

ELLYPUNK

Foundation

WEEKLY DIGEST / E101

21-27 September 2026



This week

21-27 September 2026

Available archived editions, in their original date order.

MONDAY / 21 September 2026

What Is a Primary Key? Give Every Row an Identity

3

[Open original online](#)

TUESDAY / 22 September 2026

What Is an Idempotency Key? Make Retries Safe

8

[Open original online](#)

WEDNESDAY / 23 September 2026

What Is a Webhook? An Event Calls Your App

13

[Open original online](#)

THURSDAY / 24 September 2026

What Is an Environment Variable? A Name for a Setting

18

[Open original online](#)

FRIDAY / 25 September 2026

What Is a Function? Give a Task a Name

23

[Open original online](#)

Only retained editions are included; gaps have not been filled.

Copyright Ellypunk. Contains AI-generated artwork.

ellypunk.com

ELLY EXPLAINS: WHAT IS A PRIMARY KEY?

OBJECTIVE:

Give every row
one stable identity.

A primary key is the
column—or combination
of columns—that
identifies exactly
one row.

EXAMPLE: PEOPLE TABLE

id	name	city
1	Alex	Tokyo
2	Sam	Paris
3	Alex	Berlin
4	Priya	Toronto
5	Sam	Sydney

UNIQUE + NOT NULL

Names can repeat.
Row identities must not.



UNIQUE means no duplicates.
NOT NULL means every row
must have a value.

id	name	city
1	Alex	Tokyo
2	Sam	Paris
3	Alex	Berlin
4	Priya	Toronto
5	Sam	Sydney



THREE RULES



1 • UNIQUE



No two rows share the same key.

users

id	name	email	
1	Ava	ava@example.com	✓
2	Ben	ben@example.com	✓
3	Chloe	chloe@example.com	✓

2 • NOT NULL



Every row has a key value.

products

id	name	price	
101	Notebook	2.50	✓
102	Pen	1.20	✓
103	Mug	9.00	✓
NULL	Sticker	0.80	✗

3 • ONE PRIMARY KEY



A table has one primary key. It may use one column or several.

One column

books

id	title
10	Dune
11	1984
12	Foundation

Several columns

enrollments

student_id	course_id
1	CS101
1	MA101
2	CS101

Elly says:

The smallest stable identifier is usually easiest to use.





Good Data Builds Brighter Things ♥

CHOOSE A KEY THAT LASTS

RISKY: name or email


Alex Kim


alex@example.com

⚠ They can change.
They may repeat.

BETTER: customer_id


1047283

- ✓ Stable.
- ✓ Short.
- ✓ Meaning-free.



Foreign keys can point to this identity from other tables.

orders
order_id
customer_id
order_date
total

customers
customer_id
name
email
created_at



 **Guardrail:** //
never change a key just because a description changes.

Design Data for a Brighter Tomorrow ♥

Better Data Happens Here ♥

</>



WORKED EXAMPLE

Let's see a concrete example of a primary key using a customer and order table.

PROBLEM

?

Two customers are both named Alex.

Customers

customer_id	name	city
101	Alex	Tokyo
102	Alex	London

ACTION

Give them customer_id 101 and 102.

Customers

customer_id	name	city
101	Alex	Tokyo
102	Alex	London

CHECK

Add customer (duplicate id)

customer_id	name	city
101	Sam	Osaka

✗ Duplicate 101 → REJECTED

Add customer (missing id)

customer_id	name	city
—	Jamie	Berlin

✗ Missing id → REJECTED

IMPROVEMENT

Orders store customer_id, so each order points to the right Alex.

Orders

order_id	customer_id	item
5001	101	Latte
5002	102	Espresso



Customer 101
(Alex, Tokyo)

Their Orders

order_id	item
5001	Latte



Customer 102
(Alex, London)

Their Orders

order_id	item
5002	Espresso



WHAT IS A PRIMARY KEY?

RECAP:

A good primary key gives every row one dependable identity.

PRIMARY KEY CHECKLIST

- ☐ Identifies one row
- ☐ Unique
- ☐ Never NULL
- ☐ Stable over time
- ☐ Referenced consistently

ELLY'S FINAL TIP

Describe a row with other columns. Identify it with the key.



WHAT IS AN IDEMPOTENCY KEY?

Make retries safe.

OBJECTIVE:

Retry a lost reply without creating a second order.

1. SEND ONE ORDER REQUEST

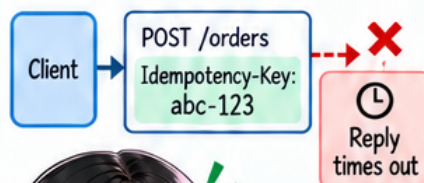
You send an order request with an idempotency key.



Include a unique idempotency key with the request.

2. THE REPLY TIMES OUT

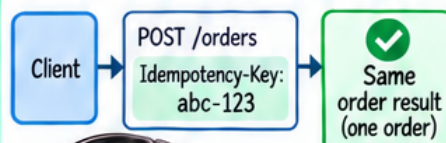
The server processes the order, but the reply is lost (e.g. timeout).



A lost reply is not proof of failure.

3. RETRY WITH THE SAME KEY

You retry with the **SAME** key. The server returns the same order result — not a second order.



Same key. Same result. No duplicates!

ONE OPERATION, ONE KEY.

Use one unique idempotency key for each logical operation. It's safe to retry, and it won't create a second order.

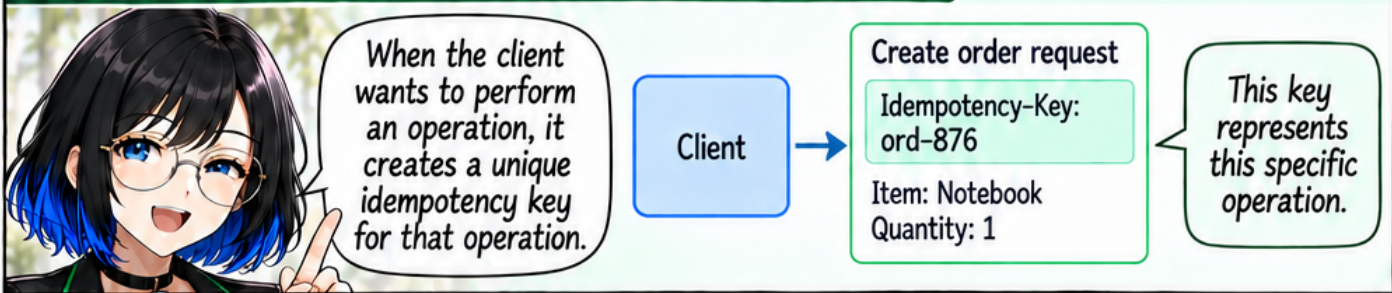
- ✓ Same key for retries
- ✓ One result, not two
- ✓ Safe and reliable
- ✓ Build with confidence

Resilient Systems Happier People

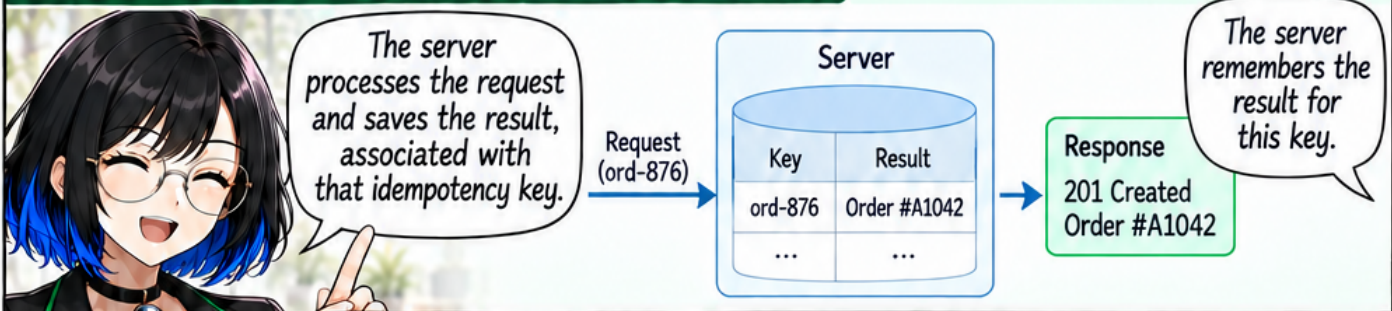




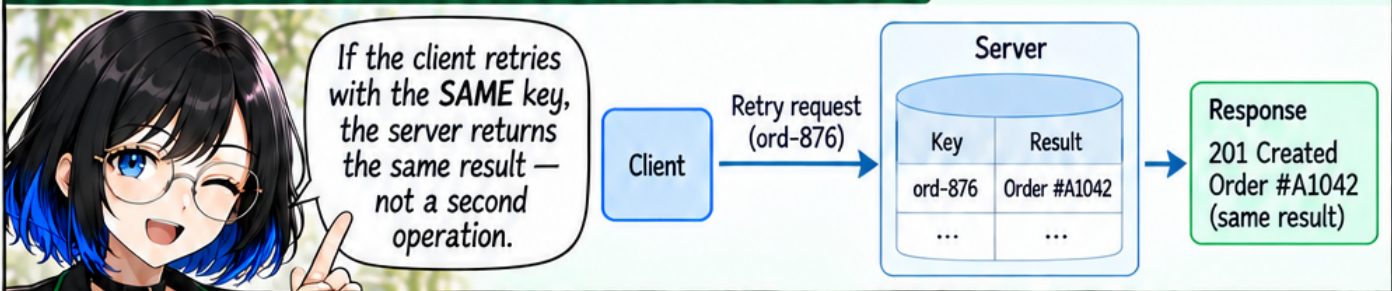
1. CLIENT CREATES A UNIQUE KEY FOR ONE OPERATION.



2. SERVER REMEMBERS THE RESULT FOR THAT KEY.



3. A RETRY WITH THE SAME KEY RETURNS THAT RESULT.



New operation?
New key!

A different operation needs a **NEW** unique key. Each operation should have its own idempotency key.

- ✓ Same operation, same key.
- ✓ Same result on retry.
- ✓ New operation, new key.
- ✓ Simple. Reliable. Safe.



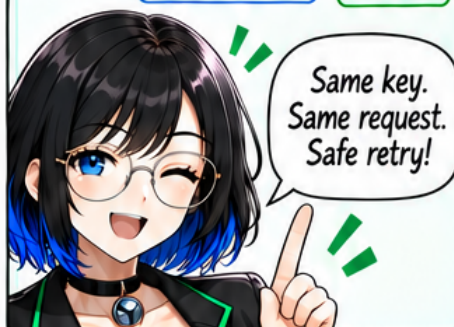
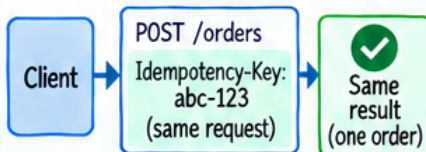


THREE RULES FOR SAFE RETRIES.

Simple rules.
Safer systems.

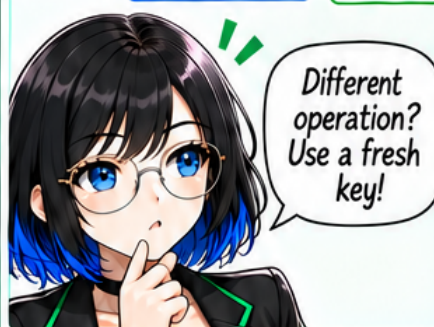
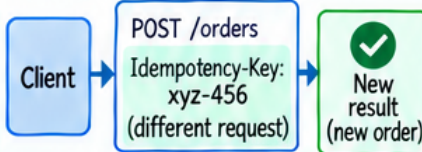
1. KEEP THE SAME KEY AND THE SAME REQUEST DETAILS.

When you retry a failed or lost request, use the **SAME** idempotency key and the **SAME** request details (e.g. same operation and data).



2. USE A FRESH KEY FOR A DIFFERENT OPERATION.

When you're performing a different operation (e.g. a new order or new set of data), use a **NEW** unique idempotency key.



3. CHECK THE PROVIDER'S KEY-RETENTION AND ERROR RULES.

Each provider may have its own rules for how long idempotency keys are kept and which errors are safe to retry. Check the provider's documentation.

- Provider Documentation**
- Key retention period
 - Which errors are safe to retry
 - Any special rules or limits



IMPORTANT GUARDRAIL:

Never put secrets in an idempotency key.

Do not include sensitive information such as passwords, API keys, tokens or personal data.

Keep keys safe and non-sensitive.

- ✓ Same key and same request details for retries.
- ✓ Fresh key for a different operation.
- ✓ Check the provider's key-retention and error rules.
- ✓ Never put secrets in an idempotency key.



Resilient
Systems
Happier
People



WORKED EXAMPLE: LOST REPLY.

Same key. One order.

Key takeaway:

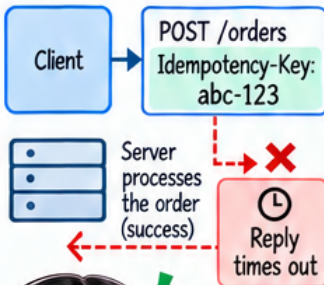
Retry with the same key.
You'll get the original order —
not a second one.

A lost reply
is not proof
that the order
failed.

Reliable
Retries
Happier
Systems

1. PROBLEM:

The order succeeds,
but the reply times out.



The order
was created,
but the reply
didn't arrive.

2. ACTION:

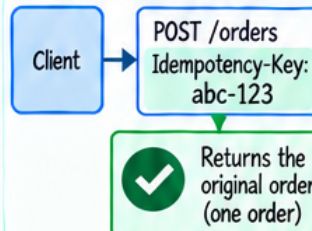
Retry the **SAME** order
with the **SAME** key.



Retry with
the same key
and the same
request details.

3. CHECK:

The service returns
the original order;
count **ONE** order.



Same key.
Same order.
One order!

4. IMPROVE:

Save the key until
the result is known.



Keep the idempotency
key until you know
the result (success
or error).

Keep the key
until the result
is known.

A lost reply is not
proof that the order failed.

Retry safely. Same key.
Same result. One order.

WHY THIS WORKS

- ✓ Retries are safe with the same key
- ✓ You get one result, not two
- ✓ Handles lost replies and timeouts
- ✓ More reliable and predictable systems
- ✓ Keep the key until the result is known

Resilient
Systems
Happier
People





RECAP: ONE OPERATION, ONE KEY.

Let's summarise!

KEY TAKEAWAYS

Simple rules.
Safer retries.
More reliable systems.

✓ 1. CREATE A UNIQUE KEY

Create a unique key for the operation.

One operation.
One key!

✓ 2. KEEP THE SAME KEY

Keep that key for retries of the SAME operation.

Same key for retries!

✓ 3. DO NOT REUSE FOR A NEW OPERATION

Do not reuse it for a NEW operation.

A different operation needs a different key.

✓ 4. CHECK SUPPORT AND RETENTION

Confirm the service supports idempotency keys and check its retention rules.

Check support and retention rules.



ELLY'S FINAL TIP:

A lost reply is not proof of failure.



Use idempotency keys.
Build with confidence.

Resilient
Systems
Happier
People



WHAT IS A WEBHOOK?

OBJECTIVE:

Learn how an event can trigger a message to your app.

1 Event happens.



2 Sender delivers a webhook.



3 Receiver acknowledges it.



An event in one system can automatically trigger a message to your app.

That's a webhook!

Same Ideas.
Clearer Learning.
A Brighter Tomorrow.
♡





1 A source detects a subscribed event.



2 It sends an HTTPS request with event data.



3 Your app receives it and returns a quick success response.





1 Use HTTPS and verify the sender's signature.



2 Ignore duplicate delivery IDs after the first success.



3 Acknowledge quickly; process longer work later.





1 Problem: Your shop gets a new order.



2 Action: It sends an order-created webhook.



3 Check: Your app accepts it and records one order.



4 Improve: Deduplicate a repeated delivery.



WEBHOOK RECAP

Here's a quick recap of how webhooks work. Keep these points in mind!

① Choose the events you need.



② Receive the HTTPS delivery.



③ Verify its signature.



④ Handle repeats safely.



Events call your app;
make every delivery
safe to replay.



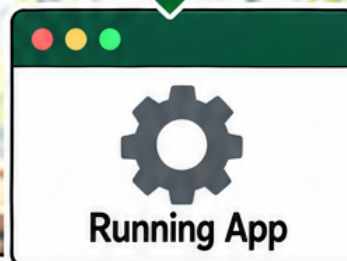
ELLY EXPLAINS: WHAT IS AN ENVIRONMENT VARIABLE?

OBJECTIVE:

See how a named setting reaches a running app.

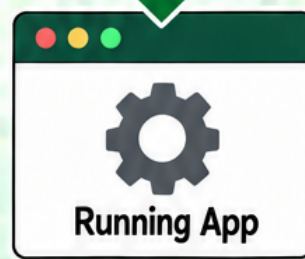
An environment variable is a name and value supplied to a process.

Name	Value
APP_MODE	demo



The app reads the name; the value can change without changing its code.

APP_MODE = demo



Change the value.
Same app.
New behaviour.



NAME, VALUE, PROCESS

1. Name the setting

Name
APP_MODE

2. Choose its value

Value
demo

3. Launch the app

Name	Value
APP_MODE	demo

Running App

The new process receives that setting.

An existing process does not automatically get later changes.

Good
Settings
Better
Tomorrow
♥



THREE SIMPLE RULES

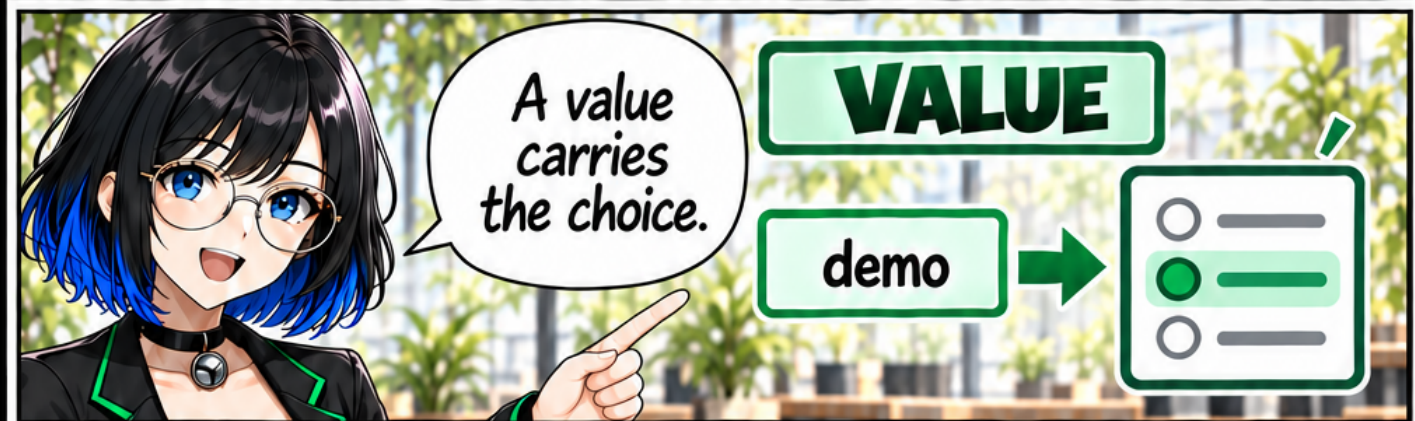


Settings
Power
Better
Apps
♡

A name identifies the setting.

NAME

APP_MODE → 



Good
Settings
Better
Tomorrow
♡

A value carries the choice.

VALUE

demo → 



Good
Settings
Better
Tomorrow
♡

A new app receives the pair when it starts.

PROCESS

APP_MODE
demo → 



Good
Settings
Better
Tomorrow
♡

Keep passwords out of screenshots and shared notes.







> A SMALL WORKED EXAMPLE <

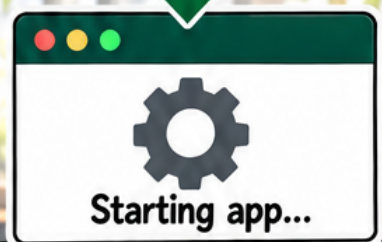


PROBLEM:
A demo app needs a practice mode.




ACTION:
Give the new process `APP_MODE` with value `demo`.

Name	Value
<code>APP_MODE</code>	<code>demo</code>




CHECK:
The app shows its demo banner.




IMPROVE:
Keep the same code; choose a different value for a new run.

Name	Value
<code>APP_MODE</code>	<code>tutorial</code>

Same app.
New behaviour.




RECAP: A SETTING FOR A NEW RUN

1



Name the setting.

2

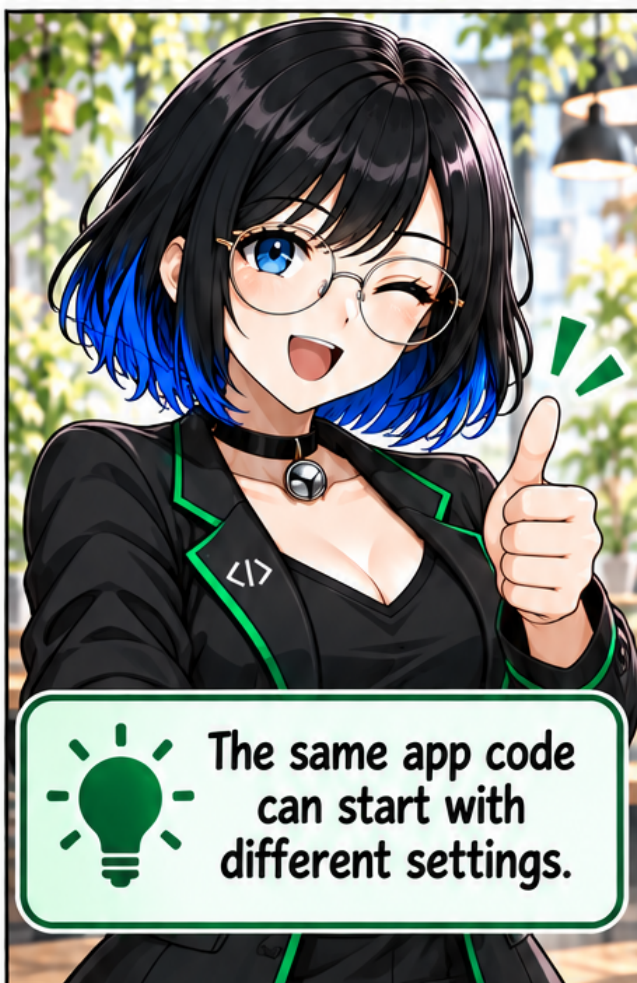


Choose its value.

3



Start the app and check its behaviour.



The same app code
can start with
different settings.



Elly's tip:
Change the value,
then start a
NEW process
to use it.



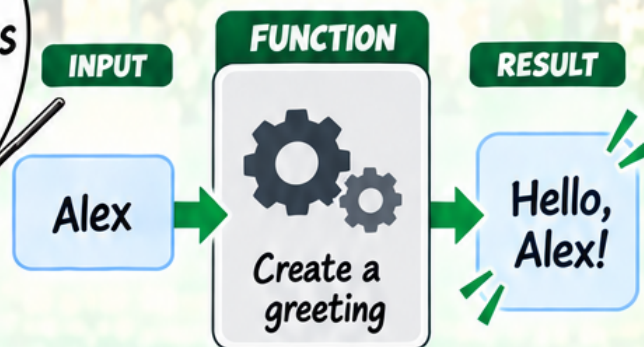
WHAT IS A FUNCTION?

OBJECTIVE:

Name a repeatable task, give it input, get a result.

A function is a named set of steps you can call again.

A SIMPLE EXAMPLE



Give a task a name, provide input, and get a result. That's a function!

Same Ideas. Clearer Tomorrow. ♥

LEARN
PRACTICE
BUILD
A BRIGHTER TOMORROW





1 • NAME



A clear name says what the task does.

NAME

greet

2 • INPUT



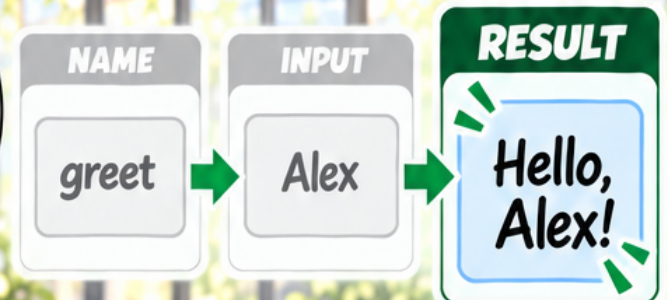
Give the task a value to work with.



3 • RESULT



The function returns a value you can use.



Start small.
Reuse the
task.

KEEP ONE TASK CLEAR!

1 Choose a clear name



greet is clearer
than doThing.

2 Give only needed input



A name is enough
for this greeting.

3 Return a useful result



Let the caller
decide what to
do next.



ONE GREETING, MANY NAMES

Let's write a simple function that gives a greeting to anyone.

Better
Ideas
Brighter
Tomorrow
♡

Same
Ideas.
Clearer
Tomorrow.
♡

PROBLEM

?

We need a greeting for each visitor.

ACTION

Define `greet(name)` once.

Python • Linux / macOS / Windows

```
def greet(name):  
    return "Hello, " + name
```

CHECK

`greet("Alex")` →
`"Hello, Alex"`

```
>>> greet("Alex")  
'Hello, Alex'
```

IMPROVE

Call the same function with a new name.

```
>>> greet("Jamie")  
'Hello, Jamie'
```



Better
Software
Brighter
Tomorrow
♡

FUNCTION CHECKLIST!

A function has a name, input, and a result.

1



Name the task clearly.

2



Pass only needed input.

3



Use the returned result.

If you repeat a task, give it a name!

Same Ideas.
Clearer
Tomorrow.

LEARN
PRACTICE
BUILD
A BRIGHTER TOMORROW

Good Data
Brighter Days

