ACADE MIND

Naming “Things”

Variables, Constants & Properties

Functions & Methods

Classes

ACADE MIND

Names should be **meaningful**

Why Names Matter

Well-named “Things” allow readers to **understand your code without going through it in detail**

```
const user = new User()
```

```
database.insert(user)
```

```
if (isLoggedIn) { … }
```

To understand the above code, we don’t need to go through the full class or function definitions and all the other code

We'll Not Always Agree!

```
const admin = new Admin();
```

This is readable

And so is this

```
const admin = new AdminUser();
```

How To Name Things Correctly

Variables & Constants	Functions / Methods	Classes
Data containers	Commands or calculated values	Use classes to create “things”
e.g. user input data, validation results, a list of products	e.g. send data to server, check if user input is valid	e.g. a user, a product, a http request body
Use **nouns** or short phrases with **adjectives**	Use **verbs** or short phrases with **adjectives**	Use **nouns** or short phrases with **nouns**

`const userData = { … }`
`const isValid = …`
	`sendData()`
	`inputIsValid()`
		`class User { … }`
		`class RequestBody { … }`

Name Casing

snake_case	camelCase	PascalCase	kebab-case
`is_valid` `send_response`	`isValid` `sendResponse`	`AdminRole` `UserRepository`	`<side-drawer>`
e.g. Python	e.g. Java, JavaScript	e.g. Python, Java, JavaScript	e.g. HTML
Variables, functions, methods	Variables, functions, methods	Classes	Custom HTML Elements

Naming Variables, Constants & Properties

Value is an Object	Value is Number or String	Value is a Boolean
Describe the value	Describe the value	Answer a true/ false question
`user` `database`	`name` `age`	`isActive` `loggedIn`
Provide more details without introducing redundancy	Provide more details without introducing redundancy	Provide more details without introducing redundancy
`authenticatedUser` `sqlDatabase`	`firstName` `age`	`isActiveUser` `loggedIn`

Examples – Variable Names

What is stored?	Bad Names	Okay Names	Good Names
A user object (name, email, age)	`u` `data`	`userData` `person`	`user` `customer`
	”u” and “data” could contain anything	“userData” is a bit redundant, “person” is too unspecific	”user” is descriptive, “customer” is even more specific
User input validation result (true/ false)			

Examples – Variable Names

What is stored?	Bad Names	Okay Names	Good Names
A user object (name, email, age)	`u` `data`	`userData` `person`	`user` `customer`
	"u" and "data" could contain anything	"userData" is a bit redundant, "person" is too unspecific	"user" is descriptive, "customer" is even more specific
User input validation result (true/ false)	`v` `val`	`correct` `validatedInput`	`isCorrect` `isValid`
	"v" could be anything, "val" could also stand for "value"	Both terms don't necessarily imply a true/ false value	Descriptive and value type is clear

ACADEMIND

Naming Functions & Methods

Function performs an operation	Function computes a Boolean
Describe the operation	Answer a true/ false question
`getUser(…)` `response.send()`	`isValid(…)` `purchase.isPaid()`
Provide more details without introducing redundancy	Provide more details without introducing redundancy
`getUserByEmail(…)` `response.send()`	`emailIsValid(…)` `purchase.isPaid()`

Examples – Function / Method Names

What does the function do?	Bad Names	Okay Names	Good Names
Save user data to a database	`process(…)` `handle(…)`	`save(…)` `storeData(…)`	`saveUser(…)` `user.store(…)`
	Both are very unspecific – what is being "processed"?	At least we know that something is saved – but what?	The intent is very clear – especially with the method
Validate the user input			

ACADEMIND

Examples – Function / Method Names

What does the function do?	Bad Names	Okay Names	Good Names
Save user data to a database	`process(…)` `handle(…)`	`save(…)` `storeData(…)`	`saveUser(…)` `user.store(…)`
	Both are very unspecific – what is being “processed”?	At least we know that something is saved – but what?	The intent is very clear – especially with the method
Validate the user input	`process(…)` `save(…)`	`validateSave(…)` `check(…)`	`validate(…)` `isValid(…)`
	Unspecific (“process”) or even misleading (“save”)	Both names are not 100% specific	Both makes sense – depends on what the function does exactly

Naming Classes

Describe the Object

```
User
Product
```

Provide more details without introducing redundancy

```
Customer
Course
```

Avoid redundant suffixes → Classes are typically instantiated

```
DatabaseManager
```

→ Instantiating a “DatabaseManager” makes no sense

Examples – Class Names

Which object is described?	Bad Names	Okay Names	Good Names
A User	`class UEntity` `class ObjA`	`class UserObj` `class AppUser`	`class User` `class Admin`
	Both are very unspecific	Both class names have redundant information	“User” is just fine – or “Admin” if it’s a more specific kind of user
A Database *(in code)*			

Examples – Class Names

Which object is described?	Bad Names	Okay Names	Good Names
A User	`class UEntity` `class ObjA`	`class UserObj` `class AppUser`	`class User` `class Admin`
	Both are very unspecific	Both class names have redundant information	“User” is just fine – or “Admin” if it’s a more specific kind of user
A Database *(in code)*	`class Data` `class DataStorage`	`class Db` `class Data`	`class Database` `class SQLDatabase`
	It’s not clear that we’re describing a database	Not 100% specific	“Database” is good, ”SQLDatabase” might be even better

Don’t Include Redundant Information In Names

```
userWithNameAndAge = User(‘Max’, 31)
```

Even without knowing the class definition, it’s easy to guess that this user has a name and age

In general, it’s expected that a “User” will contain some user data

We should look into the class definition if we want to learn more about the “User” object

Names should avoid describing unnecessary or redundant details

```
user = User(‘Max’, 31)
(newUser, loggedInUser)
```

ACADEMIND

Avoid Slang, Unclear Abbreviations & Disinformation

	✗ Avoid	✓ Do

Slang	`product.diePlease()`
	`user.facePalm()`
		`product.remove()`
		`user.sendErrorMessage()`

Unclear Abbreviations	`message(n)`
	`ymdt = ‘20210121CET’`
		`message(newUser)`
		`dateWithTimezone = ‘20210121CET’`

Disinformation	`userList = { u1: …, u2: … }`
	`allAccounts = accounts.filter()`
		`userMap = { u1: …, u2: … }`
		`filteredAccounts = accounts.filter()`

Choose Distinctive Names

✗

```
analytics.getDailyData(day)
analytics.getDayData()
analytics.getRawDailyData(day)
analytics.getParsedDailyData(day)
```

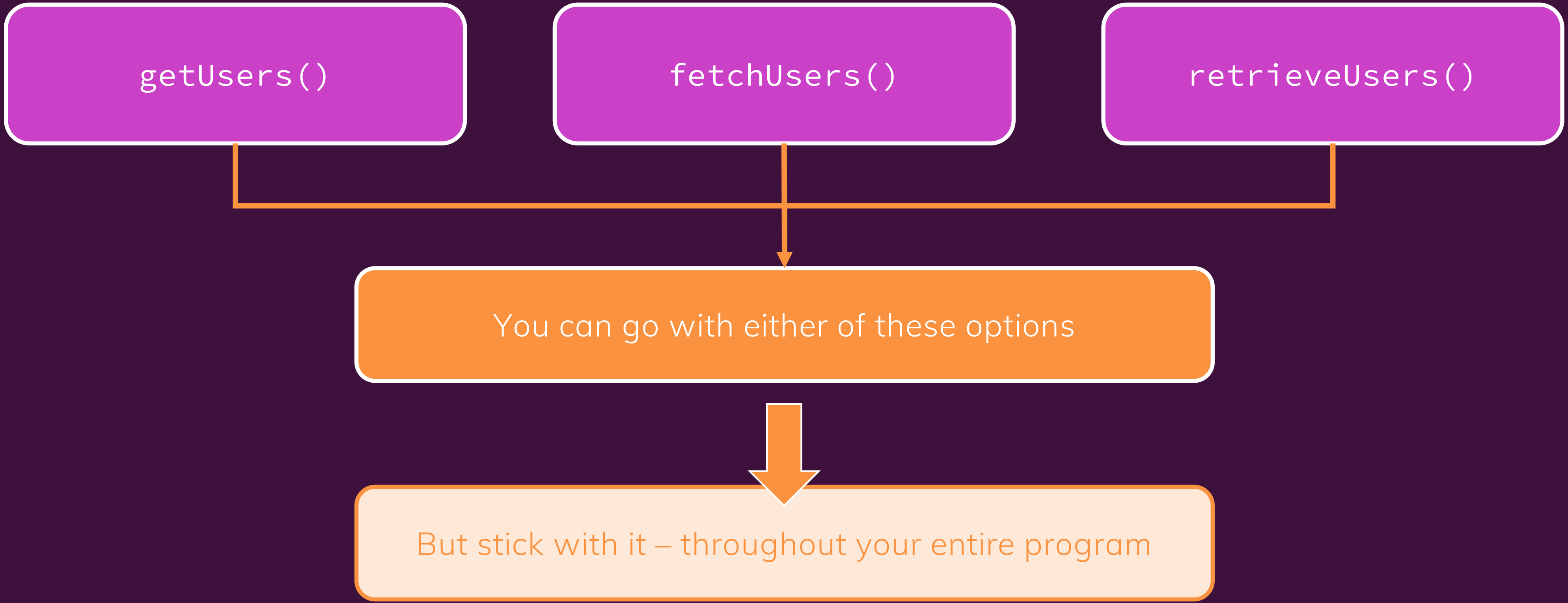
These methods all sound very similar, it’s hard to tell when you would use which method

✓

```
analytics.getDailyReport(day)
analytics.getDataForToday()
analytics.getRawDailyData(day)
analytics.getParsedDailyData(day)
```

All methods are very distinct from each other, it’s easy to choose when to call which method

Be Consistent



getUsers()

fetchUsers()

retrieveUsers()

You can go with either of these options

But stick with it – throughout your entire program