Don’t make me think!

The UX of your code

By Johan Beyers

takealot.com

Why?

takealot.com

We read more code than writing it

Keeping context in your head is HARD

Make it easy

Make it automatic

Background

takealot.com

Microservices on Kubernetes

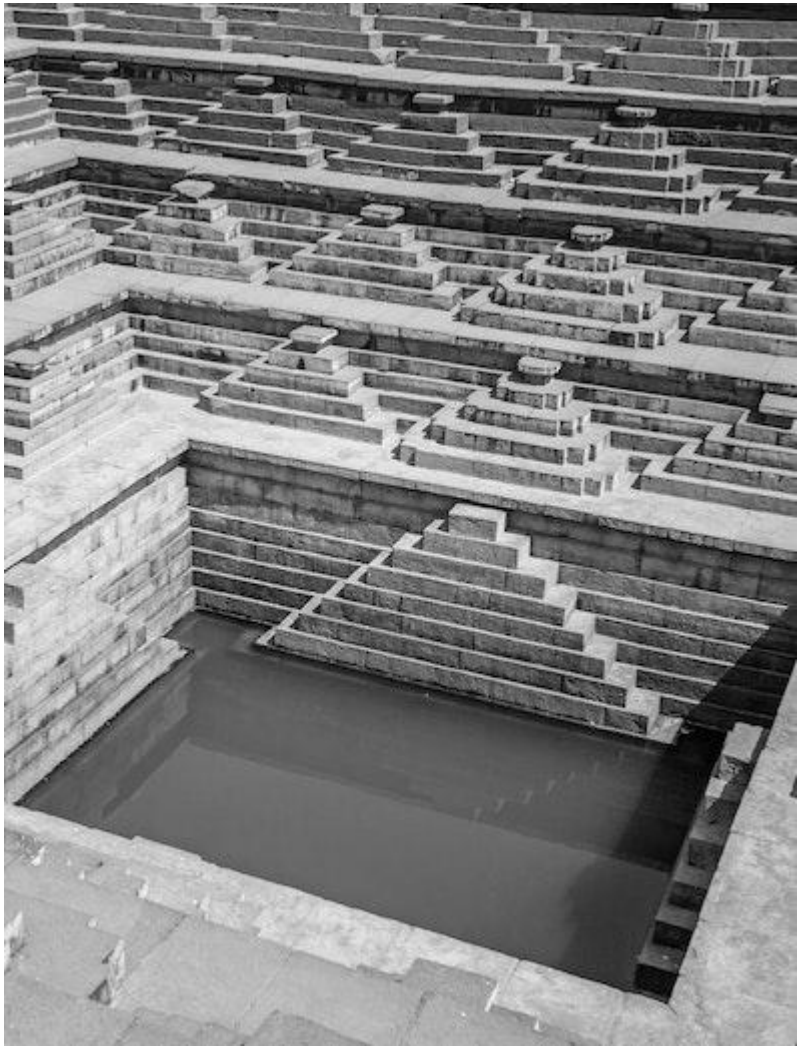
Python/React

MySql/PostgreSql/Kafka

Many teams

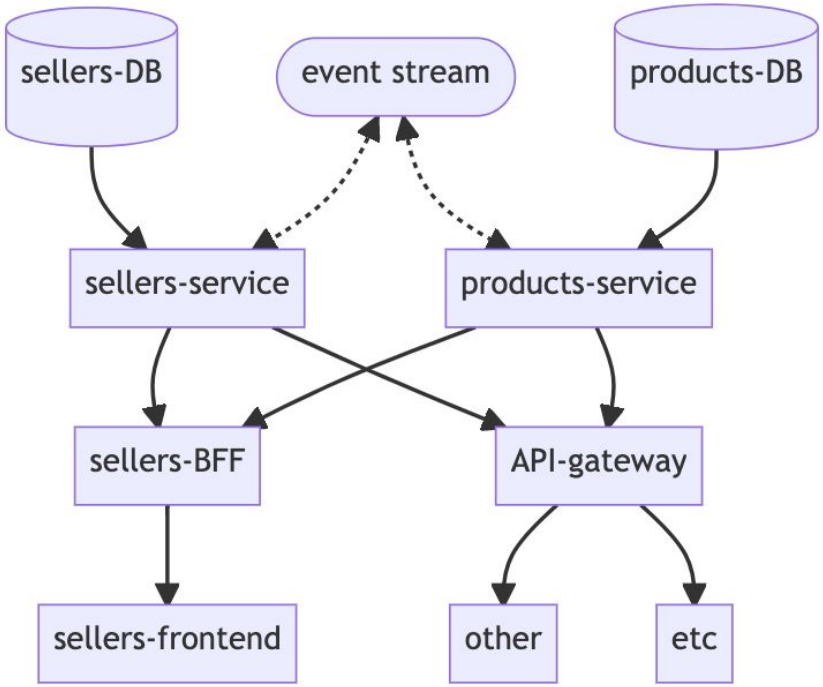
570 repositories (and counting)

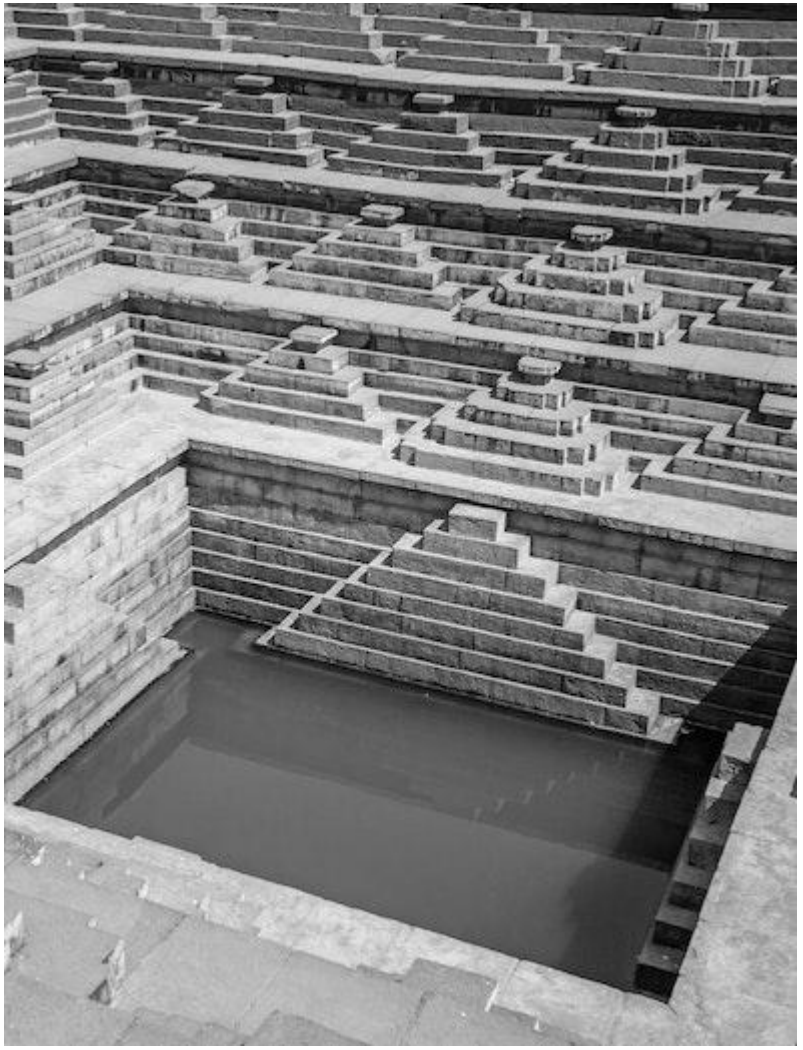
300+ engineers (and hiring)



Background

takealot.com





How?

takealot.com

CONSISTENCY

Naming things

Variables for Humans

Comments

Code organisation

Consistency

takealot.com

We are pattern matching machines

Consistent patterns become automatic

Inconsistent patterns pull us out of flow state

Consistency: Code Look and Feel

takealot.com

Project layouts
(cookiecutters / templates)

Setup
(Makefile / Setup scripts)

Formatting
(Black / Flake8 / isort)

Use of libraries
(Yes, you'd be surprised)

Consistency: Return Boring Results

takealot.com

Fail loudly

Fail loudly

Return [1,2,3] OR [], NOT None

Return all the keys

Return similar structures

Consistency: Boring Results examples

takealot.com

```
def list_foos():
    # ...
    if check_items_for_mumps(foos):
        raise MumpsError()
    return {"items": foos, "messages": ["foo 1 is
sad"]}


def list_bars():
    # ...
    return {"items": bars, "messages": "bar 1 is
sad"}


def list_bleps():
    # ...
    if check_items_for_mumps(bleps):
        return None
    return {"bleps": bleps}
```

Naming Things

takealot.com

There are only two hard things in Computer Science: cache invalidation and naming things. -- Phil Karlton

Misleading names is a MAJOR source of confusion

There are many resources

Naming Things: Variables

takealot.com

Read it in an if statement

Don't clobber builtins (list, id)

Don't reassign to the same name

If it changes from single to plural, RENAME IT

Naming Things: Conventions

takealot.com

Django ORM:
`__eq` / `__in`

SQL:
`count` / `limit` / `offset`

CRUD:
`create_` / `get_` / `list_` / `update_` / `delete_`

Naming Things: Functions

takealot.com

Read it in an if statement

Use `verb_object` pattern
(`list_foos`, `delete_bar`)

Should not sound like a variable

Not too long, if possible

Naming Things: Patterns

takealot.com

Use the same pattern for similar calls

- Easy to document
- Understand one, understand all

Return values too

Do not stray from this

Naming Things: Across services

takealot.com

Be absolutely consistent inside your stack

Translate at the edges

Use YOUR terminology

Change it all at once

Naming Things: Example

takealot.com

What do these have in common?

- idSKU
- sku_id
- product_id
- pid
- seller_listing_id
- offer_id

Naming Things: Patterns example

takealot.com

```
def list_dogs(filters):
    return upstream_client.get_doggos(filters)


def get_list_of_birds_filtered(filters):
    birds = upstream_client.get_birbs(filters)
    count = len(birds)
    return {"birds": birds, "count": count}
```

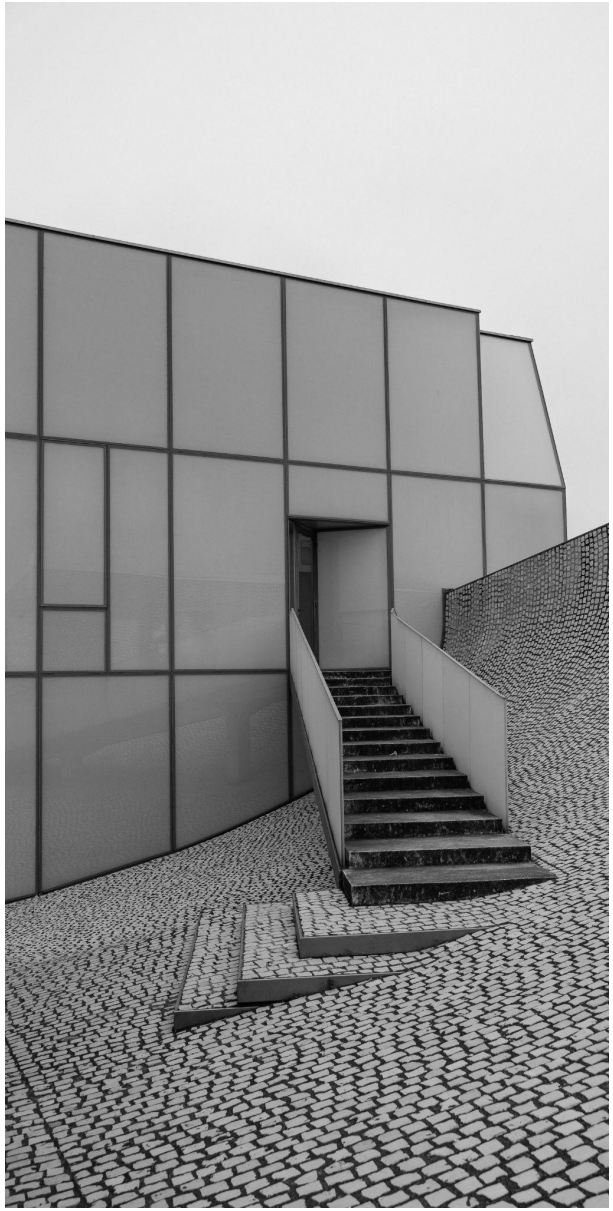
Variables for Humans

takealot.com

Words have meaning.

Humans care about meaning.

Machines don't.



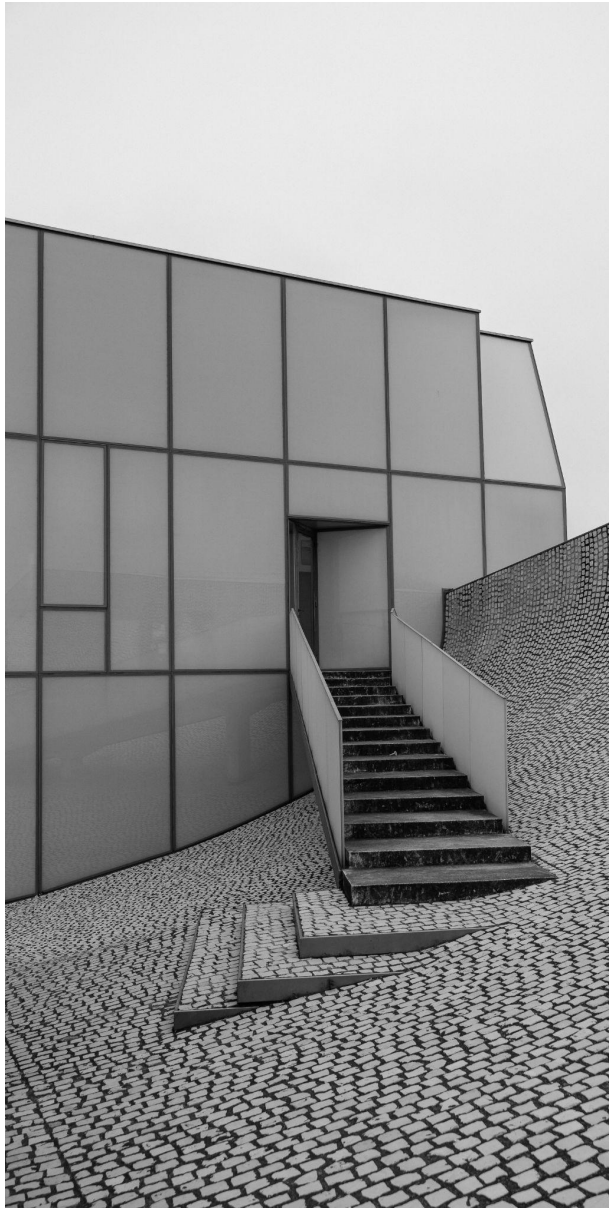
Variables for Humans: Pagination

takealot.com

Page numbers start at 1

Offset starts at 0

```
# Translate human to machine
offset = (page_number - 1) * page_size
limit = page_size
```



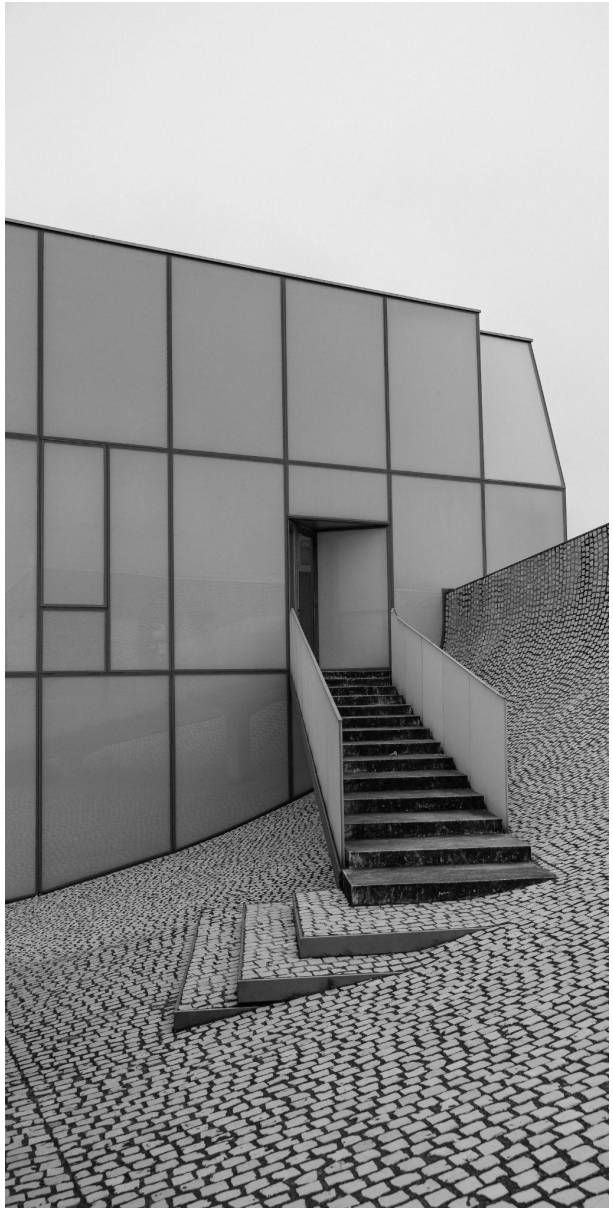
Variables for Humans: Dates

takealot.com

Dates are a nightmare:

- Human date ranges are from the START of the first day to the END of the last day.
- SQL dates are INCLUSIVE, but to the START of the last day.
- Anything other than yyyy-mm-dd leads to insanity.
- Watch out for timezones.

If you test ANYTHING, test THIS



Variables for Humans: Percentages

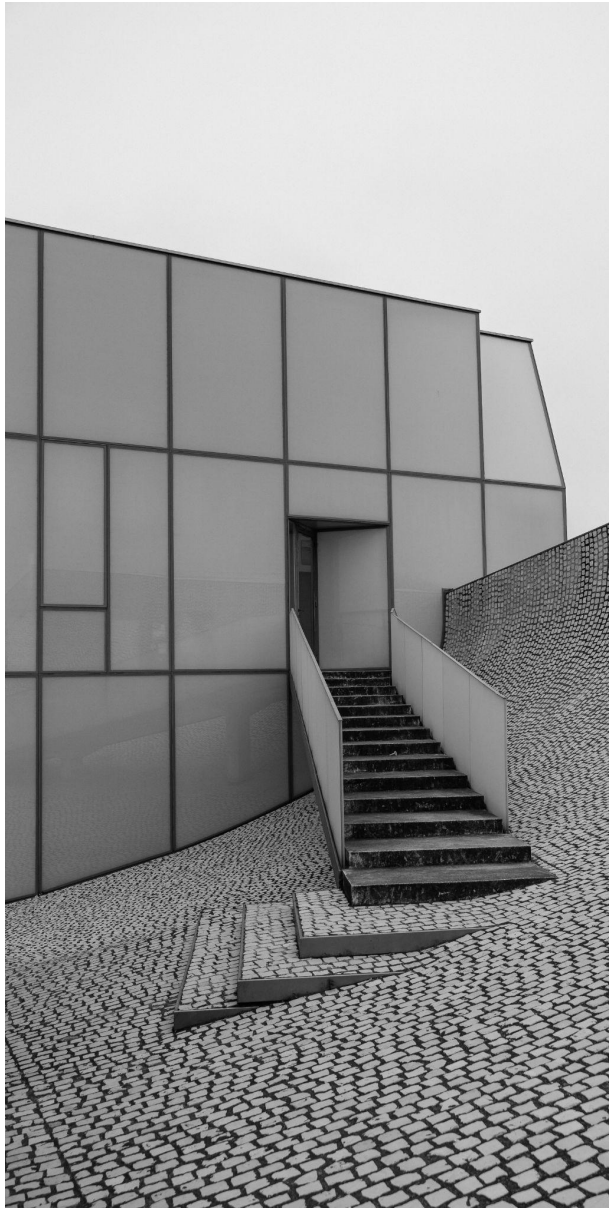
takealot.com

`discount` has no meaning

`discount_percentage` has a clear meaning

`discount_fraction` is ugly, but it works

Same goes for `markup`, `storage_fee` etc.

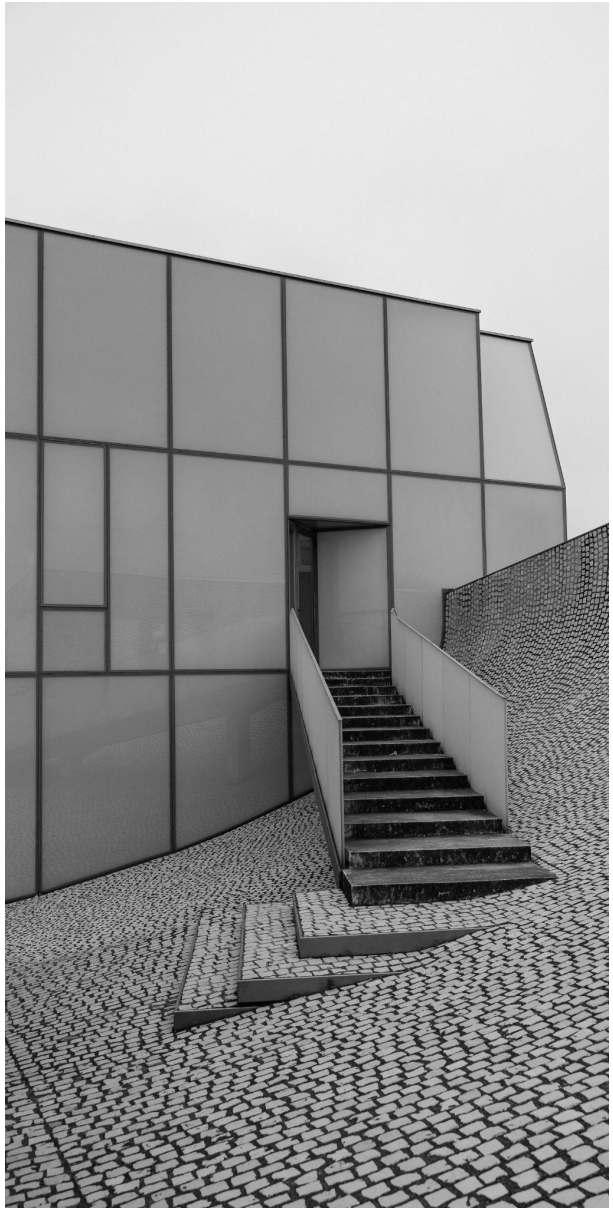


Variables for Humans: Units

takealot.com

```
# Good
width_cm = 10.0

# Bad
width = 10.0

# Ugly
width = {"value": 10.0, "unit": "cm"}
```

(All of these are in use at Takealot)



Variables for Humans: Addresses

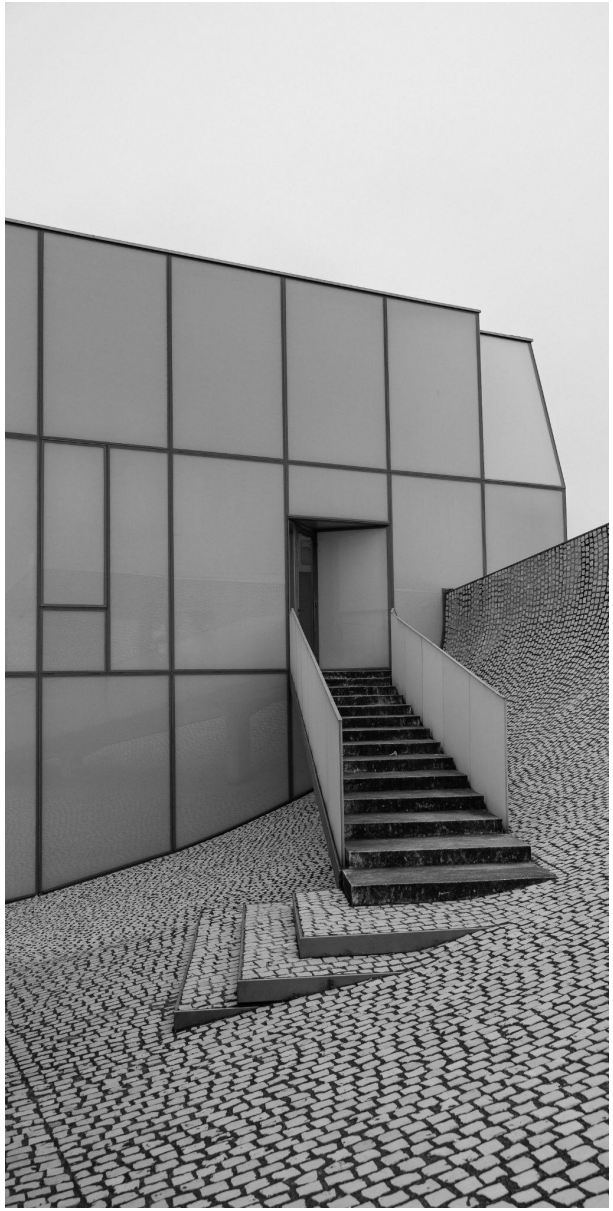
takealot.com

We all mess this up

Use a list of strings unless you need more meaning

You don't need more meaning

postal code/zipcode is NOT AN INT



Variables for Humans: Booleans

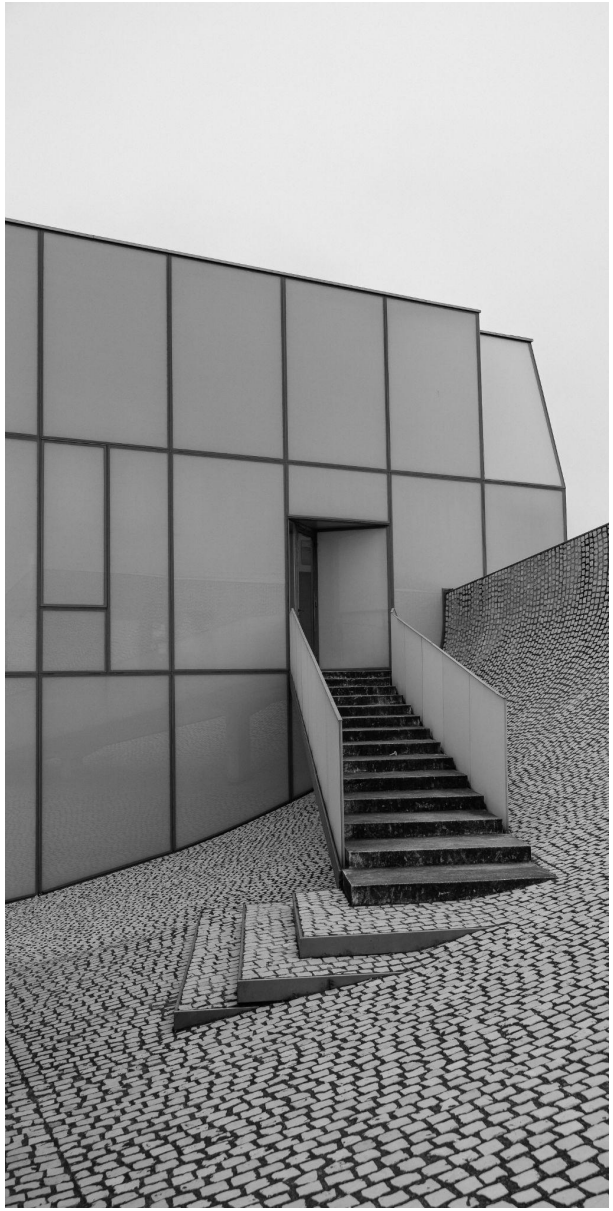
takealot.com

Use a prefix:

- `in_stock = 10` seems reasonable.
- `is_in_stock = 10` is clearly not.

Be positive:

- `has_age_limit`
- `contains_nuts`
- `can_be_rattled`



Variables for Humans: Constants

takealot.com

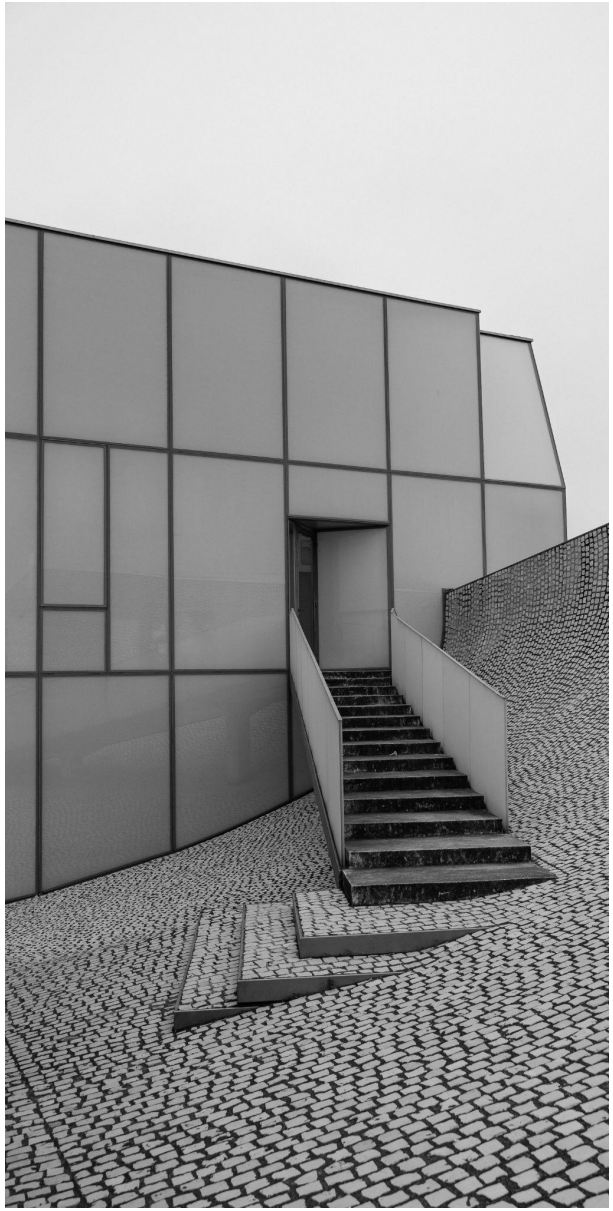
Use constants

At the top of the file

Comment each one

Group using Enums

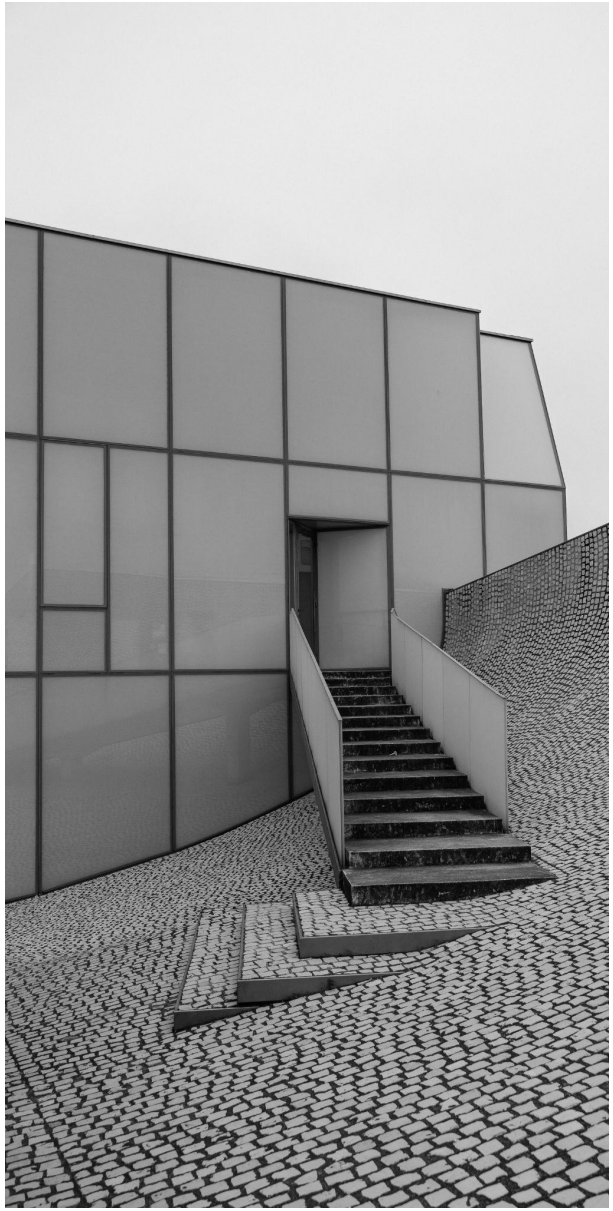
"Explicit is better than implicit" - Tim Peters, The Zen of Python



Variables for Humans: Constants

takealot.com

```
# We can only deliver after a few days
MINIMUM_DELIVERY_TIME_SECONDS = 60*60*24*5

class Regions(enum.Enum):
    # Old Cape Town Warehouse id, used everywhere
    CPT = 1
    # Old Joburg Warehouse id, used everywhere
    JHB = 3
```

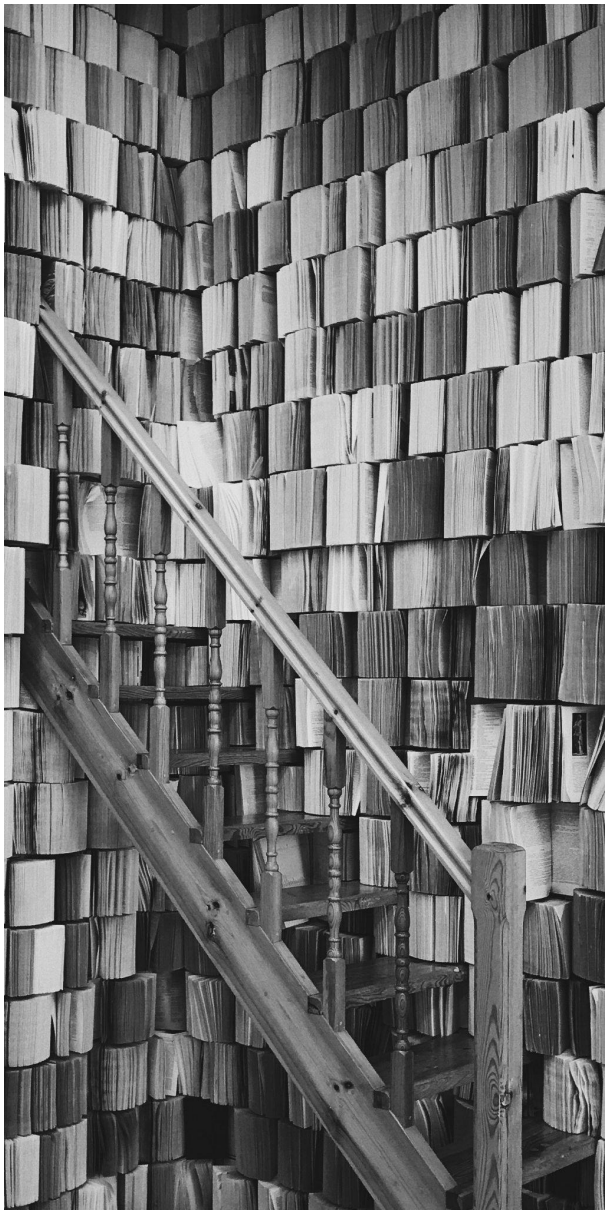


Comments

"Don't ever take a fence down until you know the reason it was put up." - G. K. Chesterton

"Readability counts" - Tim Peters, The Zen of Python

"Sparse is better than dense" - Tim Peters, The Zen of Python



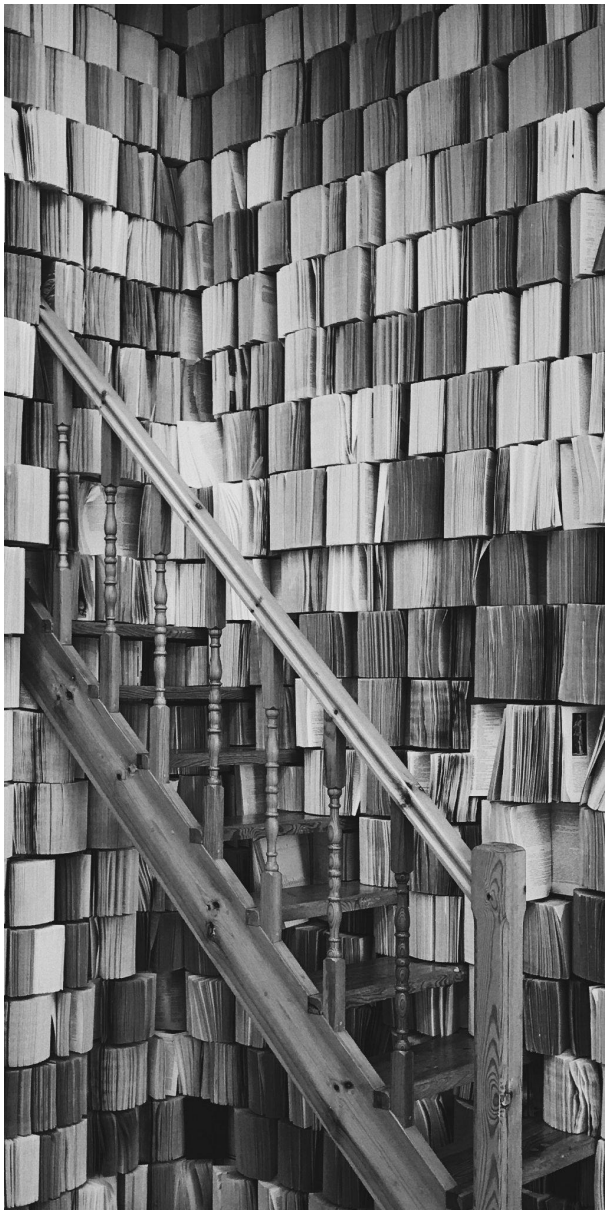
Comments: Guidelines

takealot.com

Document the WHY, or clarify the WHAT

Speak to the reader

Use comments like paragraph breaks



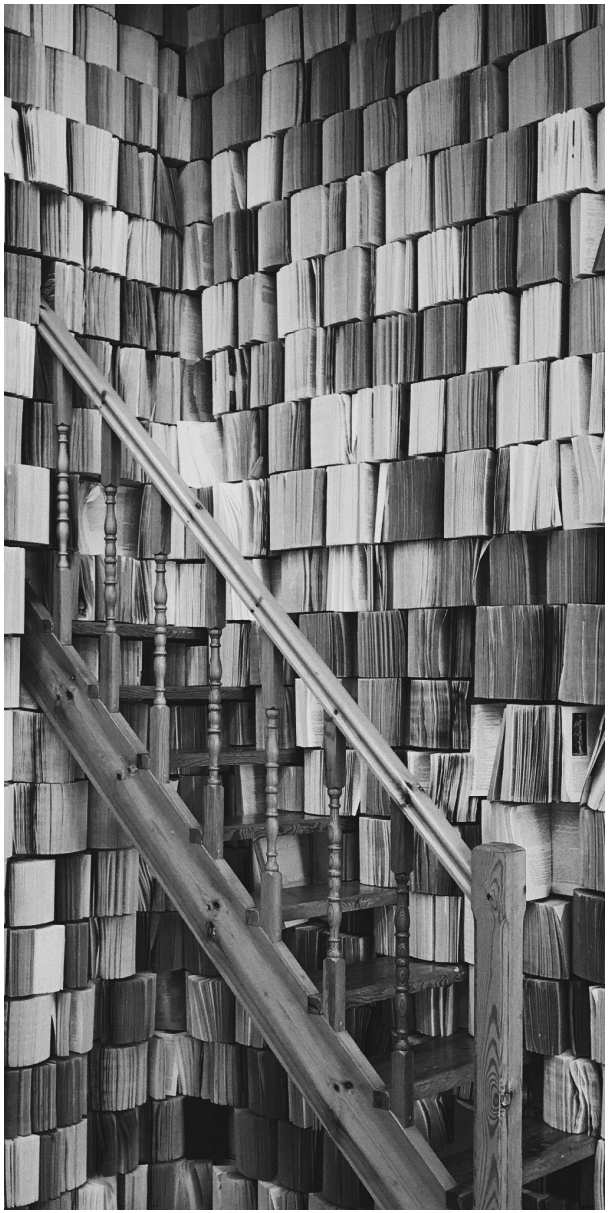
Comments: Specific types

takealot.com

TODOs
- May not get done
- Shorthand for 'This might not do what you expect'

References
- Show your research
- No shame in finding info on the internet

Rants
- provide context on WHY you are doing something
- Fun to read



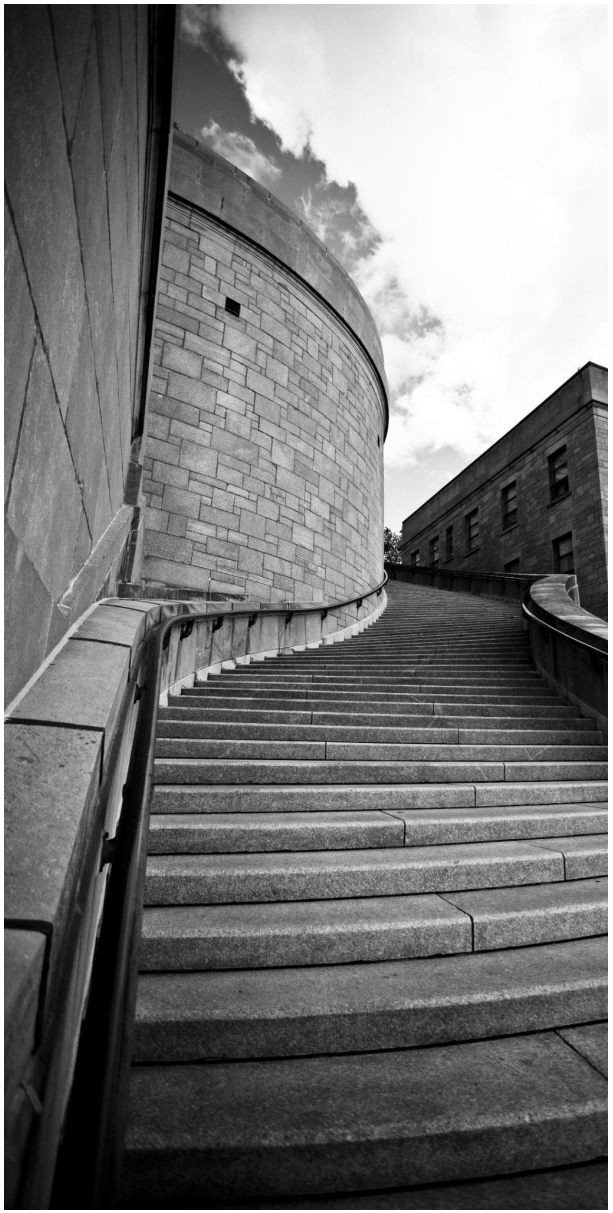
Code Organisation

takealot.com

Well-organised code is easier to understand.

Cognitive complexity is a thing.

"Flat is better than nested" - Tim Peters, The Zen of Python

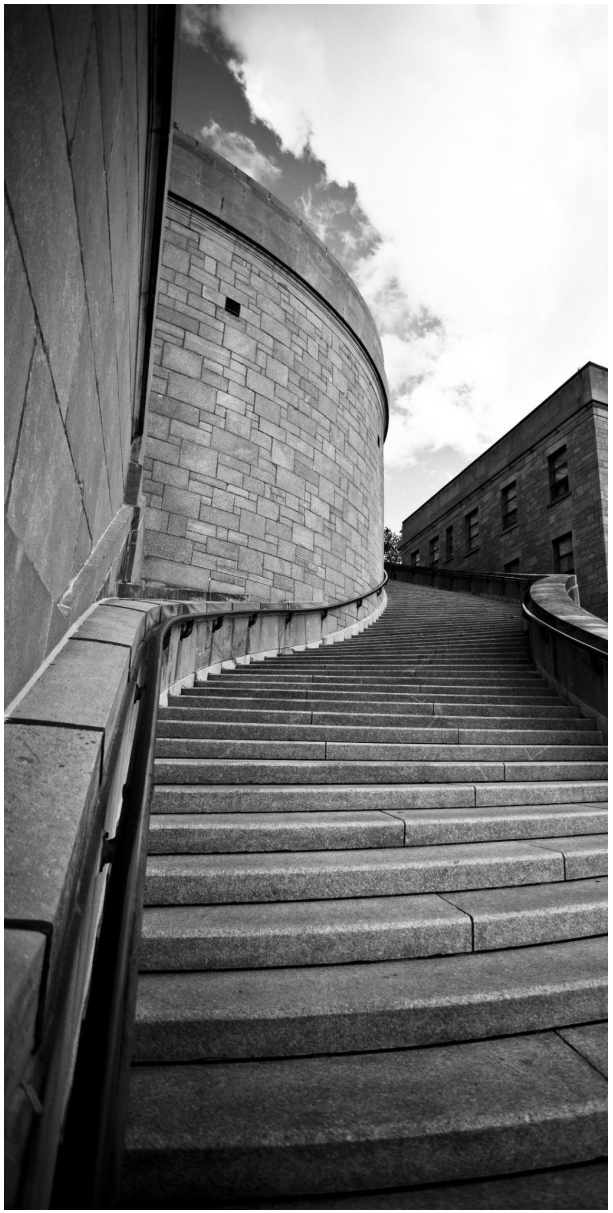


Code Organisation: Files and Classes

takealot.com

Gradually refactor upwards

- Refactor to just above where it's used
- Private and Common methods first
- Top of the file
- Move to a different file



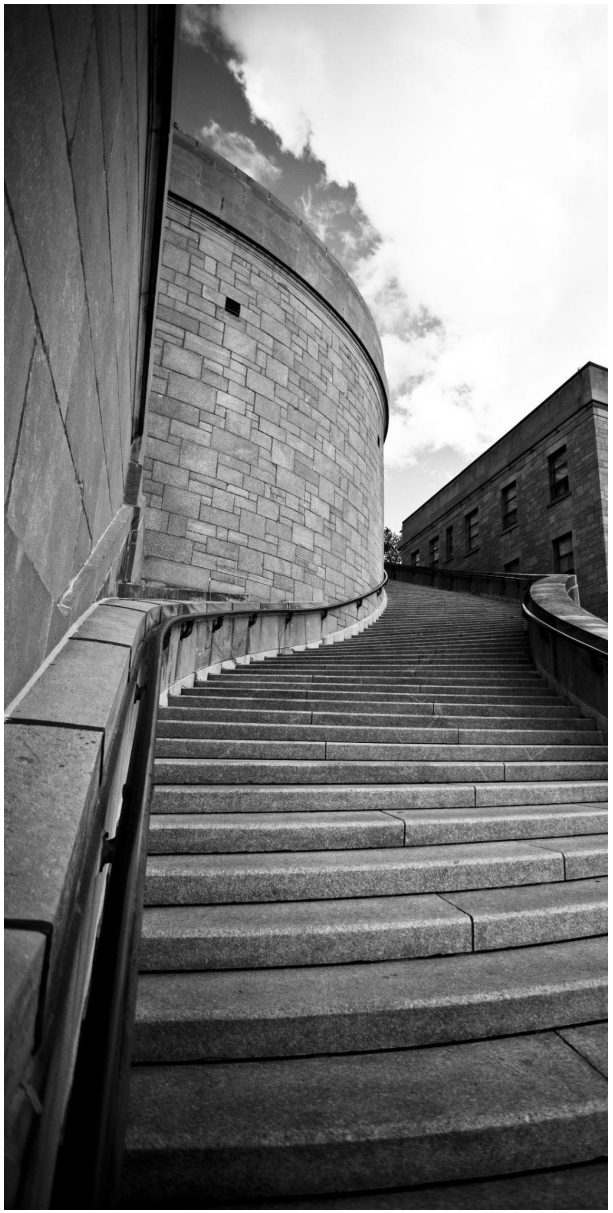
Code Organisation: Logic

takealot.com

Input cleaning and validation

Fail and return early

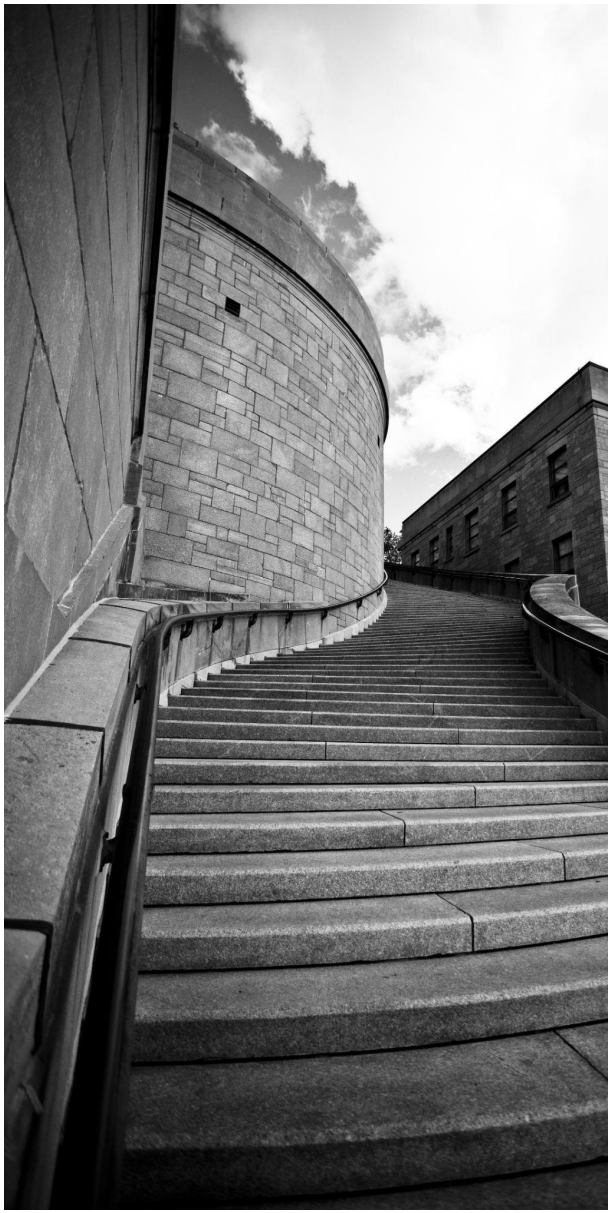
Happy path



Code Organisation: Logic Example

takealot.com

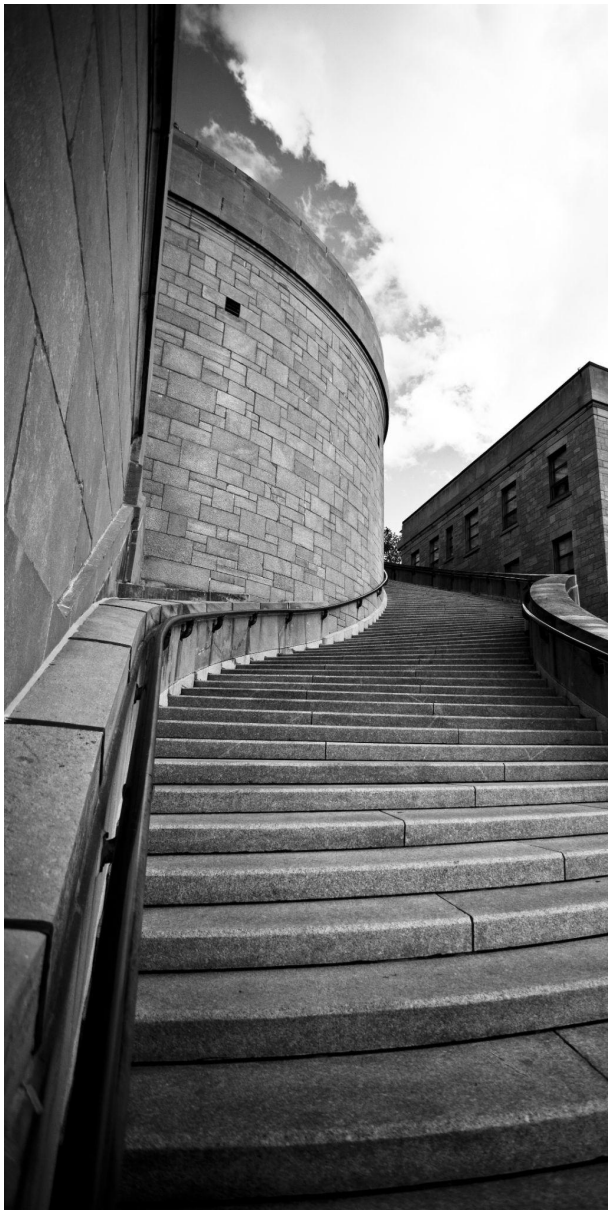
```
def get_confirmed_purchase_order(purchase_order_reference: str):
    """
    Get a full purchase order from the PO reference.
    Expects the input to be a string starting with 'PO-' (case-insensitive)
    TODO: Add in params and types
    """

    # Clean and validate

    # Remove common errors from human input (Postel's Law)
    cleaned_reference = purchase_order_reference.strip().upper()

    # TODO: The "PO-" prefix is not needed, so maybe accept just a number?
    if not cleaned_reference.startswith("PO-"):
        raise ValueError(
            "PO number %s does not start with PO-" % purchase_order_reference
        )

    try:
        # Convert to the terminology we use in the rest of the method.
        purchase_order_id = int(cleaned_reference[3:])
    except ValueError:
        raise ValueError(
            "PO number %s is not numeric after the PO- bit" %
            purchase_order_reference
        )
```



Code Organisation: Logic Example

takealot.com

```
    # PO with items, populated.
    purchase_order_dict = purchase_order_client.get_purchase_order(
        purchase_order_id=purchase_order_id
    )

    # Fail and return early

    # For some stupid reason, the upstream client returns an empty dict if
    # the order is not confirmed, but a None if it does not exist.
    if purchase_order_dict is None:
        raise ValueError("PO %s does not exist" % purchase_order_reference)

    # This is ugly, but our downstream system understands it and handles it.
    if not purchase_order_dict:
        return {
            "purchase_order_dict": {},
            "error_message": f"{purchase_order_reference} is not confirmed
yet",
        }

    # Happy path

    # This was shamelessly stolen from the supplier order service here:
    # https://github.com/blah/blah/filename.py#316
    # ... Do something like figure out discounts, add in shipping costs, etc.

    return {"purchase_order_dict": purchase_order_dict, "error_message": ""}
```



Why?

takealot.com

Software is written by humans, for other humans (and occasionally computers)

Think beyond the small change you're making right now.

People will do it wrong. Learn from that.

We are hiring!