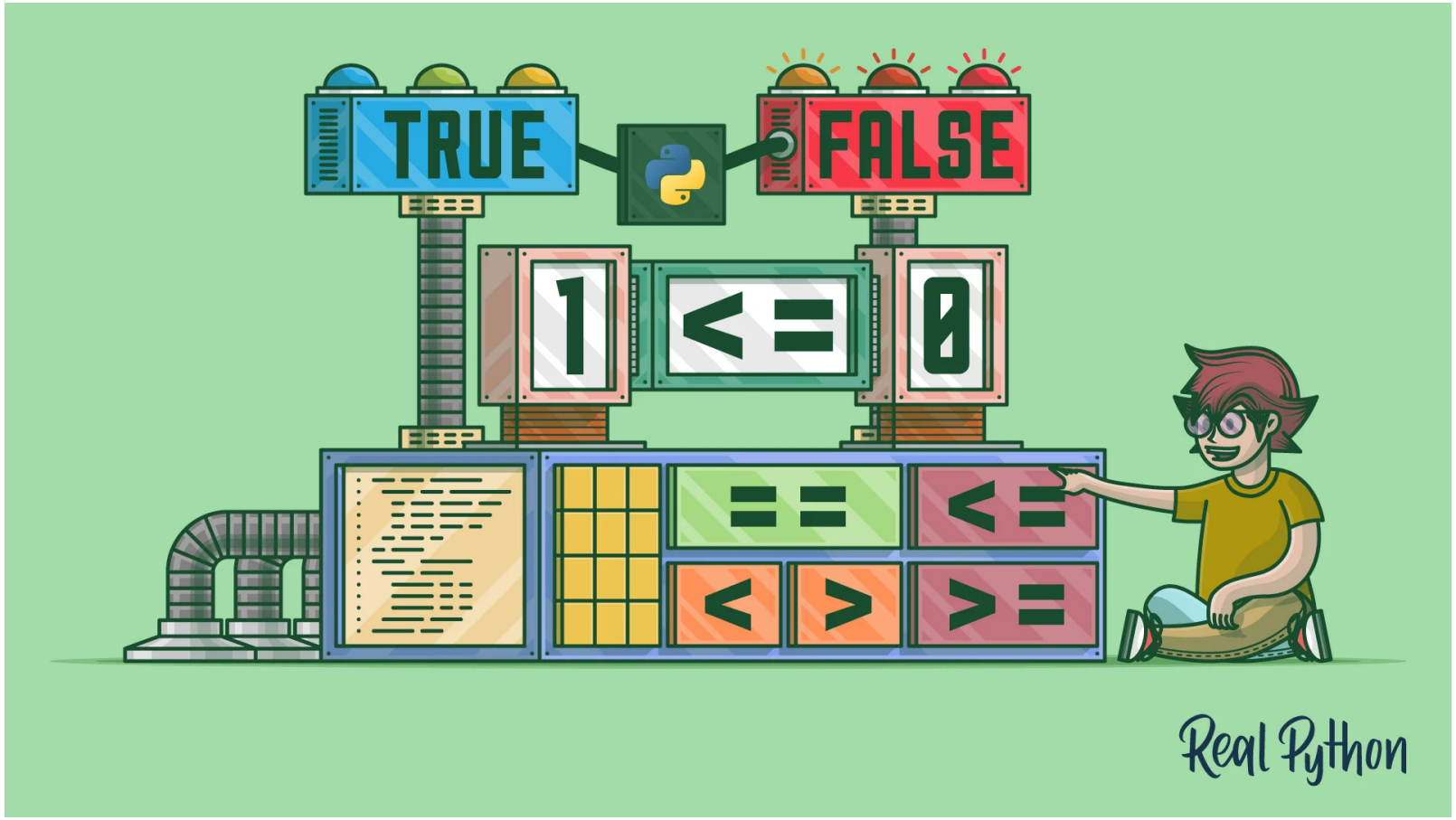
```
>>> x = ""
>>> bool(x)
False
```

`""` Equates To `False`

Python Shell:

```
>>> x = ""
>>> bool(x)
False
>>> y = "Bread"
>>> bool(y)
True
```

Python Booleans: Optimize Your Code With Truth Values



https://realpython.com/python-boolean/

Truthiness Allows Direct Use in `if` Statements

`None`

- Common Default Value
- Python Representation of Nothing
- Object That Represents `null` Value

Next: Data Types That Shouldn’t Be Used as Default Arguments

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. Using Python Optional Arguments With Default Values
   2.1 Default Values Assigned to Input Parameters
   2.2 Common Default Argument Values
   2.3 **Data Types That Shouldn't Be Used as Default Arguments**
   2.4 Error Messages Related to Input Arguments
3. Using `args` and `kwargs`

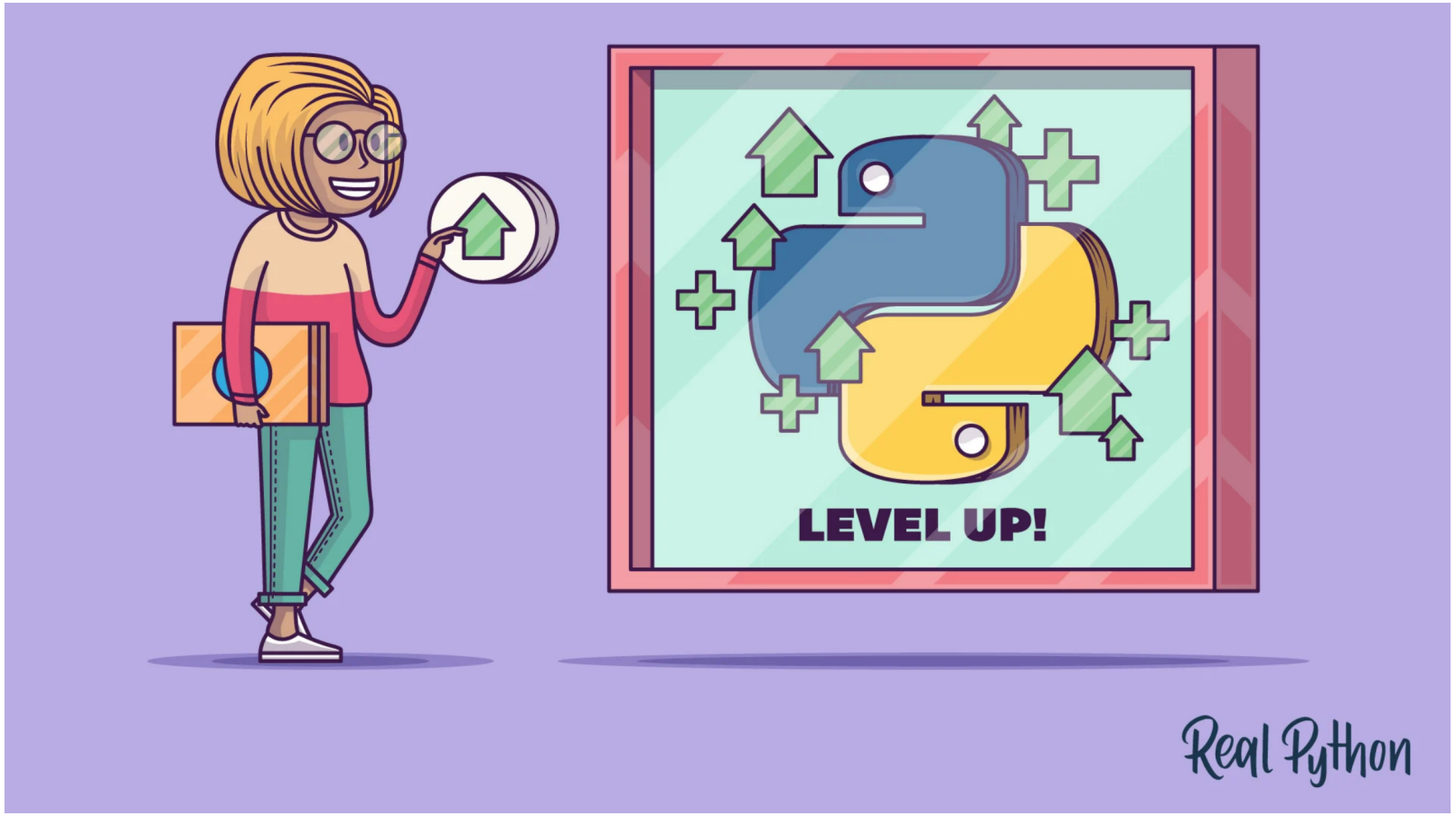
Default Values Used

- Integers - `int`
- Strings - `str`
- None - `None`

Mutable Data Types

- Values Can Be Changed
- List - `list()` - `['Bread', 'Milk']`
- Dictionary - `dict()` - `{'Bread': 2, 'Milk': 1}`

Immutability in Python



https://realpython.com/courses/immutability-python/

Passing The Shopping List To `add_item()`

Calling The Updated `add_item()`

Previous Usage:

```
add_item("Item", 2)
```

New Usage:

```
new_shopping_list = add_item("Item", 2)
```

Updating `show_list()`

Dictionaries Are Mutable

Dictionaries Are Mutable

```
def add_item(item_name, quantity, shopping_list={}):
```

Dictionaries Are Mutable

```
def add_item(item_name, quantity, shopping_list={}):
```

At First Call:

`shopping_list` = `{}`

Dictionaries Are Mutable

```
def add_item(item_name, quantity, shopping_list={}):
```

At First Call:

`shopping_list` = `{}`

At Second Call:

`shopping_list` = `{"Shirt": 3}`

Never Use Mutable Default Arguments

Next: Error Messages Related to Input Arguments

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. Using Python Optional Arguments With Default Values
   - 2.1 Default Values Assigned to Input Parameters
   - 2.2 Common Default Argument Values
   - 2.3 Data Types That Shouldn't Be Used as Default Arguments
   - 2.4 **Error Messages Related to Input Arguments**
3. Using `args` and `kwargs`

Error Messages Related to Input Arguments

Parameter Order

- **No** Default Values - Any Parameter Order
- **All** Default Values - Any Parameter Order
- Mixed Defaults and Non-Default - Default Values Last

Next: Using `args` and `kwargs`

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. Using Python Optional Arguments With Default Values
3. **Using `args` and `kwargs`**

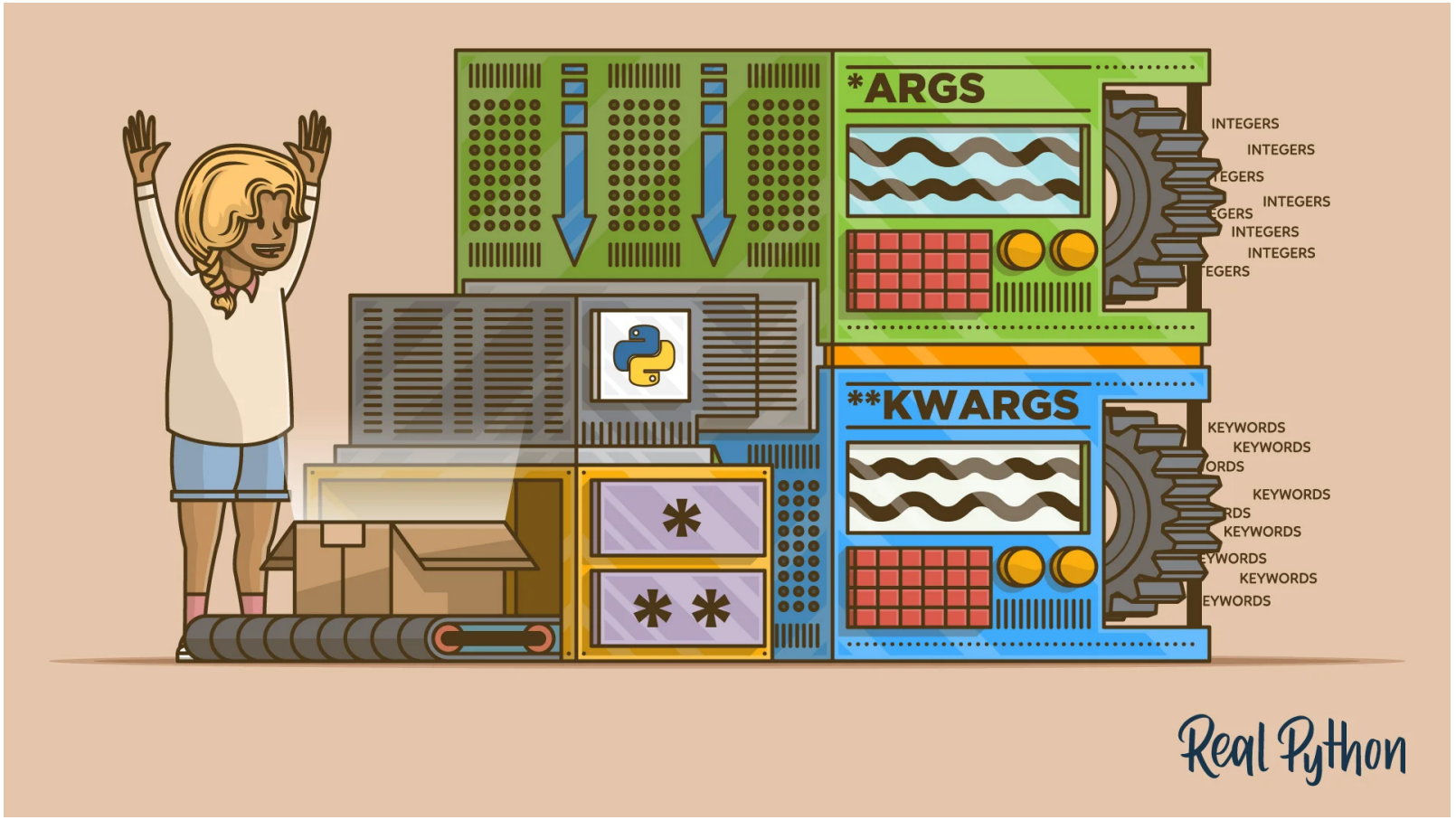
Function Arguments

- Optional Argument
- Multiple Optional Arguments
- Any Number Of Arguments
- Any Number Of Keyword Arguments

Accepting Any Number Of Arguments

- `args` - Arguments
- `kwargs` - Keyword Arguments

Python args and kwargs: Demystified



https://realpython.com/python-kwargs-and-args/

Next: Functions Accepting Any Number of Arguments

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. Using Python Optional Arguments With Default Values
3. Using `*args` and `**kwargs`
   - 3.1 **Functions Accepting Any Number of Arguments**
   - 3.2 Functions Accepting Any Number of Keyword Arguments

Functions Accepting Any Number of Arguments

Bpython Interpreter

https://bpython-interpreter.org/

Unpacking

```
some_items = ["Coffee", "Tea", "Cake", "Bread"]
```

```
some_items = ["Coffee", "Tea", "Cake", "Bread"]
```

```
*some_items = "Coffee", "Tea", "Cake", "Bread"
```

`print()` Can Accept Any Number Of Arguments

```
>>> print("One Argument")
One Argument
>>> print("Two", "Arguments")
Two Arguments
>>> print()

>>>
```

Redefining `add_items()`

- Accept a List of Items
- Set Quantity for Each to 1
- Quantities Will Be Set Later

Function Signature

```
def add_items(shopping_list, *args):
```

- `*` - Unpacks a Sequence

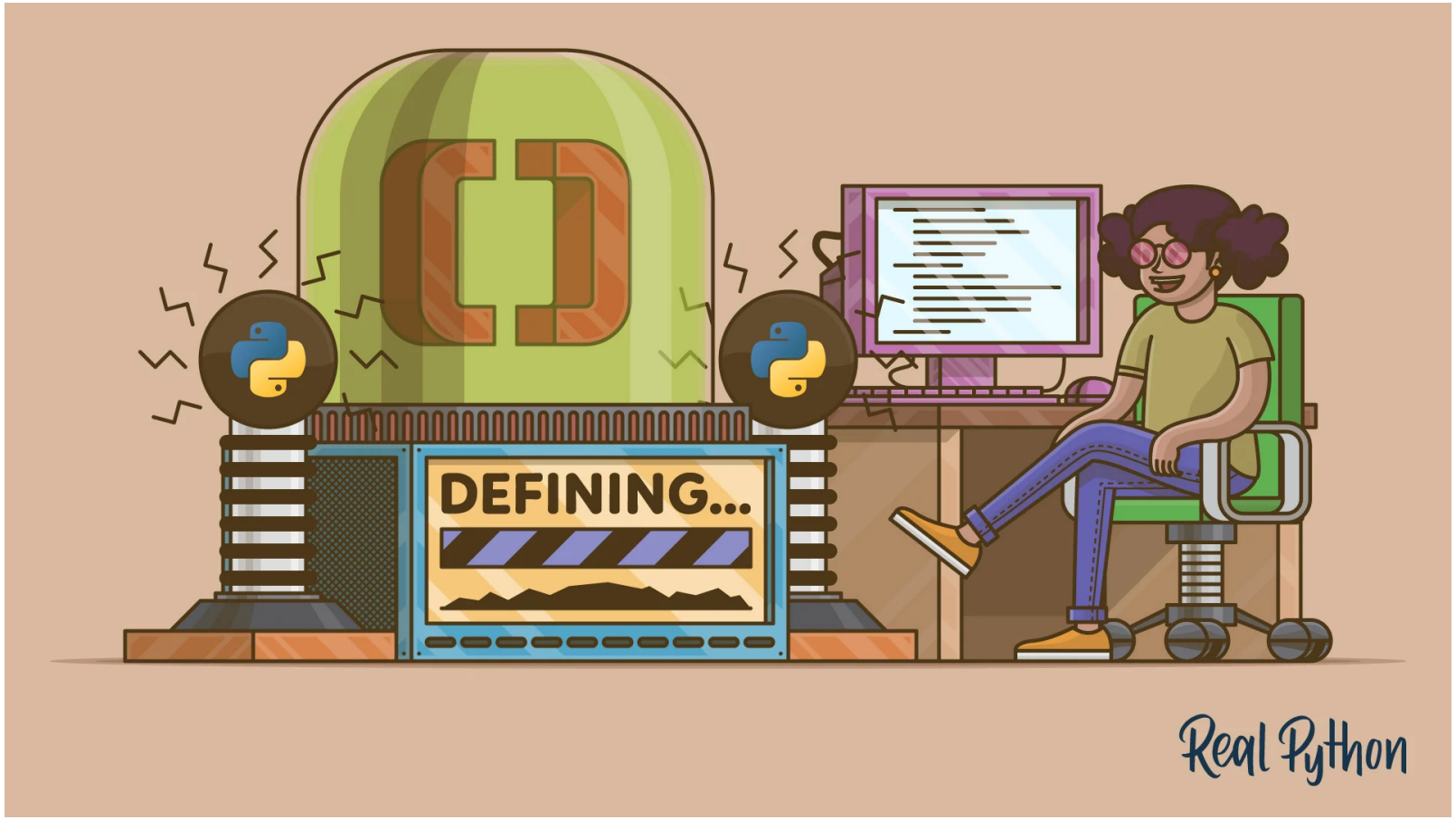
Next: Functions Accepting Any Number of Keyword Arguments

Using Python Optional Arguments When Defining Functions

1. Creating Functions in Python for Reusing Code
2. Using Python Optional Arguments With Default Values
3. Using `args` and `kwargs`
   3.1 Functions Accepting Any Number of Arguments
   3.2 **Functions Accepting Any Number of Keyword Arguments**

Functions Accepting Any Number of Keyword Arguments

Defining Your Own Python Function



https://realpython.com/defining-your-own-python-function

Function Signature

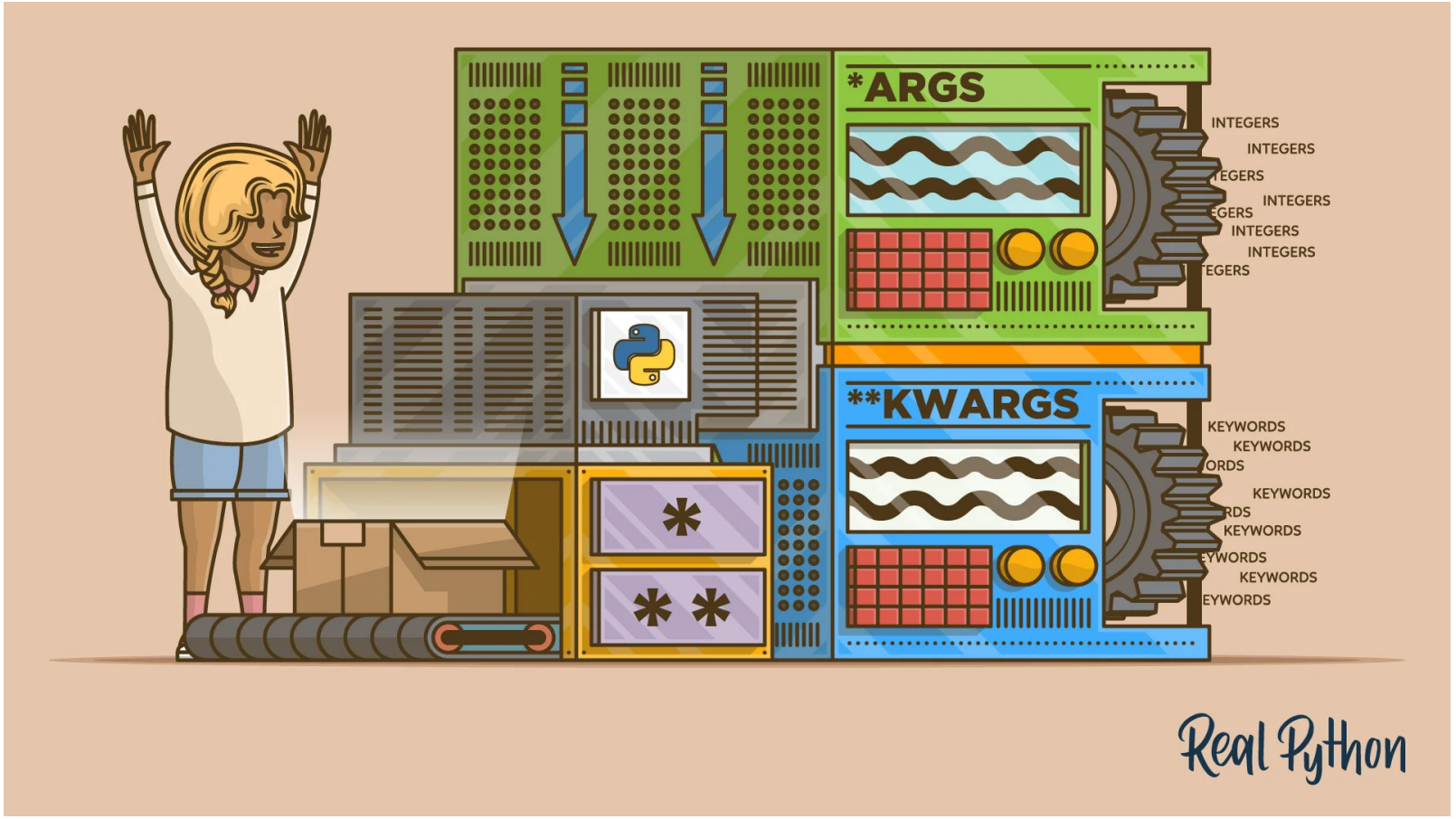
```
def add_items(shopping_list, **kwargs):
```

- `**` - Unpacks a Mapping

Argument Data Structures

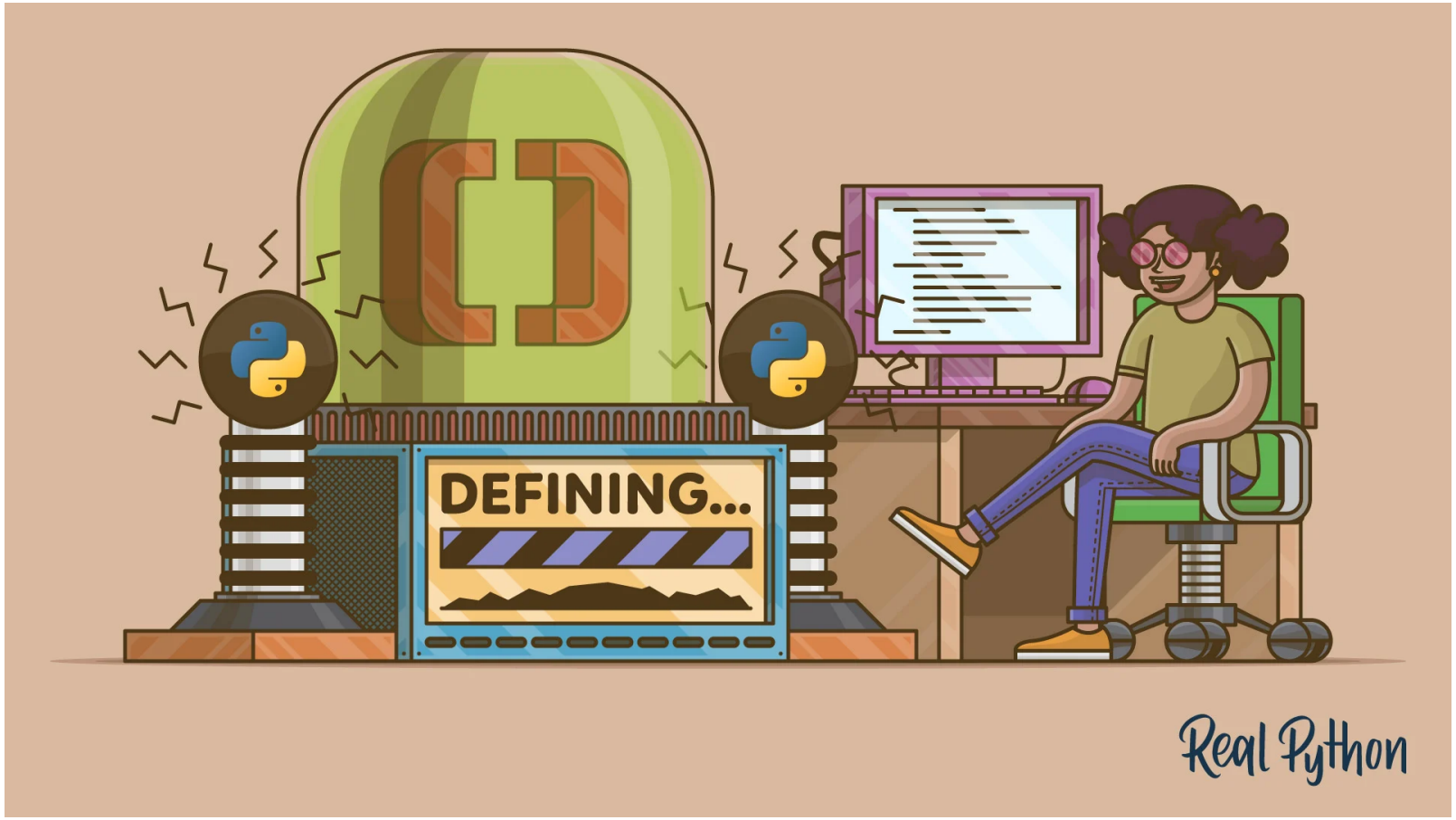
- `*args` - Tuple
- `**kwargs` - Dictionary

Python args and kwargs: Demystified



https://realpython.com/python-kwargs-and-args/

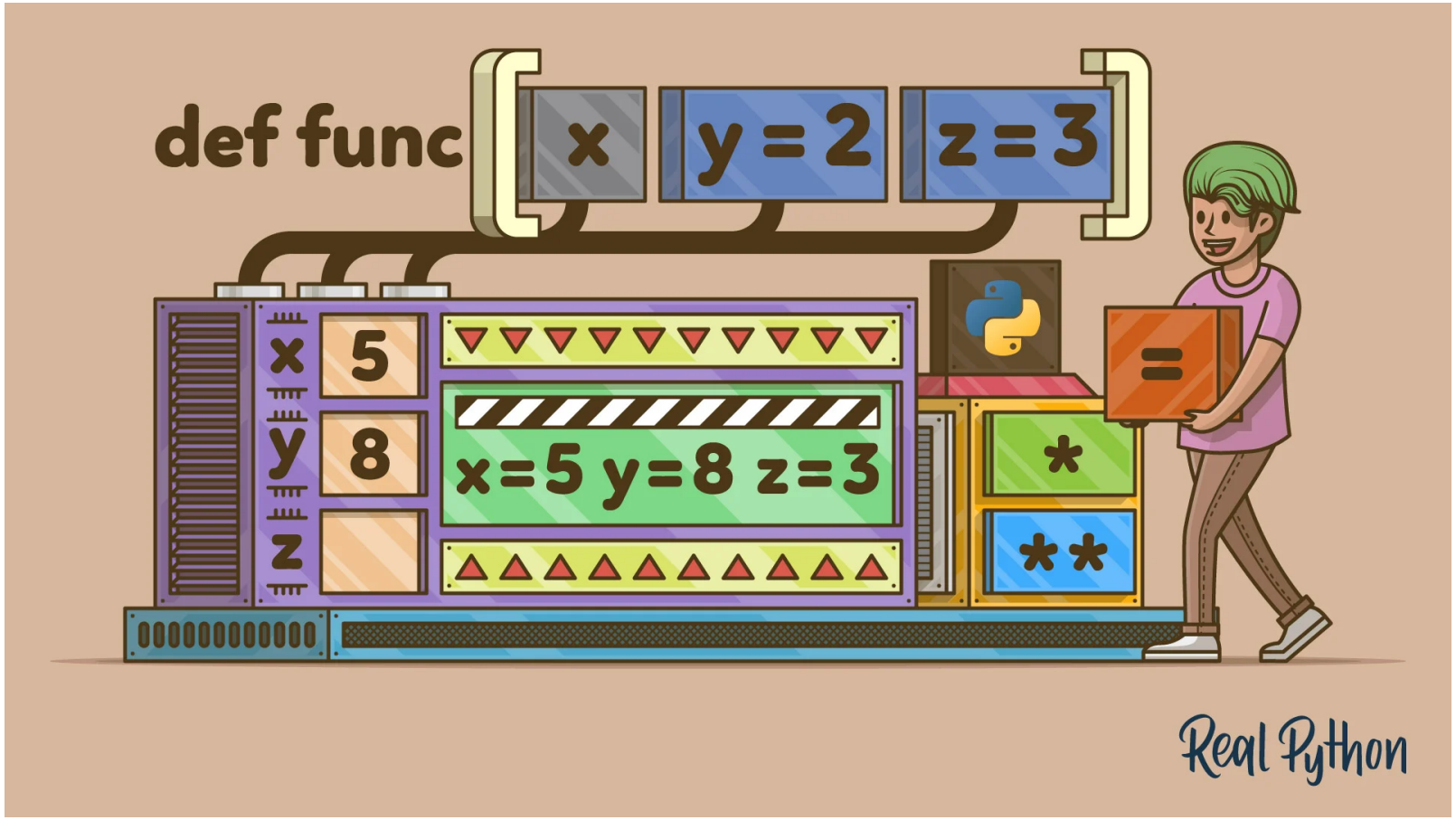
Defining Your Own Python Function



https://realpython.com/defining-your-own-python-function

Next: Summary

Using Python Optional Arguments When Defining Functions: Summary



Summary

- The Difference Between Parameters and Arguments
- Defining Functions With Optional Arguments
- Defining Functions With Default Parameter Values
- Defining Functions With `args` and `kwargs`
- Error Messages for Optional Arguments

Summary