Emacs support for the Go Programming Language

Support Fig. Conscript forward in the perspective personal process of the construction of the constructio	Description	Keystroke	Function	Note		
Comparison to your wife age mandle well probable Support	Go	Support for the Go programm	iing language is described in this pag			
Figure 1	programming	Go support requires the go-mode external package.				
** Control of the time activity of the control of t		Files with the .go extension	ons are recognized as Go source files			
- Control of the case with fixed as parties, we the pairs got but with fixed aspired but the case of the copy of the case of	Саррон		71 /			
segament of 4229 4729 from read a part visiting a los suspens consists. **Country for years underly up this eff improvision production of the production of		Generic programming language features like template text insertion handle Go comment style. See ∑ Inserting Text .,				
All process requires accord to the for congressing larguagest process of part control of the confidence of the confide		sequence or <f12> <f2> from inside a buffer visiting a Go source code file.</f2></f12>				
See also E_filled See also E_filed See also		Support for syntax checking with either flymake or flycheck via the goflymake Go program.				
See since 2 Flago Into Into Into Into Into Into Into Into Into						
Exclamation Class of 10 Set 0 423 Set 0 425	Open this PDF file.	<f11> SPC g <f1></f1></f11>		Open the <u>\$1 - Go</u> local PDF. If the prefix argument (like C-u or M) is used, then it opens		
Security	See also: <u>∑ Help/</u> Info	<f12> <f1></f1></f12>	WEB-PAGE)			
Extensive (412) < (42) pel-customize-library soptional OTER WINDOW) Class apport 1. Class	<u>∑ Customize</u> PEL	<f11> SPC g <f2></f2></f11>	(pel-customize-pel &optional			
Set lab width for current buffer Set 12 × 613	Go support	<f12> <f2></f2></f12>	OTHER-WINDOW)	If OTHER-WINDOW is non-nil (use C-u), display in another window.		
CELL 15 SEC 9 N-1 (Fill 5 SEC 9	<u> ∑ Customize</u>	<f11> SPC g <f3></f3></f11>				
The change is temporary and affects the current buffer only, To change he to see with used or pile of source or other file only. To change he the see with used or pile of source or other file, or annea the "pil-go-lag-width" Toggle goffert run Cf12	Emacs Go support	<f12> <f3></f3></f12>	OTHER-WINDOW)	If OTHER-WINDOW is non-nil (use C-u), display in another window.		
The change is temporary and effects the current buffer only, 10 barry for the seventh used or 10 be source on de files, change the 'pel-go-lag-width' 10 be of the work that of the the work that of the seventh used or 10 be source on de files, change the 'pel-go-lag-width' 10 be of the work that of the seventh used or 10 be source or de files, change the 'pel-go-lag-width' 10 be of the work that of the seventh used or 10 be source or de files, change the 'pel-go-lag-width' 10 be of the work that of the seventh wild so that per file. After the last of imports a least that the series of the seventh wild seventh the seventh wild seventh wild seventh the seventh wild seventh the seventh wild seventh wild seventh the seventh wild seventh the seventh wild seventh the seventh seventh wild seventh the seventh seventh the seventh seventh the seventh	Set tab width for	<f11> SPC q M-t</f11>	(pel-go-set-tab-width N)	Change the tab width used in current buffer.		
Toggle goffmir run on file save cf.13 > RPC g M-s pel-go-setup-info) Cf.2 > K-s per-go-setup-info) Deplay Go setup information: - tab width - whether gloriful sexecuted before saving buffer. Add one will might be severed by the sexual point in sexu	current buffer		,	The change is temporary and affects the current buffer only.		
Add one import products to the company processor to the company to				user-option variable instead.		
** whether goffind is executed before saving buffer. **Add new ImPORT to the list of imports. Don't move point. **Comports assterned** **Comports a	Toggle gofmt run on file save		(pel-go-setup-info)	' ' '		
When called with a pretix ARG also for an attentative name to import the pockage as.		<f12> M-s</f12>				
If no list exists yet, one will be created if possible.	Add new import	C-c C-a	(go-import-add ARG IMPORT)			
Describe expression at point. C=c C=d (godef-describe POINT) Describe the expression of DOINT. This uses the agents associately, a Go program. To install it, run the following command from a shell: go get github.com/ compense/code the expression defination. Move to expression C=c C=f (godef-jump-POINT &optional OTHER-WINDOW) Affect to expression a chemical from a shell: go get github.com/ point and the defination of the expression and point. Affect to expression a chemical from a shell: go get github.com/ point and the defination of the expression at POINT or the defination of the expression at POINT or the other window. - after that command, use k→, to go back to original point. - Affect the command, use k→, to go back to original point. - Affect the command, use k→, to go back to original point. - Affect the command, use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that command use k→, to go back to original point. - Affect that that command use k→, to go back to original point.	module package			If no list exists yet, one will be created if possible.		
This uses the golf executable. a Co program.	import statement					
Move to expression comment from a shell: go get github.com/ repreper/gotone C=c C-j (godef-jump-POINT apptional definition of THER-KINDOW) THER-KINDOW) THER-KINDOW) Move to expression comment variable must be setup and GOPATH/bin must be in the PATH to be able to run goder definition of the expression of POINT of THER-KINDOW) Move to expression of C=c C-j (godef-jump-other-window POINT) Move to expression of C=c C-f a (go-get-arguments &optional ARG) C=c C-f a (go-get-arguments &optional ARG) C=c C-f d (go-get-arguments &optional ARG) C=c C-f d (go-get-arguments &optional ARG) C=c C-f d (go-get-do-docstring &optional ARG) C=c C-f d (go-get-do-docstring &optional ARG) C=c C-f d (go-get-docstring &optional ARG) C=c C-f d (go-get-function &optional ARG) C=c C-f d (go-get-function &optional ARG) C=c C-f f (go-get-functi	Describe	C-c C-d	(godef-describe POINT)			
Move to expression definition C-c C-j (godef-jump-POINT & optional DIHER-WINDOW) (godef-jump-POINT & optional DIHER-WINDOW) (godef-jump-other-window POINT) (god	expression at point.					
Move to expression C-c C-j Godef-jump POINT & optional OTHER-WINDOW) after that command, use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original point. after that command use M-, to go back to original po				rogpeppe/godef.		
Affection of the procession of Policy Godef-jump-other-window Folicy Godef-jump-other-window Affection in other window C-z 4 C-c C-j Godef-jump-other-window Jump to the definition of the expression at POINT but into other window Affection in other window Affection in other window Affection Go-goto-decesting & optional Affection Affectio						
Move to expression definition in other window C = 2 4 C = C - 3 (go-goto-arguments &optional ARG) Jump to the definition of the expression at POINT but into the other window. • after that command, use № −, to go back to original point. Move to current function arguments C = C - E a (go-goto-docstring &optional ARG) Go to the arguments of the current function. • If ARG is non-nill, anonymous functions are skipped. Move to current function C - C - F d (go-goto-docstring &optional ARG) Go to the top of the docstring at the current function. • If there is none, add one beginning with the name of the current function. • Anonymous functions on not have docstrings, so when this is called interactively anonymous functions on not have docstring so when this is called interactively anonymous functions on not have docstring so when this is called interactively anonymous functions on not have docstring down. • If we are on a docstring, follow the docstring down. • If we are on a docstring, follow the docstring down. • If we are on a docstring, follow the docstring down. • If we are on a docstring, follow that the text at the top of a file and search forward instead. • If we are on a docstring, follow that the text are the top of a file and search forward instead. • If we are on a docstring, follow that the text is explored. • If we are on a docstring, follow that the text are the top of a file and search forward instead. • If we are on a docstring, follow that the text is explored. • If we are on a docstring, follow that the text is explored. • If we are on a docstring, follow that the text is explored. • If we are on a docstring, foll	Move to expression definition	C-c C-j		· •		
Window Move to current function arguments C=C C=f a (go-goto-arguments &optional ARG) Go to the arguments of the current function. If ARG is non-nil, anonymous functions are skipped. Move to current function docstring C=C C=f d (go-goto-docstring &optional ARG) Go to the top of the docstring of the current function. If there is none, add one beginning with the name of the current function. Move to function definition C=C C=f f (go-goto-function &optional ARG) Go to the top of the docstring, follow the docstring, so when this is called interactively anonymous functions do not have docstrings, so when this is called interactively anonymous functions do not have docstring, so when this is called interactively anonymous functions do not have docstring from the current function. Move to function C=C C=f f (go-goto-function &optional ARG) Go to the trunction falled interactively anonymous functions are ignored. Move to imports statement C=C C=f i (go-goto-imports) Move point to the block of imports.	Move to expression	C-x 4 C-c C-j	(godef-jump-other-window			
Move to current method receiver C-c C-f m (go-goto-imports)	definition in other window		POINT)	after that command, use M-, to go back to original point.		
### ARG) ### ARG) ### If there is none, add one beginning with the name of the current function. ### ARG) ### If there is none, add one beginning with the name of the current function. ### ARG) ### ARG) ### Or to function ### Or to function ### Or to function ### ARG) ### Or to function ### Or	Move to current function arguments	C-c C-f a	, , ,			
Anonymous functions do not have docstrings, so when this is called interactively anonymous functions the skipped. If called programmatically, an error is raised unless ARG is non-nil. Go to the function definition	Move to current	C-c C-f d				
Move to function definition C = C = f f (go-goto-function &optional ARG) Go to the receiver function will be skipped. If called programmatically, an error is raised unless ARG is non-nil. Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-method-receiver & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name ARG is non-nil, anonymous functions are skipped. Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name ARG is non-nil, anonymous function are skipped. Move to current function name C = C = f n (go-goto-function-name & optional ARG) Move to current function name ARG is non-nil, anonymous functions are skipped. Move to current function name ARG is non-nil, anonymous functions are skipped. Move to current function name C = C = f n (go-goto-function-name & option name & opti	function docstring		ARG)	Anonymous functions do not have docstrings, so when this is called interactively		
If we are no a docstring, follow the docstring down. If no function is una an account of the surrounding down. If no function is una docstring, follow the docstring down. If no function is una summer was a the top of a file and search forward instead. If point is looking at the func keyword of an anonymous function, go to the surrounding function. If ARG is non-nil, anonymous functions are ignored. Move point to the block of imports. If using import (
If no function is found, assume that we are at the top of a file and search forward instead. If point is looking. If point is looking the func keyword of an anonymous function, go to the surrounding function. If ARG is non-nil, anonymous functions are ignored. Move to imports statement	Move to function	C-c C-f f	(go-goto-function &optional ARG)			
Move to imports statement C-c C-f i (go-goto-imports) (go-goto-imports) Move point to the block of imports. If using	definition			If no function is found, assume that we are at the top of a file and search forward instead.		
C-c C-f i (go-goto-imports) (go-goto-imports) Move point to the block of imports. If using import (function.		
If using import ("foo" "bar" it will move point directly behind the last import. it will move point directly behind the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will move point to the next line after the last import. it will nove point to the next line after the last import. it will nove point to the next line after the last import. it will move point to the next line after the last import. it will nove point on the last import. it will move point to the next line after the last import. it will nove point will be moved after the package declaration. Go to the name of the current method. if there is none, add parenthesis to add one. if the function san the parenthesis is called interactively anonymous functions will be skipped. Go to the name of the current function. if the function is anonymous functions are skipped. Go to the name of the current function. if there are multiple ones contained in a parenthesis, enter the parenthesis. if there is none, make space for one to be added. if ARG is non-nil, anonymous functions are skipped. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginnin						
it will move point directly behind the last import. If using import "foo" it will move point to the next line after the last import. If no imports can be found, point will be moved after the package declaration. Move to current method receiver C-c C-f m (go-goto-method-receiver & optional ARG) Move to current function name C-c C-f n (go-goto-function-name & optional ARG) Move to current function name C-c C-f n (go-goto-function-name & optional ARG) Move to current function return value declaration C-c C-f r (go-goto-return-values & optional ARG) Move to current function return value declaration C-c C-f r (go-goto-return-values & optional ARG) Move to current function return value declaration C-c C-f r (go-goto-return-values & optional ARG) Move to current function return value declaration C-c C-f r (go-goto-return-values & optional ARG) ARG) C-c C-f r (go-goto-return-values & optional ARG) ARG) Move to current function return value declaration of the current function. If there are multiple ones contained in a parenthesis, enter the parenthesis. If there are multiple ones contained in a parenthesis, enter the parenthesis. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. Move backward to the beginning of a defun. C-M-<\no move beginning of a defun. Soft hat many times. Negative ARG means move forward to the ARGth following beginning of defun. Soft hat many times. Negative ARG means move forward to the ARGth following beginning of defun. Soft hat many times one, not in terminal mode (for C-M-a and C-M-shome>). However<\(\frac{6}{6} > p \) and \(\frac{6}{6} < \text{up} > \) and \(\frac{6}{6} < \text{up} > \) and \(\frac{6}{6} < \text{up} > \) however<\(\frac{6}{6} > p \) and \(\frac{6}{6} < \text{up} > \) however<\(\frac{6}{6} > p \) and \(\frac{6}{6} < \text{up} > \) however<\(\frac{6}{6} > p \) and \(\frac{6}{6} < \text{up} > \) however<\(\frac{6}{6} > p \) and \(\frac{6}{6} < \text{up} > \) however<\(\frac{6}{6} > p \)	Move to imports statement	C-c C-f i	(go-goto-imports)	• If using import (
If using import "foo" import "bar" it will move point to the next line after the last import.						
import "bar" it will move point to the next line after the last import. if no imports can be found, point will be moved after the package declaration. Move to current method receiver						
it will move point to the next line after the last import. If no imports can be found, point will be moved after the package declaration. Move to current method receiver Move to current function name C-c C-f m (go-goto-method-receiver & optional ARG) Move to current function name C-c C-f n (go-goto-function-name & optional ARG) Move to current function name C-c C-f n (go-goto-function-name & optional ARG) Move to current function E-c C-f n (go-goto-function-name & optional ARG) Move to current function E-c C-f r (go-goto-return-values & optional ARG) Move to current function If the function is a test, place point after Test. If the function is anonymous, place point on the 'func' keyword. If there is none, add parenthesis to add one. And promain function. If the function is a test, place point after Test. If the function is a rest, place point after Test. If the function is anonymous, place point on the 'func' keyword. If there is none, make space for one to be added. If there is none, make space for one to be added. If there is none, make space for one to be added. If there is none, make space for one to be added. If there is none, make space for one to be added. If there is none, make space for one to be added. If there is none, make space for one to be added. If there is none, make space for one to be added. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- Shift marking is available in graphics mode, not in terminal						
Move to current method receiver C-c C-f m (go-goto-method-receiver & optional ARG) (go-goto-function-name & opt						
• Anonymous functions cannot have method receivers, so when this is called interactively anonymous functions will be skipped. If called programmatically, an error is raised unless ARG is non-nil. Move to current function name C-c C-f n (go-goto-function-name & optional ARG) Move to current function is a test, place point after 'Test'. If the function is a test, place point on the 'func' keyword. If ARG is non-nil, anonymous functions are skipped. Go to the return value declaration of the current function. If there are multiple ones contained in a parenthesis, enter the parenthesis. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. Backward to beginning of function definition C-M-a C-M-a C-M-chome> (beginning-of-defun & optional ARG) Move backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M-chome>). However <f6> p and <f6> <up> handle Shift-marking fine in terminal</up></f6></f6>	Move to current	C-c C-f m	1.5	Go to the receiver of the current method.		
ARG is non-nil. Move to current function name C-c C-f n (go-goto-function-name & optional ARG) Move to current function is a test, place point after 'Test'. If the function is a test, place point on the 'func' keyword. If ARG is non-nil, anonymous functions are skipped. Go to the return value declaration of the current function. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. C-c C-f r (go-goto-return-values & optional ARG) ARG) C-d C-f r (go-goto-return-values & optional ARG) Go to the return value declaration of the current function. If there are multiple ones contained in a parenthesis, enter the parenthesis. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. C-M-a C-M-a C-M-a C-M-c-home> C-M-a ARG) Move backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- - home>). However <f6> p and <f6> <up> handle Shift-marking fine in terminal</up></f6></f6>	method receiver		&optional ARG)	Anonymous functions cannot have method receivers, so when this is called interactively		
Move to current function name C-c C-f n (go-goto-function-name & of the current function. If the function is a test, place point after 'Test'. If the function is anonymous, place point on the 'func' keyword. If ARG is non-nil, anonymous functions are skipped. C-c C-f r (go-goto-return-values & optional ARG) Move to current function return value declaration Final time function return value declaration C-c C-f r (go-goto-return-values & optional ARG) Go to the return value declaration of the current function. If there are multiple ones contained in a parenthesis, enter the parenthesis. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. C-M-a C-M-a C-M-A-home> C-M-a C-M-A-home> C-M-A-home> C-M-A-home> C-M-C-M-A ARG) Move backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M-home>). However <f6> p and <f6> <up> handle Shift-marking fine in terminal</up></f6></f6>				anonymous functions will be skipped. If called programmatically, an error is raised unless		
• If the function is anonymous, place point on the 'func' keyword. • If ARG is non-nil, anonymous functions are skipped. Move to current function return value declaration C-c C-f r (go-goto-return-values &optional ARG) Go to the return value declaration of the current function. • If there are multiple ones contained in a parenthesis, enter the parenthesis. • If there is none, make space for one to be added. • If ARG is non-nil, anonymous functions are skipped. Move backward to the beginning of a defun. • C-M-a • C-M- <home> • <f6> p • <f6> < up> • C-I C-a (beginning-of-defun &optional ARG) Move backward to the beginning of a defun. • With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. • Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M-<home>). However <f6> p and <f6> < up> handle Shift-marking fine in terminal</f6></f6></home></f6></f6></home>	Move to current	C-c C-f n		Go to the name of the current function.		
Move to current function return value declaration C-c C-f r (go-goto-return-values & optional ARG) Go to the return value declaration of the current function. If there are multiple ones contained in a parenthesis, enter the parenthesis. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. Backward to beginning of function definition C-M-a C-M-(home) C-M-a C-M-(home) C-M-(home) C-G-C-C ARG) (beginning-of-defun & optional ARG) Move backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- If ARG is non-nil, anonymous functions are skipped. Move backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- If ARG is non-nil, anonymous functions are skipped.	function name		&optional ARG)	If the function is anonymous, place point on the 'func' keyword.		
function return value declaration ARG If there are multiple ones contained in a parenthesis, enter the parenthesis. If there is none, make space for one to be added. If ARG is non-nil, anonymous functions are skipped. C-M-a C-M-a C-M-chome> (beginning-of-defun & optional ARG) Move backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- <nome> C-I C-a Legal (beginning-of-defun & optional ARG) Nove backward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- <nome> Nowebackward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- <nome> Nowebackward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- <nome> Nowebackward to the beginning of a defun. With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun.</nome></nome></nome></nome>		_	,	If ARG is non-nil, anonymous functions are skipped.		
• If there is none, make space for one to be added. • If ARG is non-nil, anonymous functions are skipped. Backward to beginning of function definition • C-M-a • C-M- <home> • (beginning-of-defun &optional ARG) Move backward to the beginning of a defun. • With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. • Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M-<home>). However <f6> p and <f6> <up>handle Shift-marking fine in terminal</up></f6></f6></home></home>	Move to current function return	C-c C-f r	, , ,	If there are multiple ones contained in a parenthesis, enter the parenthesis.		
Backward to beginning of function definition • C-M-a • C-M-(home) • (f6> p • (f6> < up) • C-[C-a • C-M-a • With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. ⇒ Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- • Nowever < f6> p and < f6> < up> handle Shift-marking fine in terminal	value declaration					
function definition • <f6> p • <f6> <up> • C-[C-a following beginning of defun. Following beginning of defun.</up></f6></f6>	Backward to			Move backward to the beginning of a defun.		
• <f6> <up> • C-[C-a Shift marking is available in graphics mode, not in terminal mode (for C-M-a and C-M- <norm> \text{Norm} N</norm></up></f6>	beginning of function definition		ARG)	following beginning of defun.		
, c-[c-u		• <f6> <up></up></f6>				
		•		, -		

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Forward to end of function and class definition	• C-M-e • C-M- <end> • <f6> <right> • C-[C-e • Esc C-e</right></f6></end>	(end-of-defun &optional ARG)	Move forward to next end of defun. With argument, do it that many times. Negative argument -N means move back to Nth preceding end of defun. ➡Shift marking is available in graphics mode, not in terminal mode (for C-M-e, C-[C-e and Esc C-e keys). However <f6> <right> handle Shift-marking fine in terminal mode.</right></f6>
Forward to start of next function definition	• <f6> n • <f6> <down></down></f6></f6>	(pel-beginning-of-next-defun &optional SILENT DONT- PUSH_MARK)	Move forward to the beginning of the next function definition. • Beeps if does not find beginning of next function unless SILENT is non-nil. • If the beginning of next function is found, push the start location to the mark ring unless DONT-PUSH_MARK is non-nil. • Move back to previous position with M−ˆ. ➡Shift marking is available.
Backward to end of previous function definition	<f6> <left></left></f6>	(pel-end-of-previous-defun &optional SILENT DONT- PUSH_MARK)	Move backwards to the end of the previous function definition. • Beeps if does not find end of previous function unless SILENT is non-nil. • If the end of previous function is found, push the start location to the mark ring unless DONT-PUSH_MARK is non-nil. • Move back to previous position with M−ˆ. ➡Shift marking is available.
Indent expression at point	C-M-q	(prog-indent-sexp &optional DEFUN)	Indent the expression after point. When interactively called with prefix, indent the enclosing defun instead.
Go Syntax Checking Using either: • flycheck or • flymake	Syntax checking for the Go programming language can be done with Emacs built-in flymake as well as with the external package flycheck. To activate either set the pel-use-goflymake user option is set to either 'use-flycheck or 'use-flymake. By default, the syntax checker is not automatically launched. If you want to start your selected syntax checker as soon as a .go file is opened, add 'go-mode to the pel-modes-activating-syntax-check user-option. PEL automatically installs and activates flycheck when pel-use-goflymake user option is set to 'use-flycheck. flymake is built-in Emacs. Support for those is provided by the external go-flymake.el and go-flycheck.el files. These 2 packages use the goflymake Go program, which must be installed separately. To install the goflymake executable do the following: Install Go on your computer if this is not already done. See instruction at the top of this page. Set the GOPATH for your project. Run the following command: go get -u github.com/dougm/goflymake The above command will get goflymake source and install the goflymake executable file inside the bin directory of your Go project identified by the GOPATH. You will probably want to edit code in several Go projects, so it might be a good idea to either copy or create a symlink in one of the directories in your PATH to that file, allowing you to change GOPATH and continue to use the goflymake binary.		
Activate/deactivate selected syntax checker	<f11> SPC g ! <f12> !</f12></f11>	(pel-go-toggle-syntax-checker)	Toggle the selected Go syntax checker mode on/off. The syntax checker activated or deactivated is either flycheck or flymake, as selected by the user-option variable `pel-use-goflymake'. See the required settings above to activate this command and select the syntax checker.

Go- References

Document	Notes
Go Programming Language	
The Go Programming Language - Wikipedia	