```python
if response.status_code == 200:
    success()
elif response.status_code == 400:
    bad_request()
elif response.status_code == 401:
    redirect_to_login()
elif response.status_code == 429:
    wait_and_repeat()
```

grape up

```
switch(response.statusCode){
    case 200: success(); break;
    case 400: badRequest(); break;
    case 401: redirectToLogin(); break;
    case 429: waitAndRepeat(); break;
}
```

grape up

```
match response.status_code:
    case 200:
        success()
    case 400:
        bad_request()
    case 401:
        redirect_to_login()
    case 429:
        wait_and_repeat()
```

grape up

grape up

# Pattern Matching

Marcin Dąbrowski

# Regexp

grape up

```
/[a-zA-Z0-9]+@[a-zA-Z0-9]+.(pl|com)/g
```

```
marcindabrowski94@gmail.com
marcin.dabrowski@gmail.com
marcin!@gmail.com
marcin@gmail.net
```

# Pattern Matching

grape up

```
1 match response.status_code:
2     case 200:
3         success()
4     case 400:
5         bad_request()
6     case 401:
7         redirect_to_login()
8     case 429:
9         wait_and_repeat()
```

# Pattern Matching

grape up

```
1 match response.status_code:
2     case 200:
3         success()
4     case 400:
5         bad_request()
6     case 401:
7         redirect_to_login()
8     case 429:
9         wait_and_repeat()
```

# Pattern Matching

grape up

```
1 match response.status_code:
2     case 200:
3         success()
4     case 400:
5         bad_request()
6     case 401:
7         redirect_to_login()
8     case 429:
9         wait_and_repeat()
```

# Gra tekstowa

grape up

# Gra tekstowa

grape up

```
1 command = input("What are you doing?")
2 commands = command.split()
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

# Gra tekstowa

grape up

```
1  command = input("What are you doing?")
2  commands = command.split()
3
4
5  match commands:
6
7
8
9
10
11
12
13
14
15
16
```

# Gra tekstowa

grape up

```
command = input("What are you doing?")
commands = command.split()


match commands:
    case ["wait"]:
```

# Gra tekstowa

grape up

```
command = input("What are you doing?")
commands = command.split()


match commands:
    case ["wait"]:
        wait()
```

# Gra tekstowa

grape up

```
1  command = input("What are you doing?")
2  commands = command.split()
3
4
5  match commands:
6      case ["wait"]:
7          wait()
8      case ["looking"]:
9          looking_around()
10
11
12
13
14
15
16
```

# Gra tekstowa

grape up

```
1  command = input("What are you doing?")
2  commands = command.split()
3
4
5  match commands:
6      case ["wait"]:
7          wait()
8      case ["looking"]:
9          looking_around()
10     case ["go"]:
11         go()
12
13
14
15
16
```

# Gra tekstowa

grape up

```
command = input("What are you doing?")
commands = command.split()


match commands:
    case ["wait"]:
        wait()
    case ["looking"]:
        looking_around()
    case ["go"]:
        go()
```

# Gra tekstowa

grape up

```
command = input("What are you doing?")
commands = command.split()


match commands:
    case ["wait"]:
        wait()
    case ["looking"]:
        looking_around()
    case ["go"]:
        go()
```

```
match commands:
    case ["go", …]:
```

grape up

```
match commands:
    case ["go", ("north" | "south" | "east" | "west")]:
        go()
```

```
match commands:
    case ["go", ("north" | "south" | "east" | "west") as direction]:
        go(direction)
```

```
match commands:
    case ["go", ("north" | "south" | "east" | "west") as direction]:
        go(direction)
    case ["go", _ ]:
        print("You can't go here")
```

```
1  match commands:
2      case ["go", ("north" | "south" | "east" | "west") as direction]:
3          go(direction)
4      case ["go", _ ]:
5          print("You can't go here")
6      case _:
7          print("You can't do it")
8
9
10
11
12
13
```

grape up

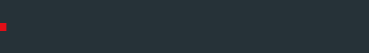
```
commands = [“go”, “north”]
```

```
1 match commands:
2     case ["go", ("north" | "south" | "east" | "west") as direction]:
3         go(direction)
4     case ["go", _ ]:
5         print("You can't go here")
6     case _:
7         print("You can't do it")
8
9
10
11
12
13
```

grape up

```
commands = [“go”, “back”]
```

```
1 match commands:
2     case ["go", ("north" | "south" | "east" | "west") as direction]:
3         go(direction)
4     case ["go", _ ]:
5         print("You can't go here")
6     case _:
7         print("You can't do it")
8
9
10
11
12
13
```



grape up

```
commands = [“look”]
```

```
1 match commands:
2     case ["go", ("north" | "south" | "east" | "west") as direction]:
3         go(direction)
4     case ["go", _ ]:
5         print("You can't go here")
6     case _:
7         print("You can't do it")
8
9
10
11
12
13
```

grape up

```
commands = [“go”]
```

```
1 match commands:
2     case ["go", ("north" | "south" | "east" | "west") as direction]:
3         go(direction)
4     case ["go", _ ]:
5         print("You can't go here")
6     case _:
7         print("You can't do it")
8
9
10
11
12
13
```

grape up

```
commands = [“go”]
```

```
1 match commands:
2     case ["go", ("north" | "south" | "east" | "west") as direction]:
3         go(direction)
4     case ["go", _ ]:
5         print("You can't go here")
6     case _:
7         print("You can't do it")
8
9
10
11
12
13
```

grape up

```
commands = [“go”, “north”, “west”]
```

```
match commands:
    case ["go", ("north" | "south" | "east" | "west") as direction]:
        go(direction)
    case ["go", _ ]:
        print("You can't go here")
    case _:
        print("You can't do it")
```

grape up

```
commands = [“go”, “north”, “west”]
```

```
1  match commands:
2      case ["go", ("north" | "south" | "east" | "west") as direction]:
3          go(direction)
4      case ["go", *directions]:
5          for direction in directions:
6              go(direction)
7      case ["go", _ ]:
8          print("You can't go here")
9      case _:
10         print("You can't do it")
11
12
13
```

grape up

```
commands = [“go”, “north”, “west”]
```

```
match commands:
    case ["go", ("north" | "south" | "east" | "west") as direction]:
        go(direction)
    case ["go", *directions]:
        for direction in directions:
            go(direction)
    case ["go", _ ]:
        print("You can't go here")
    case _:
        print("You can't do it")
```

grape up

```
commands = [“go”, “north”, “west”]
```

```
match commands:
    case ["go", ("north" | "south" | "east" | "west") as direction]:
        go(direction)
    case ["go", _ ]:
        print("You can't go here")
    case ["go", *directions]:
        for direction in directions:
            go(direction)
    case _:
        print("You can't do it")
```

```
1
2
3
4
5
6
```

```
1
2
3
4
5
6
7
8
9
```

grape up

```
player = {
    "health": 10,
    "stamina": 20,
    "items": ["knife"],
    "gold": 0,
}
```

```

```

grape up

```
player = {
    "health": 10,
    "stamina": 20,
    "items": ["knife"],
    "gold": 0,
}
```

```
match player:


```

grape up

```
player = {
    "health": 10,
    "stamina": 20,
    "items": ["knife"],
    "gold": 0,
}
```

```
match player:
   case {"health": 0}:
      print("You are dead")
```

grape up

```
player = {
    "health": 10,
    "stamina": 20,
    "items": ["knife"],
    "gold": 0,
}
```

```
match player:
    case {"health": 0}:
        print("You are dead")
    case {"stamina": stamina} if stamina < 5:
        print("You are exhausted")
```

grape up

```
player = {
    "health": 10,
    "stamina": 20,
    "items": ["knife"],
    "gold": 0,
}
```

```
match player:
    case {"health": 0}:
        print("You are dead")
    case {"stamina": stamina} if stamina < 5:
        print("You are exhausted")
    case {"items": items} if "knife" in items:
        print("You can fight")
```

grape up

```
player = {
    "health": 10,
    "stamina": 20,
    "items": ["knife"],
    "gold": 0,
}
```

```
match player:
    case {"health": 0}:
        print("You are dead")
    case {"stamina": stamina} if stamina < 5:
        print("You are exhausted")
    case {"items": items} if "knife" in items:
        print("You can fight")
    case {"health": health, **attrs}:
        do_something(health, attrs)
```

grape up

```
player = {
    "wounded": "arm",
}
```

grape up

```
match player:
    case {"wounded": _}:
        print("You need healing.")
```

```
1 match event:
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

grape up

```
match event:
    case PointEvent(position=(x,y)):
        handle_click_on(x,y)
```

grape up

```
match event:
    case PointEvent(position=(x,y)):
        handle_click_on(x,y)
    case KeyEvent(key_name="enter"):
        confirm()
```

grape up

```
match event:
    case PointEvent(position=(x,y)):
        handle_click_on(x,y)
    case KeyEvent(key_name="enter"):
        confirm()
    case KeyEvent(key_name="escape"):
        exit()
```

grape up

```
match event:
    case PointEvent(position=(x,y)):
        handle_click_on(x,y)
    case KeyEvent(key_name="enter"):
        confirm()
    case KeyEvent(key_name="escape"):
        exit()
    case KeyEvent():
        pass
```

grape up

# Czy to jest szybkie?

# Czy to jest szybkie?

grape up

```
1
2
3
4
5
6
7
8
9
10
11
12
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

# Czy to jest szybkie?

grape up

```python
def simple_if():
    status_code = 404
    if status_code == 200:
        return "ok"
    elif status_code == 400:
        return "bad_request"
    elif status_code == 404:
        return "not_found"
    else:
        return "unknown"

timeit.timeit(simple_if, number=1_000_000)
```

# Czy to jest szybkie?

grape up

```python
def simple_if():
    status_code = 404
    if status_code == 200:
        return "ok"
    elif status_code == 400:
        return "bad_request"
    elif status_code == 404:
        return "not_found"
    else:
        return "unknown"

timeit.timeit(simple_if, number=1_000_000)
```

```python
def simple_match():
    status_code = 404
    match status_code:
        case 200:
            return "ok"
        case 400:
            return "bad_request"
        case 404:
            return "not_found"
        case _
            return "unknown"

timeit.timeit(simple_match, number=1_000_000)
```

# Czy to jest szybkie?

grape up

```
1
2
3
4
5
6
7
8
9
10
11
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

# Czy to jest szybkie?

grape up

```
 1 def list_if():
 2     numbers = [1,1,2,4]
 3     if numbers[0] == 0:
 4         return "wrong list"
 5     if len(numbers) > 5:
 6         return "too big list"
 7     if sum(numbers) > 10:
 8         return "too big numbers"
 9     return "ok"
10
11 timeit.timeit(list_if,
number=1_000_000)
>>>
```

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
```

# Czy to jest szybkie?

grape up

```python
def list_if():
    numbers = [1,1,2,4]
    if numbers[0] == 0:
        return "wrong list"
    if len(numbers) > 5:
        return "too big list"
    if sum(numbers) > 10:
        return "too big numbers"
    return "ok"

timeit.timeit(list_if, number=1_000_000)
>>>
```

```python
def list_match():
    numbers = [1,1,2,4]
    match numbers:
        case [0, *_]:
            return "wrong list"
        case [*nums] if len(nums) > 5:
            return "too big list"
        case [*nums] if sum(nums) > 10:
            return "too big numbers"
        case _
            return "ok"

timeit.timeit(list_match, number=1_000_000)
>>>
```

# Czy to jest szybkie?

grape up

```
1
2
3
4
5
6
7
8
9
10
```

# Czy to jest szybkie?

grape up

```
 1 def class_if:
 2     event = KeyEvent(key="enter")
 3     if isinstance(event, PointEvent):
 4         return f"kliknięto w {event.x} {event.y}"
 5     if isinstance(event, KeyEvent) and event.key == "escape":
 6         return "wyłączono"
 7     if isinstance(event, KeyEvent) and event.key == "enter":
 8         return "potwierdzono"
 9     return "nie znam eventu"
10 timeit.timeit(class_if, number=1_000_000)
>>>
```

# Czy to jest szybkie?

grape up

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

# Czy to jest szybkie?

grape up

```
 1 def class_match:
 2     event = KeyEvent(key="enter")
 3     match event:
 4         case PointEvent(x=x, y=y):
 5             return f"kliknięto w {x} {y}"
 6         case KeyEvent(key="escape"):
 7             return "wyłączono"
 8         case KeyEvent(key="enter"):
 9             return "potwierdzono"
10         case _:
11             return "nie znam eventu"
12
13 timeit.timeit(class_match, number=1_000_000)
>>>
```

# Czy to jest szybkie?

grape up

```
1 player = {
2     "health": 10,
3     "stamina": 10,
4     "items": ["apple", "knife"],
5 }
```

# Czy to jest szybkie?

grape up

```
1
2
3
4
5
6
7
8
9
10
11
```

# Czy to jest szybkie?

grape up

```
 1 def dict_if():
 2     if player["health"] == 0:
 3         return "umarłeś"
 4     if player["stamina"] < 5:
 5         return "jesteś zmęczony"
 6     if player.get("wounded"):
 7         return "jesteś ranny"
 8     if "knife" in player["items"]:
 9         return "możesz walczyć"
10
11 timeit.timeit(dict_if, number=1_000_000)
>>>
```

# Czy to jest szybkie?

grape up

```
1
2
3
4
5
6
7
8
9
10
11
12
```

# Czy to jest szybkie?

grape up

```
 1 def dict_match():
 2     match player:
 3         case {"health": 0}:
 4             return "umarłeś"
 5         case {"stamina": stamina} if stamine < 5:
 6             return "jesteś zmęczony"
 7         case {"wounded": _}:
 8             return "jesteś ranny"
 9         case {"items": items} if "knife" in items:
10             return "możesz walczyć"
11
12 timeit.timeit(dict_match, number=1_000_000)
>>>
```

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | | | |
| Listy | | | |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484441598331556 | | |
| Listy | | | |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | |
| Listy | | | |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | | | |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | | |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | |

# Czy to jest szybkie?

grape up

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

# Czy to jest szybkie?

grape up

## Python 3.10

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

# Czy to jest szybkie?

grape up

## Python 3.10

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

## Python 3.11

| | If | Match | Różnica | Różnica do 3.10 |
|---|---|---|---|---|
| Proste inty | | | | |
| Listy | | | | |
| Klasy | | | | |
| Słowniki | | | | |

# Czy to jest szybkie?

grape up

## Python 3.10

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

## Python 3.11

| | If | Match | Różnica | Różnica do 3.10 |
|---|---|---|---|---|
| Proste inty | 0.039457625010982156 | 0.037669875018764 29 | +4.5% | +40% |
| Listy | | | | |
| Klasy | | | | |
| Słowniki | | | | |

# Czy to jest szybkie?

grape up

## Python 3.10

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

## Python 3.11

| | If | Match | Różnica | Różnica do 3.10 |
|---|---|---|---|---|
| Proste inty | 0.039457625010982156 | 0.037669875018764 29 | +4.5% | +40% |
| Listy | 0.127737500006333 | 0.066880042024422 44 | +48% | +31% |
| Klasy | | | | |
| Słowniki | | | | |

# Czy to jest szybkie?

grape up

## Python 3.10

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

## Python 3.11

| | If | Match | Różnica | Różnica do 3.10 |
|---|---|---|---|---|
| Proste inty | 0.039457625010982156 | 0.037669875018764 29 | +4.5% | +40% |
| Listy | 0.127737500006333 | 0.066880042024422 44 | +48% | +31% |
| Klasy | 0.21754362498177215 | 0.360631582967471 33 | -65% | +22% |
| Słowniki | | | | |

# Czy to jest szybkie?

grape up

## Python 3.10

| | If | Match | Różnica |
|---|---|---|---|
| Proste inty | 0.064484415983315 56 | 0.062942499993368 96 | +2% |
| Listy | 0.163971417001448 57 | 0.097484167024958 88 | +40% |
| Klasy | 0.3243987919995561 | 0.461802250007167 46 | -42% |
| Słowniki | 0.125791124999523 16 | 0.715146665985230 4 | -468% |

## Python 3.11

| | If | Match | Różnica | Różnica do 3.10 |
|---|---|---|---|---|
| Proste inty | 0.039457625010982156 | 0.037669875018764 29 | +4.5% | +40% |
| Listy | 0.127737500006333 | 0.066880042024422 44 | +48% | +31% |
| Klasy | 0.21754362498177215 | 0.360631582967471 33 | -65% | +22% |
| Słowniki | 0.091264790971763 43 | 0.463326750032138 1 | -407% | +35% |

# Czy to jest czytelne?

# Sqlalchemy

grape up

```
1 session.query(Article).all()
2 session.query(Article).filter(Article.is_active == True).all()
3 session.query(Article).filter(
4     or_(User.name == "Marcin", User.name == "Dawid"
5 ).all()
```

# Sqlalchemy

grape up

```
1 session.query(Article).all()
2 session.query(Article).filter(Article.is_active == True).all()
3 session.query(Article).filter(
4     or_(User.name == "Marcin", User.name == "Dawid"
5 ).all()
```

# Sqlalchemy

grape up

```
1 session.query(Article).all()
2 session.query(Article).filter(Article.is_active == True).all()
3 session.query(Article).filter(
4     or_(User.name == "Marcin", User.name == "Dawid"
5 ).all()
```

# Sqlalchemy

grape up

```
1 session.query(Article).all()
2 session.query(Article).filter(Article.is_active == True).all()
3 session.query(Article).filter(
4     or_(User.name == "Marcin", User.name == "Dawid"
5 ).all()
```

```python
class Operators(Enum):
    EQUALS = "="
    GREATER = ">"
    LIKE = "like"
    …
```

grape up

```python
class Operators(Enum):
    EQUALS = "="
    GREATER = ">"
    LIKE = "like"
    …

@dataclass
class CustomFilter:
    operator: Operators
    field: any
    value: any
```

grape up

```
1 def get_filter(query: Query, filter: CustomFilter):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {



    }
```

grape up

```
1  def get_filter(query: Query, filter: CustomFilter):
2      comparators = {
3          Operators.EQUALS:
4
5
6      }
7
8
9
10
11
12
13
14
15
16
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,


    }
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),

    }
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator =
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[]
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)


def get_users(filters):
    query = session.query(User)


```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)


def get_users(filters):
    query = session.query(User)
    for filter in filters:
        query = get_filter(query, filter)

```

grape up

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)


def get_users(filters):
    query = session.query(User)
    for filter in filters:
        query = get_filter(query, filter)
    return query.all()
```

```
1 filters = [
2
3
4 ]
5
6
```

grape up

```
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, True),

]


```

grape up

```
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, True),
    CustomFilter(Operators.EQUALS, User.is_banned, False),
]


```

grape up

```
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, True),
    CustomFilter(Operators.EQUALS, User.is_banned, False),
]

users = repository.get_users(filters)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)
```

```python
class Operators(Enum):
    EQUALS = "="
    GREATER = ">"
    LIKE = "like"
    …
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)
```

```python
class Operators(Enum):
    EQUALS = "="
    GREATER = ">"
    LIKE = "like"
    OR = "or"
    …
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda: filter.field == filter.value,
        Operators.LIKE: lambda: filter.field.like(filter.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }

    comparator = comparators[filter.operator]()
    return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:



    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [

        ]

    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
                                    for f in filter.value
        ]

    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]

    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return
    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter()
    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_())
    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    else:
        comparator = comparators[filter.operator]()
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    else:
        comparator = comparators[filter.operator](filter)
        return query.filter(comparator)
```

grape up

```python
# Sposób użycia
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, True),
    CustomFilter(Operators.EQUALS, User.is_banned, False),
]

users = repository.get_users(filters)
```

grape up

```python
# Sposób użycia
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, False),
    CustomFilter(Operators.EQUALS, User.is_banned, True),
]
or_filter = CustomFilter(Operators.OR, None, filters)

users = repository.get_users([or_filter])
```

grape up

```
# Sposób użycia
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, False),
    CustomFilter(Operators.EQUALS, User.is_banned, True),
]
or_filter = CustomFilter(operator=Operators.OR, value=filters)

users = repository.get_users([or_filter])
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    else:
        comparator = comparators[filter.operator](filter)
        return query.filter(comparator)
```

grape up

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_compara
    else:
        comparator = comparators[filter.opera
        return query.filter(comparator)
```

```python
@dataclass
class CustomFilter:
    operator: Operators
    field: any
    value: any
```

```
1 def get_filter(query: Query, filter: CustomFilter):
2
3
4
5
6
7
8
9
```

grape up

```
1 def get_filter(query: Query, filter: CustomFilter):
2     match filter.operator:
3
4
5
6
7
8
9
```

grape up

```
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
```

grape up

```
1 def get_filter(query: Query, filter: CustomFilter):
2     match filter.operator:
3         case Operators.EQUALS:
4             return query.filter(filter.field == filter.value)
5         case Operators.LIKE:
6             return query.filter(filter.field.like(filter.value)
7
8
9
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:

```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter.or_([get_filter(f) for f in filter.value])
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter.or_([get_filter(f) for f in filter.value])
```

```python
@dataclass
class CustomFilter:
    operator: Operators
    field: any
    value: any
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field =
        case Operators.LIKE:
            return query.filter(filter.field.l
        …
        case Operators.OR:
            return query.filter.or_([get_filte
```

grape up

```python
@dataclass
class CustomFilter:
    operator: Operators
    field: any
    value: any
```

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.fi
        case Operators.LIKE:
            return query.filter(filter.fi
        …
        case Operators.OR:
            return query.filter.or_([get_
```

grape up

```python
@dataclass
class CustomFilter:
    operator: Operators
    field: any
    value: any

@dataclass
class OrFilter:
    values: List[CustomFilter]
```

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter.or_([get_filter(f) for f in filter.value])
```

grape up

```python
def get_filter(query: Query, filter: Union[CustomFilter, OrFilter]):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter.or_([get_filter(f) for f in filter.value])
```

grape up®

```python
def get_filter(query: Query, filter: Union[CustomFilter, OrFilter]):
    match filter:
        case CustomFilter(operator=Operators.EQUALS):
            return query.filter(filter.field == filter.value)
        case CustomFilter(operator=Operators.LIKE):
            return query.filter(filter.field.like(filter.value)
        …
        case OrFilter():
            return query.filter.or_([get_filter(f) for f in filter.values])
```

grape up®

grape up

```python
class Filter(ABC):
```

```

```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError
```

```

```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
```

```

```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```

```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [


    ]
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]

class OrFilter(MultipleFilter):


```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]

class OrFilter(MultipleFilter):
  def prepare_query(self, query):

```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return self.field == self.value

class LikeFilter(SingleFilter):
  def get_filter(self):
    return self.field.like(self.value)
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]

class OrFilter(MultipleFilter):
  def prepare_query(self, query):
    return self.query.filter(or_(*self.get_filter()))
```

grape up

```
# Sposób użycia
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, False),
    CustomFilter(Operators.EQUALS, User.is_banned, True),
]
or_filter = CustomFilter(operator=Operators.OR, value=filters)

users = repository.get_users([or_filter])
```

grape up

```python
# Sposób użycia
filters = [
    EqualsFilter(User.is_active, False),
    EqualsFilter(User.is_banned, True),
]
or_filter = OrFilter(filters=filters)

users = repository.get_users([or_filter])
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    else:
        comparator = comparators[filter.operator](filter)
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter.or_([get_filter(f) for f in filter.value)
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return (self.field == self.value)

class LikeFilter(SingleFilter):
  def get_filter(self):
    return(self.field.like(self.value))
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]

class OrFilter(MultipleFilter):
  def prepare_query(self, query):
    return self.query.(or_(*self.get_filter()))
```

grape up

```
# Sposób użycia
filters = [
    CustomFilter(Operators.EQUALS, User.is_active, False),
    CustomFilter(Operators.EQUALS, User.is_banned, True),
]
or_filter = CustomFilter(operator=Operators.OR, value=filters)

users = repository.get_users([or_filter])
```

grape up

```
# Sposób użycia
filters = [
    CustomFilter(Operators.OR, value=[
        CustomFilter(Operators.AND, value=[
            CustomFilter(Operators.EQUALS, User.is_active, True),
            CustomFilter(Operators.LOWER, User.age, 18)
        ]),
        CustomFilter(Operators.AND, value=[
            CustomFilter(Operator.EQUALS, User.is_active, False),
            CustomFilter(Operator.GREATER, User.age, 65),
        ])
    ])
]
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    else:
        comparator = comparators[filter.operator](filter)
        return query.filter(comparator)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    elif filter.operator == Operators.AND:
        many_comparators = [
            comparators[f.operator](f) for f in filter.value
        ]
        return query.filter(and_(*many_comparators))
    else:
        comparator = comparators[filter.operator](filter)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    comparators = {
        Operators.EQUALS: lambda f: f.field == f.value,
        Operators.LIKE: lambda f: f.field.like(f.value),
        …
    }
    if filter.operator == Operators.OR:
        many_comparators = [
            get_filter(f) for f in filter.value
        ]
        return query.filter(or_(*many_comparators))
    elif filter.operator == Operators.AND:
        many_comparators = [
            get_filter(f) for f in filter.value
        ]
        return query.filter(and_(*many_comparators))
    else:
        comparator = comparators[filter.operator](filter)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter.or_([get_filter(f) for f in filter.value)
```

grape up

```python
def get_filter(query: Query, filter: CustomFilter):
    match filter.operator:
        case Operators.EQUALS:
            return query.filter(filter.field == filter.value)
        case Operators.LIKE:
            return query.filter(filter.field.like(filter.value)
        …
        case Operators.OR:
            return query.filter(or_([get_filter(f) for f in filter.value])
        case Operators.AND:
            return query.filter(and_([get_filter(f) for f in filter.value])
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return (self.field == self.value)

class LikeFilter(SingleFilter):
  def get_filter(self):
    return(self.field.like(self.value))
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]

class OrFilter(MultipleFilter):
  def prepare_query(self, query):
    return self.query.filter(
      or_(*self.get_filter())
    )
```

grape up

```python
class Filter(ABC):
  def get_filter(self):
    raise NotImplementedError

  def prepare_query(self, query):
    raise NotImplementedError

@dataclass
class SingleFilter(Filter):
  field: any
  value: any

  def prepare_query(self, query):
    return query.filter(self.get_filter())


class EqualsFilter(SingleFilter):
  def get_filter(self):
    return (self.field == self.value)

class LikeFilter(SingleFilter):
  def get_filter(self):
    return(self.field.like(self.value))
```

```python
@dataclass
class MultipleFilter(Filter):
  filters: List[Filter]

  def get_filter(self):
    return [
      filter.get_filter()
      for filter in self.filters
    ]

class OrFilter(MultipleFilter):
  def prepare_query(self, query):
    return self.query.filter(
      or_(*self.get_filter())
    )

class AndFilter(MultipleFilter):
  def prepare_query(self, query):
    return self.query.filter(
      or_(*self.get_filter())
    )
```

grape up

grape up

# Czy używać Pattern Matching?

| Plusy | Minusy |
| --- | --- |
| | |
| | |
| | |
| | |

| Plusy | Minusy |
| --- | --- |
| Może być szybszy w pisaniu | |
| | |
| | |
| | |

| Plusy | Minusy |
| --- | --- |
| Może być szybszy w pisaniu | |
| Może być czytelniejszy przy wielu przypadkach | |
| | |
| | |

| **Plusy** | **Minusy** |
| --- | --- |
| Może być szybszy w pisaniu | Czasami może być mniej czytelny przy skomplikowanym przypadku |
| Może być czytelniejszy przy wielu przypadkach | |
| | |
| | |

```python
if user["is_active"] is True and user["age"] < 18:
    foo()
```

grape up

```python
if user["is_active"] is True and user["age"] < 18:
    foo()

match user:
    case {"is_active": True, age: age} if age < 18:
        foo()
```

grape up

```python
if user["is_active"] is True and user["age"] < 18:
    foo()

match user:
    case {"is_active": True, age: age} if age < 18:
        foo()
```

grape up

```python
if is_legal_age(user):
    foo()

match user:
    case {"is_active": True, age: age} if age < 18:
        foo()
```

grape up

| **Plusy** | **Minusy** |
| --- | --- |
| Może być szybszy w pisaniu | Czasami może być mniej czytelny przy skomplikowanym przypadku |
| Może być czytelniejszy przy wielu przypadkach | |
| | |
| | |

| Plusy | Minusy |
| --- | --- |
| Może być szybszy w pisaniu | Czasami może być mniej czytelny przy skomplikowanym przypadku |
| Może być czytelniejszy przy wielu przypadkach | |
| Lepiej sobie radzi kiedy zmienna może być różnych typów | |
| | |

| Plusy | Minusy |
| --- | --- |
| Może być szybszy w pisaniu | Czasami może być mniej czytelny przy skomplikowanym przypadku |
| Może być czytelniejszy przy wielu przypadkach | Może być wolniejszy |
| Lepiej sobie radzi kiedy zmienna może być różnych typów | |
| | |

grape up

# Dziękuję :)