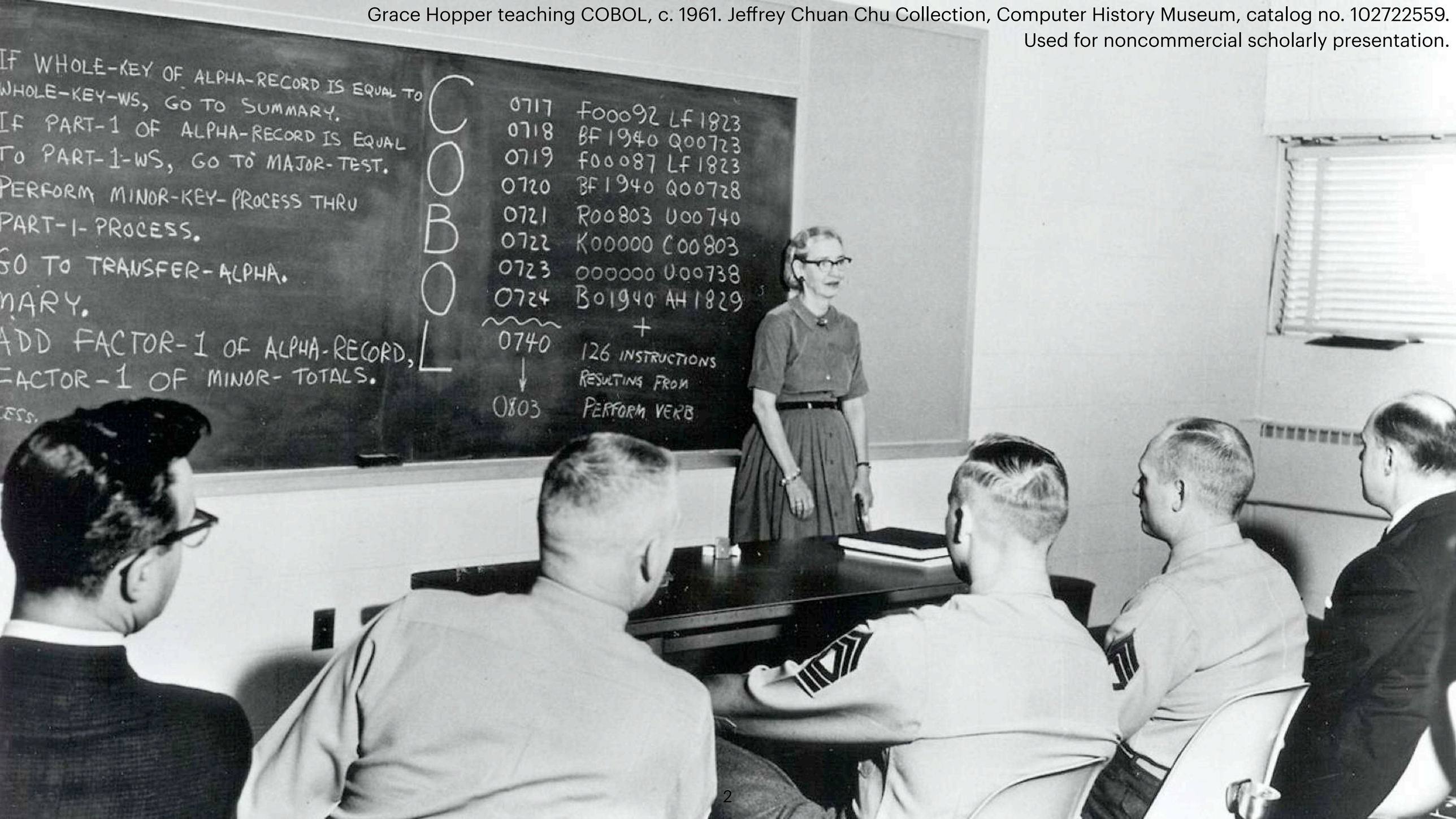
Morphosyntactic Programming

Case, Mood, and Type-Directed Disambiguation for Turkish-Like Syntax

Joomy Korkut, Bloomberg
Alperen Keles, University of Maryland, College Park
Onur Akdemir, Topsort

New Jersey Programming Languages and Systems Seminar
May 22, 2026



Grace Hopper teaching COBOL, c. 1961. Jeffrey Chuan Chu Collection, Computer History Museum, catalog no. 102722559.
Used for noncommercial scholarly presentation.

```
247 ПРОЦ ВПЕРЕД();
248  ЕСЛИ ПЕРЕДСОСТАВОМ()="ТУПИК" ТО
249     ВЫВОД:"??НЕ МОГУ, НЕКУДА?";
250     ВЫХОД
251  ИНАЧЕ
252     <**>->ВОЗВСТР;
253     ОКНОДАЙ(КАРТ);
254     ШАГСОСТАВА();
255     ПОКА ПЕРЕДСОСТАВОМ()="СВОБОДНО"::
256        ШАГСОСТАВА()
257     ВСЕ;
258     ДЛЯ И ИЗ ВОЗВСТР::
259        ПУТИ[И,9]->ПУТИ[И,8];
260     ВСЕ;
261     ДЛЯ И ОТ 2 ДО #ПУТИ::
262        ЕСЛИ ПУТИ[И,1]="СТРЕЛКА" ТО
263           КРАСЬСТРЕЛК(И)
264        ИНАЧЕ
265           ПОЗ(0,ПУТИ[И,5]*4);
266           ВЫВОД БПС:"?",ФТЕКСТ(3*ПУТИ[И,4]-1,"="),"<"
267        ВСЕ
268     ВСЕ;
269     ДЛЯ И ОТ 1 ДО #ВАГОНЫ::
270        РИСВАГ(И,0)
271     ВСЕ;
272  ВСЕ;
273  ОКНОДАЙ(ДИАЛ);
274 КНЦ;
275
276 ФУНК ПЕРЕДСОСТАВОМ();
277 ИМЕНА:ПУТЬ,МЕСТО,Р;
278 _ВАГОНЫ[СОСТАВ[1],4]->ПУТЬ;
```

Rapira (1982)

```
قلب: لغة برمجة - مترجم ٣،١،٠
رمزي ناصر ٢٠١٢

أمثلة - النجدة - ما هذا؟

<<< (أمثلة)
سهل: مرحبا يا عالم
متوسط: عدد فيبوناتشي
متقدم: لعبة الحياة لكونواي
<<< (حدد فيبوناتشي (لامدا (ن)
...     (إذا (أصغر؟ ن ٢)
...         ن
...     (جمع (فيبوناتشي (طرح ن ١))
...         (فيبوناتشي (طرح ن ٢)))))
...     (قول (فيبوناتشي ١٠))
                                ٥٥
                            ==> ٥٥
                            <<<
```

Qalb (2013)

Give him the book.

Case	Ending	Example		Translation	
Nominative	-∅-	*ev* ("house")	*adam* ("man")	"(the) house"	"(the) man"
Accusative	*-(y)ı-, -(y)i-, -(y)u-, -(y)ü-*	*evi*	*adamı*	"the house"	"the man"
Dative	*-(y)a-, -(y)e-*	*eve*	*adama*	"to the house"	"to the man"
Locative	*-da-, -de-, -ta-, -te-*	*evde*	*adamda*	"at home"	"in/on the man"
Ablative	*-dan-, -den-, -tan-, -ten-*	*evden*	*adamdan*	"from the house"	"from the man"
Genitive	*-(n)ın-, -(n)in-, -(n)un-, -(n)ün-*	*evin*	*adamın*	"the house's"	"the man's"
Instrumental	*-(y)le-, -(y)la-*	*evle*	*adamla*	"with the house"	"with the man"

difference of 5 and 3

difference of 5 and 3

5’le 3’ün farkı

difference of 5 and 3

instrumental case

genitive case

nominative case with possessive suffix

difference of 5 and 3

genitive case

instrumental case

nominative case with possessive suffix


Kip


```
Kip> 5'le 3'ün farkı
2

Kip> 3'ün 5'le farkı
2
```

```
# let difference ~x ~y = x - y;;
val difference : x:int -> y:int -> int = <fun>

# difference ~x:(printf "x"; 5) ~y:(printf "y"; 3) ;;
yx- : int = 2

# difference ~y:(printf "y"; 3) ~x:(printf "x"; 5) ;;
yx- : int = 2

# (difference ~y:(printf "y"; 3)) ~x:(printf "x"; 5);;
xy- : int = 2
```

Currying changes the order of side effects!

Let’s make side effects
explicit and **isolated** to avoid this.

```
selamlamak bitimi,
  isim için okuyup,
  ("Merhaba "yla ismin birleşimini) yazmaktır.
```

Effectful functions are infinitive verbs!

Binds of effectful functions are sequential converbs!

```
selamlamak bitimi,
  isim için okuyup,
  ("Merhaba "yla ismin birleşimini) yazmaktır.
```

```
greet :: IO ()
greet = do
  name ← getLine
  putStrLn ("Hello " ++ name)
```

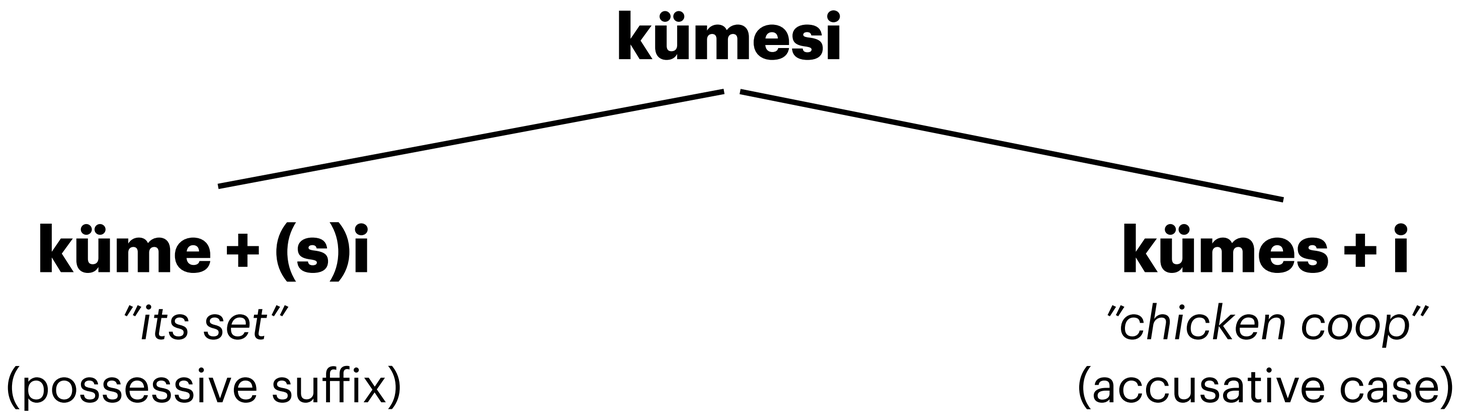
Effectful functions are infinitive verbs!

Binds of effectful functions are sequential converbs!

```
selamlamak bitimi,
  isim için okuyup,
  ("Merhaba "yla ismin birleşimini) yazmaktır.

selamla.
```

Invoke effectful functions with the imperative mood!

Ambiguity Is Deferred to Elaboration



Flexible Application Order Everywhere

κ	::=	:NOM | :ACC | :DAT | :LOC | :ABL | :GEN | :INS | :COND | :P3S	*grammatical case labels*
κ_r	::=	:NOM | :P3S	*return-case labels*
κ_{res}	::=	$\varnothing \mid \kappa_r$	*result-case marker*
μ	::=	pure | eff	*effect modes*
c	::=	$n \mid fl \mid s \mid ch$	*integer, float, string, and character literals*
τ	::=	$\alpha \mid$ Int | Float | String | Char $\mid D\{\tau_1\kappa_1, \ldots, \tau_m\kappa_m\} \mid \tau_1 \to \tau_2$	*types*
υ	::=	(τ, κ_{res})	*result types*
σ	::=	$\forall\overline{\alpha}.\ [\tau_1\kappa_1, \ldots, \tau_n\kappa_n] \Rightarrow \tau_r\kappa_r$	*signature*
p	::=	$\_ \mid x \mid c \mid C\{p_1\kappa_1, \ldots, p_n\kappa_n\}$	*patterns*
e	::=	$x \mid c \mid (e : \tau)$	*variables, literals, ascription*
	|	$f\{e_1\kappa_1, \ldots, e_n\kappa_n\} \mid C\{e_1\kappa_1, \ldots, e_n\kappa_n\}$	*case-labeled application*
	|	match e with $p_1 \Rightarrow e_1 \mid \cdots \mid p_k \Rightarrow e_k$	*match*
	|	let $x = e_1$ in e_2	*pure local binding*
	|	$e_1; e_2 \mid x \leftarrow e_1; e_2$	*effect sequencing/binding*
cl	::=	$f\{p_1\kappa_1, \ldots, p_n\kappa_n\} = e$	*function clause*
ϕ	::=	$[cl_1, \ldots, cl_m]$	*function definition family*
δ	::=	data $D\{\alpha_1\kappa_1, \ldots, \alpha_m\kappa_m\}$ where $C_1 : \sigma_1 \mid \cdots \mid C_k : \sigma_k$	*ADT declaration*
P	::=	$\delta_1; \cdots; \delta_m; \phi_1; \cdots; \phi_\ell; e$	*programs*

Fig. 1. Syntax of the core calculus.

Head-Final Syntax is like Reverse Polish Notation

difference of 5 and 3

5’le 3’ün farkı

Head-Final Syntax is like Reverse Polish Notation

the product of the difference of 5 and 3 and the difference of 4 and 1

(5’le 3’ün farkıyla) (4’le 1’in farkının) çarpımı

Head-Final Syntax is like Reverse Polish Notation*

the product of the difference of 5 and 3 and the difference of 4 and 1

5’le 3’ün farkıyla 4’le 1’in farkının çarpımı

* if you don’t take into account function overloading with different arities.

On the foolishness of "natural language programming".

Since the early days of automatic computing we have had people that have felt it as a shortcoming that programming required the care and accuracy that is characteristic for the use of any formal symbolism. They blamed the mechanical slave for its strict obedience with which it carried out its given instructions, even if a moment's thought would have revealed that those instructions contained an obvious mistake. "But a moment is a long time, and thought is a painful process." (A.E.Housman). They eagerly hoped and waited for more sensible machinery that would refuse to embark on such nonsensical activities as a trivial clerical error evoked at the time.

Playground

Write custom Kip code, or pick an existing example, and run it.

Example:

Greeting

Run | Stop | Generate JavaScript | Fullscreen

Source

```
selamlamak,
  isim için okuyup,
  ("Merhaba "yla ismin birleşimini) yazmaktır.

selamla.
```

Output

Playground

Write custom Kip code, or pick an existing example, and run it.

Example:

Greeting

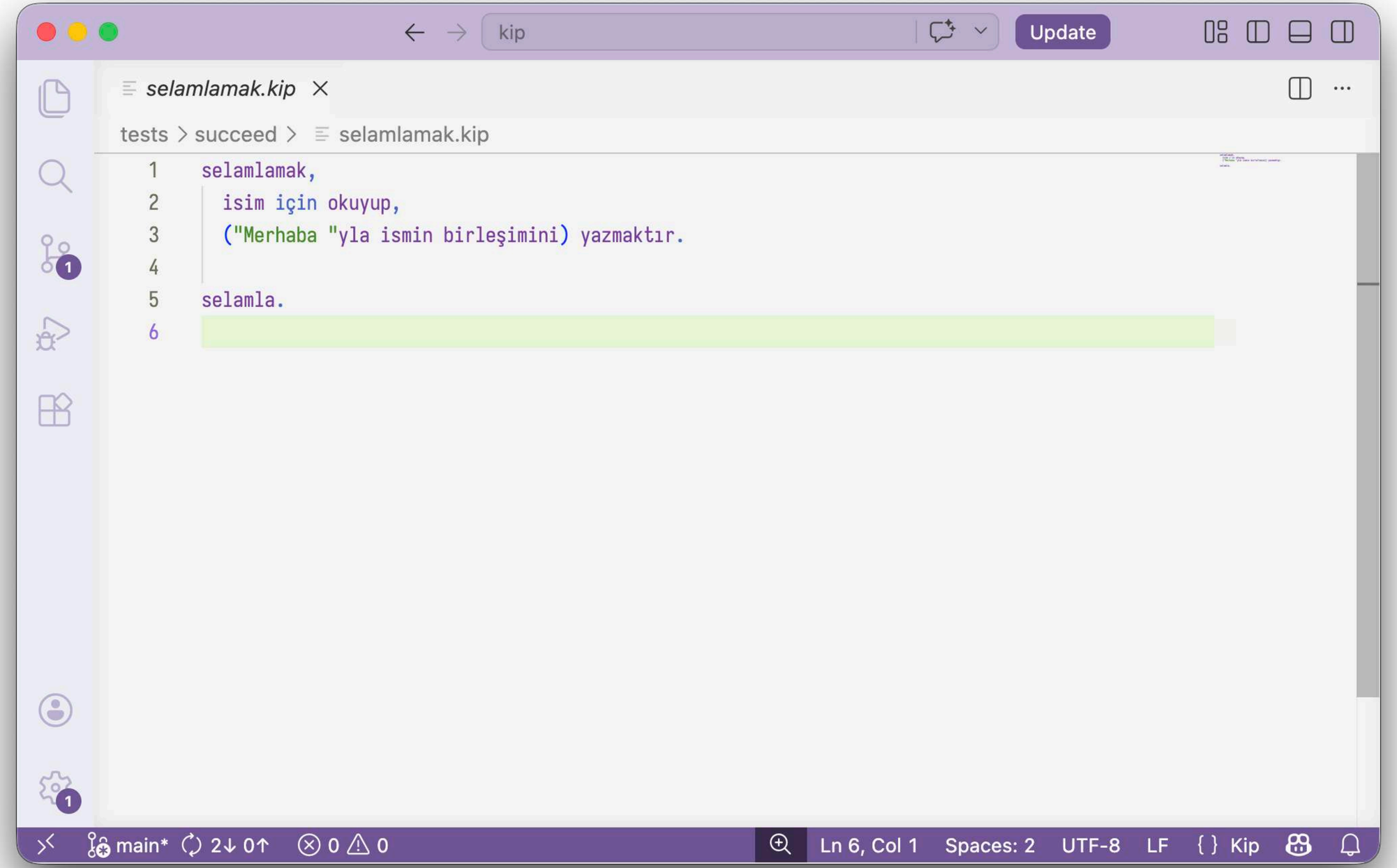
Run | Stop | Generate JavaScript | Fullscreen

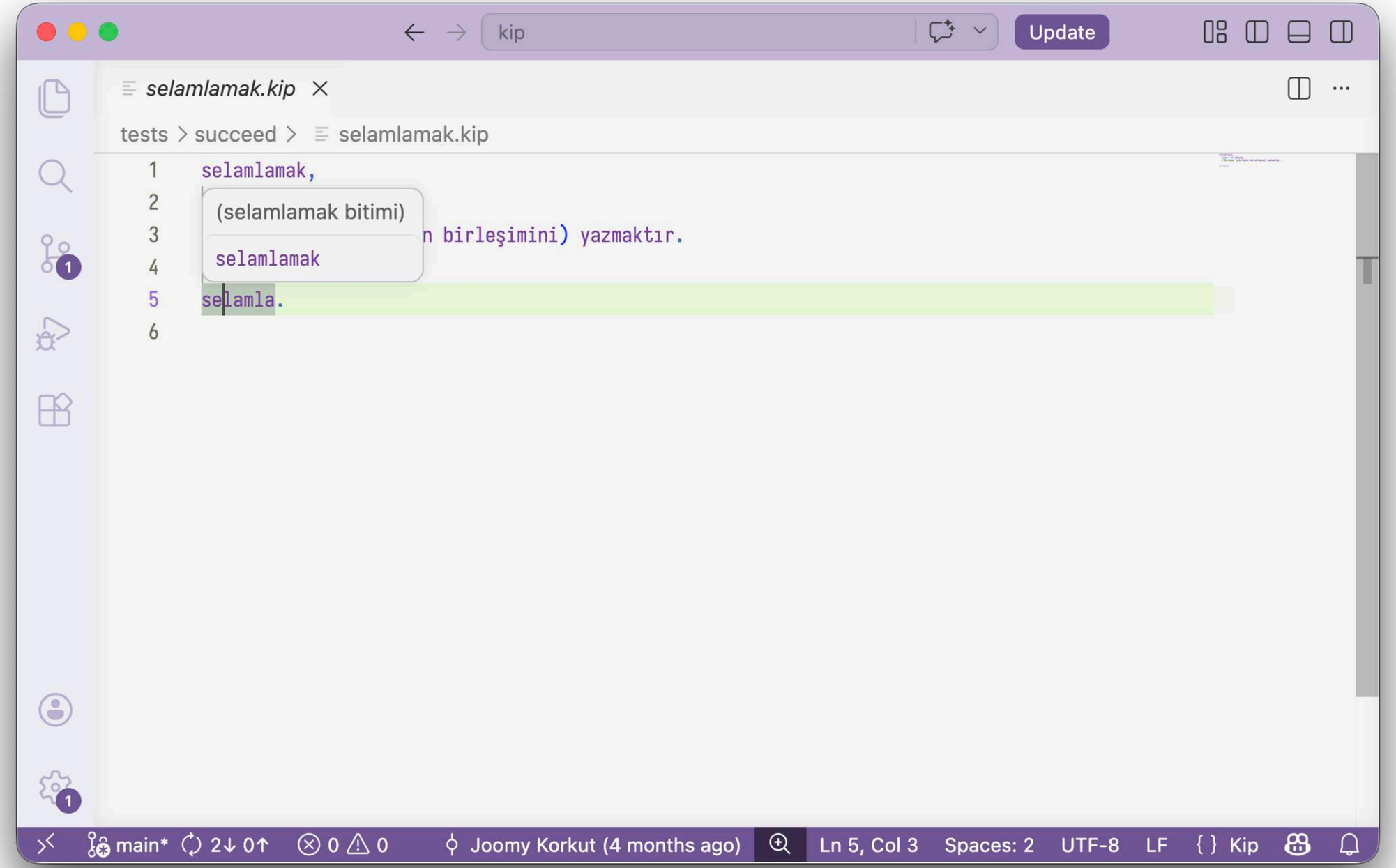
Source

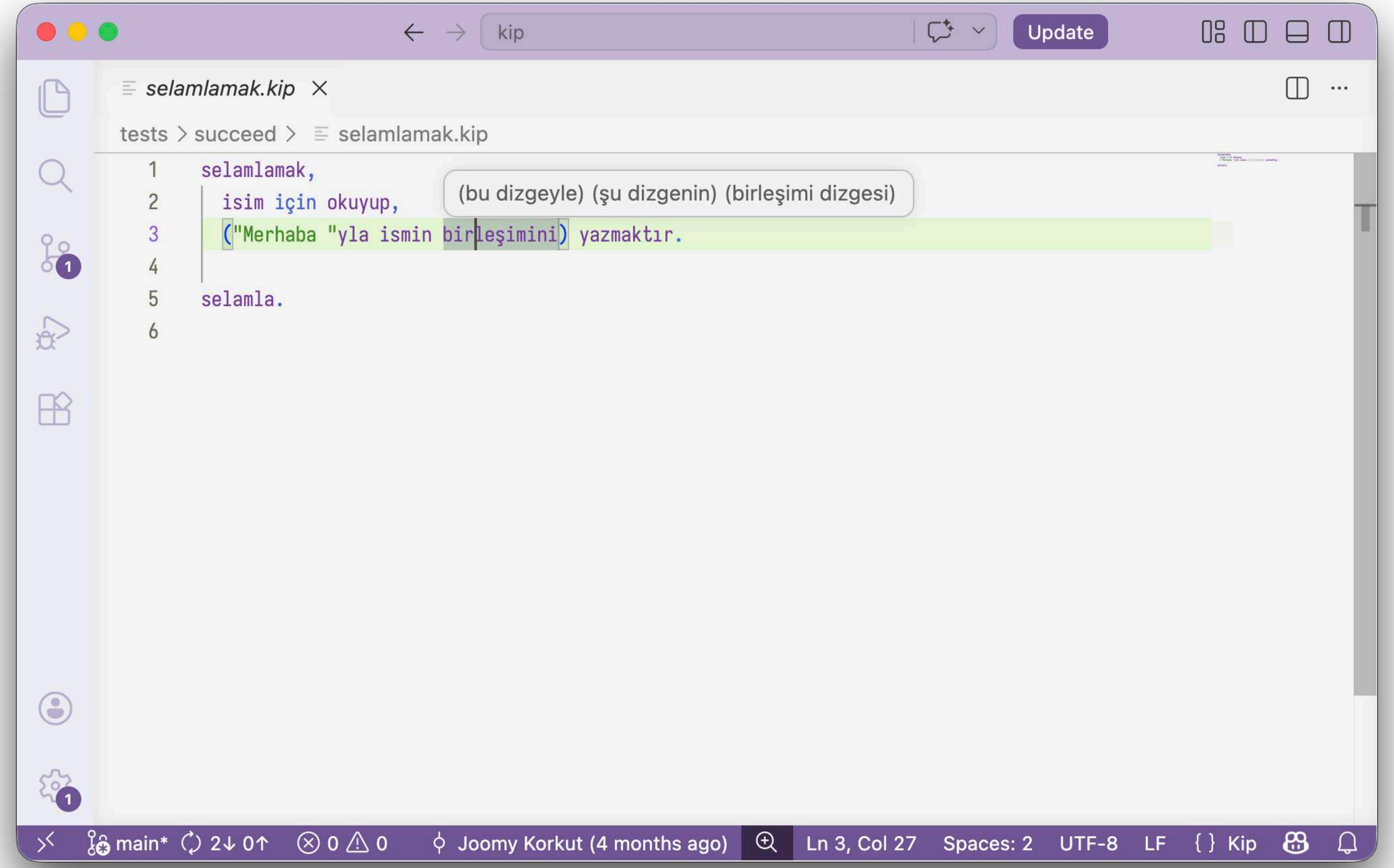
```
selamlamak,
  isim için okuyup,
  ("Merhaba "yla ismin birleşimini) yazmaktır.

selamla.
```

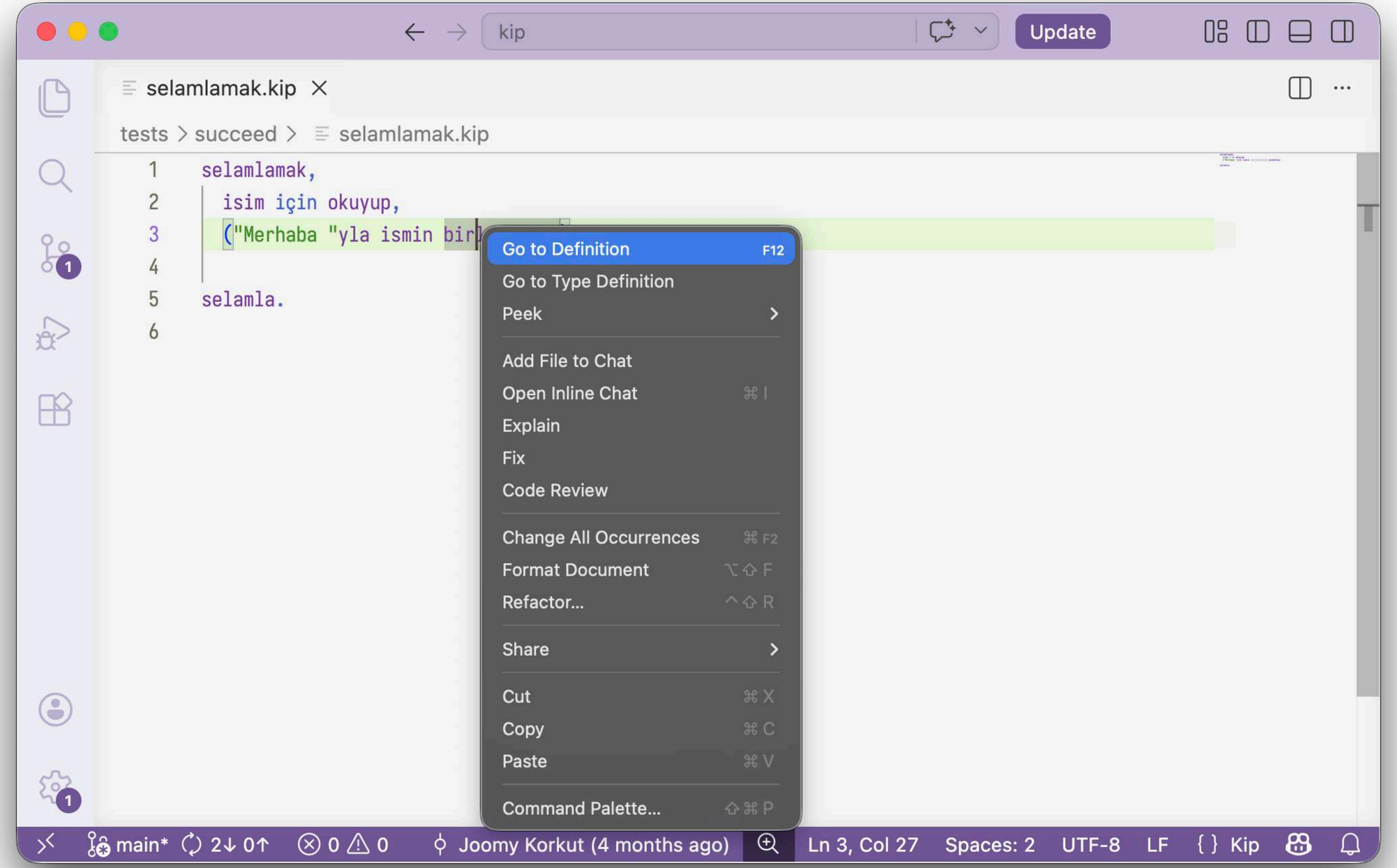
Output

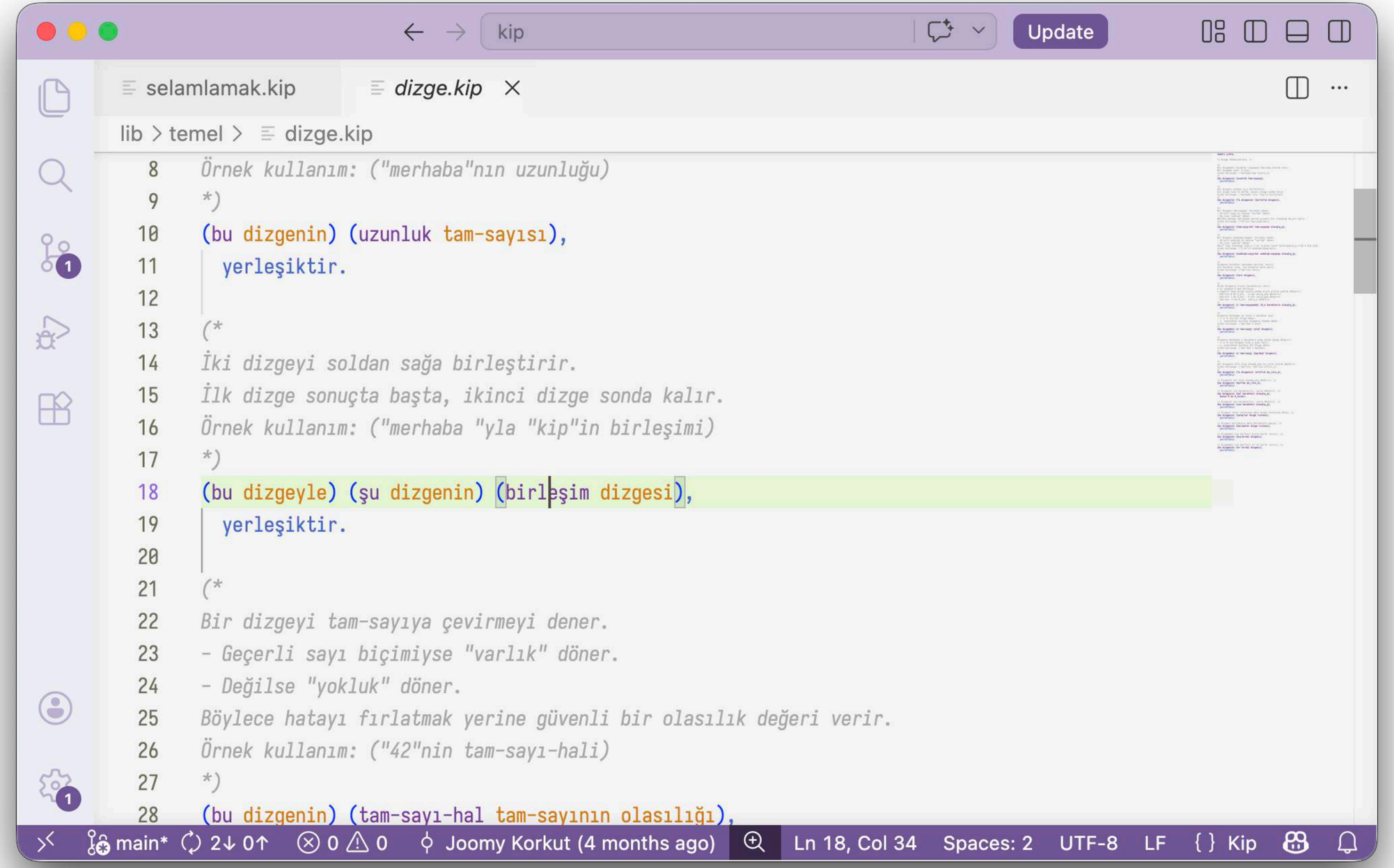
```
› Joomy
Merhaba Joomy
```


kip
Update
selamlamak.kip
tests > succeed > selamlamak.kip
1 selamlamak,
2 isim için okuyup,
3 ("Merhaba "yla ismin birleşimini) yazmaktır.
4
5 selamla.
6
main* 2↓ 0↑ 0 0
Ln 6, Col 1 Spaces: 2 UTF-8 LF { } Kip


kip
Update
selamlamak.kip
tests > succeed > selamlamak.kip
1 selamlamak,
2
3 n birleşimini) yazmaktır.
4
5 selamla.
6
(selamlamak bitimi)
selamlamak
main* 2↓ 0↑ 0 0 Joomy Korkut (4 months ago) Ln 5, Col 3 Spaces: 2 UTF-8 LF { } Kip


kip
Update
selamlamak.kip
tests > succeed > selamlamak.kip
1 selamlamak,
2 isim için okuyup,
3 ("Merhaba "yla ismin birleşimini) yazmaktır.
4
5 selamla.
6
(bu dizgeyle) (şu dizgenin) (birleşimi dizgesi)
main* 2↓ 0↑ 0 0 Joomy Korkut (4 months ago) Ln 3, Col 27 Spaces: 2 UTF-8 LF { } Kip


kip
Update
selamlamak.kip
tests > succeed > selamlamak.kip
1 selamlamak,
2 isim için okuyup,
3 ("Merhaba "yla ismin bir
4
5 selamla.
6
Go to Definition F12
Go to Type Definition
Peek
Add File to Chat
Open Inline Chat ⌘ I
Explain
Fix
Code Review
Change All Occurrences ⌘ F2
Format Document ⌥ ⇧ F
Refactor... ^ ⇧ R
Share
Cut ⌘ X
Copy ⌘ C
Paste ⌘ V
Command Palette... ⇧ ⌘ P
main* 2↓ 0↑ 0 0 Joomy Korkut (4 months ago) Ln 3, Col 27 Spaces: 2 UTF-8 LF { } Kip


kip
Update
selamlamak.kip
dizge.kip
lib > temel > dizge.kip
8 Örnek kullanım: ("merhaba"nın uzunluğu)
9 *)
10 (bu dizgenin) (uzunluk tam-sayısı),
11 yerleşiktir.
12
13 (*
14 İki dizgeyi soldan sağa birleştirir.
15 İlk dizge sonuçta başta, ikinci dizge sonda kalır.
16 Örnek kullanım: ("merhaba "yla "kip"in birleşimi)
17 *)
18 (bu dizgeyle) (şu dizgenin) (birleşim dizgesi),
19 yerleşiktir.
20
21 (*
22 Bir dizgeyi tam-sayıya çevirmeyi dener.
23 - Geçerli sayı biçimiyse "varlık" döner.
24 - Değilse "yokluk" döner.
25 Böylece hatayı fırlatmak yerine güvenli bir olasılık değeri verir.
26 Örnek kullanım: ("42"nin tam-sayı-hali)
27 *)
28 (bu dizgenin) (tam-sayı-hal tam-sayının olasılığı),
main* 2↓ 0↑
0 0
Joomy Korkut (4 months ago)
Ln 18, Col 34
Spaces: 2
UTF-8
LF
{} Kip

https://github.com/kip-dili/kip

https://kip-dili.github.io