letter

Output	Say
a	alfa
b	bat
c	cap
d	delta
e	echo
f	fine
g	golf
h	hotel
i	sit
j	jury
k	kilo
l	lima
m	mike
n	near
o	odd
p	papa
q	quebec
r	red
s	sun
t	trap
u	urge
v	vest
w	whiskey
x	plex
y	yank
z	zulu

number key

Output	Say
0	zero
1	one
2	two
3	three
4	four
5	five
6	six
7	seven
8	eight
9	nine

modifier key

Output	Say
alt	alt
ctrl	control
shift	shift
super	super

special key

Output	Say
end	end
enter	enter
escape	escape
home	home
insert	insert
pagedown	pagedown
pageup	pageup
space	space
tab	tab
backspace	delete
delete	forward delete
pageup	page up
pagedown	page down
menu	menu key
printscr	print screen

symbol key

Output	Say
.	dot
.	point
'	quote
'	apostrophe
[	L square
[	left square
[	square
]	R square
]	right square
/	slash
\	backslash
-	minus
-	dash
=	equals
+	plus
~	tilde
!	bang
_	down score
_	under score
(	paren
(	L paren
(	left paren
)	R paren
)	right paren
{	brace
{	left brace
}	R brace
}	right brace
<	angle
<	left angle
<	less than
>	rangle
>	R angle
>	right angle
>	greater than
*	star
#	hash
%	percent
^	caret
&	amper
|	pipe
"	dubquote
"	double quote
$	dollar
£	pound
`	`
,	,
`	back tick
`	grave
,	comma
.	period

arrow key

Output	Say
down	down
left	left
right	right
up	up

punctuation

Output	Say
`	`
,	,
`	back tick
`	grave
,	comma
.	period
.	full stop
;	semicolon
:	colon
/	forward slash
?	question mark
!	exclamation mark
!	exclamation point
*	asterisk
#	hash sign
#	number sign
%	percent sign
@	at sign
&	and sign
&	ampersand
$	dollar sign
£	pound sign

function key

Output	Say
f1	F one
f2	F two
f3	F three
f4	F four
f5	F five
f6	F six
f7	F seven
f8	F eight
f9	F nine
f10	F ten
f11	F eleven
f12	F twelve

generic browser

Output	Say
focus_address()	address bar | go address | go url
focus_address() sleep(50ms) edit.copy()	address copy | url copy | copy address | copy url
go_home()	go home
go_forward()	[go] forward
go_back()	go (back | backward)
go(website)	go to {user.website}
open_private_window()	go private
bookmark()	bookmark it
bookmark_tabs()	bookmark tabs
reload()	(refresh | reload) it
reload_hard()	(refresh | reload) it hard
bookmarks()	bookmark show
bookmarks_bar()	bookmark bar [show]
show_downloads()	downloads show
show_extensions()	extensions show
show_history()	history show
show_clear_cache()	cache show
toggle_dev_tools()	dev tools [show]
show_downloads()	show downloads
show_extensions()	show extensions
show_history()	show history
show_clear_cache()	show cache

generic snippets

Output	Say
snippet_insert(user.snippets)	snip {user.snippets}
snippet_search(user.text)	snip hunt <user.text>
snippet_search("")	snip hunt
snippet_create()	snip create
snippet_toggle()	snip show

generic snippets open

Output	Say
snippet_hide()	snip close

generic terminal

Output	Say
terminal_list_directories()	lisa
terminal_list_all_directories()	lisa all
terminal_change_directory(text or "")	katie [<user.text>]
terminal_change_directory_root()	katie root
terminal_clear_screen()	clear screen
terminal_run_last()	run last
terminal_rerun_search(text or "")	rerun [<user.text>]
terminal_rerun_search("")	rerun search
terminal_kill_all()	kill all
edit.copy() sleep(50ms) edit.paste()	copy paste

tmux

Output	Say
"tmux "	mux
insert('tmux new ')	mux new session
key(ctrl-b) key(s)	mux sessions
key(ctrl-b) key($)	mux name session
insert('tmux kill-session -t ') #window management	mux kill session
key(ctrl-b) key(c)	mux new window
key(ctrl-b) key('{number}')	mux window <number>
key(ctrl-b) key(p)	mux previous window
key(ctrl-b) key(n)	mux next window
key(ctrl-b) key(,)	mux rename window
key(ctrl-b) key(&) #pane management	mux close window
key(ctrl-b) key(%)	mux split horizontal
key(ctrl-b) key(")	mux split vertical
key(ctrl-b) key(o)	mux next pane
key(ctrl-b) key(arrow_key)	mux move <user.arrow_key>
key(ctrl-b) key(x) #Say a number right after this command, to switch to pane	mux close pane
key(ctrl-b) key(q)	mux pane numbers

1password

Output	Say
password_new()	password new
password_duplicate()	password dup
password_edit()	password edit
password_delete()	password delete

1password global

Output	Say
password_fill()	password fill
password_show()	password show

anaconda

Output	Say
"conda "	anaconda
"conda –help\n"	anaconda help
"conda –version\n"	anaconda version
"conda env list\n"	anaconda environment list
"conda env create -f "	anaconda environment create
"conda env remove -n "	anaconda environment remove
"conda activate "	anaconda activate
"conda clean "	anaconda clean
"conda compare "	anaconda compare
"conda config "	anaconda config
"conda create "	anaconda create
"conda info "	anaconda info
"conda init "	anaconda init
"conda install "	anaconda install
"conda list "	anaconda list
"conda package "	anaconda package
"conda remove "	anaconda remove
"conda uninstall "	anaconda uninstall
"conda run "	anaconda run
"conda search "	anaconda search
"conda update "	anaconda update
"conda upgrade "	anaconda upgrade
"conda build "	anaconda build
"conda convert "	anaconda convert
"conda debug "	anaconda debug
"conda develop "	anaconda develop
"conda env "	anaconda environment
"conda index "	anaconda index
"conda inspect "	anaconda inspect
"conda metapackage "	anaconda metapackage
"conda render "	anaconda render
"conda server "	anaconda server
"conda skeleton "	anaconda skeleton
"conda verify "	anaconda verify

discord

Output	Say
discord_mentions_last()	[channel] mentions last
discord_mentions_next()	[channel] mentions next
discord_oldest_unread()	oldest unread
discord_toggle_pins()	toggle pins
discord_toggle_inbox()	toggle inbox
discord_toggle_members()	toggle (members | member list)
discord_emoji_picker()	pick emoji
discord_gif_picker()	pick (jif | gif | gift)
discord_mark_inbox_read()	mark inbox channel read
discord_mute()	[toggle] (mute | unmute)
discord_deafen()	[toggle] (deafen | undeafen)
discord_answer_call()	answer call
discord_decline_call()	decline call

signal

Output	Say
key("ctrl-/")	show shortcuts
key("ctrl-t")	(next|nav|navigate) [by] (sec|section)
key("alt-down")	(prev|previous) (chat|conversation)
key("alt-up")	next (chat|conversation)
key("alt-shift-down")	(prev|previous) unread
key("alt-shift-up")	next unread
key("ctrl-,")	[open] (pref|preferences)
key("ctrl-shift-l")	open conversation menu
key("ctrl-f")	search
key("ctrl-shift-f")	search chat
key("ctrl-shift-t")	focus (chat|composer)
key("ctrl-shift-m")	open media
key("ctrl-shift-j")	open emoji
key("ctrl-shift-s")	open sticker
key("ctrl-shift-v")	record [voice] message
key("ctrl-shift-a")	archive chat
key("ctrl-shift-u")	unarchive chat
key("ctrl-up")	(first|top) message
key("ctrl-down")	(last|bottom) message
key("ctrl-shift-c")	close chat
key("enter")	send it
key("ctrl-d")	message details
key("ctrl-shift-r")	reply [message]
key("ctrl-shift-e")	react [message]
key("ctrl-s")	save attachment
key("ctrl-shift-d")	delete [message]
key("ctrl-enter")	send message
key("ctrl-shift-x")	expand chat
key("ctrl-u")	attach [file]
key("ctrl-p")	remove [link] preview
key("ctrl-shift-p")	remove [link] attachment

tmux

Output	Say
"tmux "	mux
insert('tmux new ')	mux new session
key(ctrl-b) key(s)	mux sessions
key(ctrl-b) key($)	mux name session
insert('tmux kill-session -t ') #window management	mux kill session
key(ctrl-b) key(c)	mux new window
key(ctrl-b) key('{number}')	mux window <number>
key(ctrl-b) key(p)	mux previous window
key(ctrl-b) key(n)	mux next window
key(ctrl-b) key(,)	mux rename window
key(ctrl-b) key(&) #pane management	mux close window
key(ctrl-b) key(%)	mux split horizontal
key(ctrl-b) key(")	mux split vertical
key(ctrl-b) key(o)	mux next pane
key(ctrl-b) key(arrow_key)	mux move <user.arrow_key>
key(ctrl-b) key(x) #Say a number right after this command, to switch to pane	mux close pane
key(ctrl-b) key(q)	mux pane numbers

explorer

Output	Say
file_manager_open_volume("{letter}:")	go <user.letter>
file_manager_open_directory("%AppData%")	go app data
file_manager_open_directory("%programfiles%")	go program files

wsl

Output	Say
file_manager_open_volume("/mnt/{letter}")	go <user.letter>
wsl_reset_path_detection()	(wsl|weasel) reset path detection
wsl_speak()	(wsl|weasel) speak

slack win

Output	Say
key("ctrl-{number}")	workspace <number>
key(ctrl-shift-i)	(slack | lack) [channel] info
key(ctrl-`)	focus (move | next)
key(f6)	(section | zone) [next]
key(shift-f6)	(section | zone) (previous | last)
key(ctrl-shift-k)	(slack | lack) [direct] messages
key(ctrl-shift-t)	(slack | lack) threads
key(alt-left)	(slack | lack) (history [next] | back | backward)
key(alt-right)	(slack | lack) forward
key(tab)	(element | bit) [next]
key(shift-tab)	(element | bit) (previous | last)
key(ctrl-shift-m)	(slack | lack) (my stuff | activity)
key(ctrl-shift-e)	(slack | lack) directory
key(ctrl-shift-s)	(slack | lack) (starred [items] | stars)
key(ctrl-shift-a)	(slack | lack) unread [messages]
key(shift-up)	grab left
key(shift-down)	grab right
key(shift-enter)	add line
key(ctrl-shift-\)	(slack | lack) (react | reaction)
key(ctrl-shift-c)	(insert command | commandify)
insert("```")	insert code
key(ctrl-shift-8)	(slack | lack) (bull | bullet | bulleted) [list]
key(ctrl-shift-7)	(slack | lack) (number | numbered) [list]
key(ctrl-shift-9)	(slack | lack) (quotes | quotation)
key(ctrl-b)	bold
key(ctrl-i)	(italic | italicize)
key(ctrl-shift-x)	(strike | strikethrough)
key(ctrl-shift-enter)	(slack | lack) snippet
key(m)	([toggle] mute | unmute)
key(v)	(slack | lack) ([toggle] video)
key(a)	(slack | lack) invite
key(ctrl-/)	(slack | lack) shortcuts
"{text}"	emote <user.text>
key(ctrl-shift-d)	toggle left sidebar

talon repl

Output	Say
phrase = user.history_get(1) command = "sim('{phrase}')" insert(command) key(enter)	test last
insert("sim('{phrase}')") key(enter)	test <phrase>

phrase = user.history_get(number_small)
command = "sim('{phrase}')"
#to do: shouldn't this work?
#user.paste("sim({phrase})")
insert(command)
key(enter) # requires user.talon_populate_lists tag. do not use with dragon
	test numb <number_small>

insert("actions.find('{user.talon_actions}')") key(enter) # requires user.talon_populate_lists tag. do not use with dragon	debug action {user.talon_actions}
insert("actions.user.talon_pretty_print(registry.lists['{talon_lists}'])") key(enter)	debug list {user.talon_lists}
insert("actions.user.talon_pretty_print(registry.tags)") key(enter)	debug tags
insert("actions.user.talon_pretty_print(registry.settings)") key(enter)	debug settings
insert("actions.user.talon_pretty_print(scope.get('mode'))") key(enter) # requires user.talon_populate_lists tag. do not use with dragon	debug modes
insert("actions.user.talon_pretty_print(scope.get('{talon_scopes}'))") key(enter)	debug scope {user.talon_scopes}
insert("actions.user.talon_pretty_print(ui.apps(background=False))") key(enter)	debug running apps
insert("actions.user.talon_pretty_print(ui.windows())") key(enter)	debug all windows
insert("actions.user.talon_debug_app_windows('{running}')")	debug {user.running} windows

github

Output	Say
key(s)	focus search
insert("gn")	go to notifications
insert("gd")	go to dashboard
key(?)	(keyboard shortcuts show | show keyboard shortcuts)
key(j)	(selection move down | move selection down)
key(k)	(selection move up | move selection up)
key(x)	(selection toggle | toggle selection)
key(o)	(selection open | open selection)
insert("gc")	go to code
insert("gi")	go to issues
insert("gp")	go to pull requests
insert("gw")	go to wiki
insert("ga")	go to actions
insert("gb")	go to projects
insert("gg")	go to discussions
key(.)	[web] editor open
key(t)	(file find | find file)
key(l)	jump to line
key(w)	((branch|tag) switch | switch (branch|tag))
key(y)	(url expand | expand url)
key(i)	(show|hide) comments
key(b)	blame view open
key(a)	(show|hide) annotations
key(c)	(issue create | create [an] issue)
key(/)	search (issues|[pull] requests)
key(l)	(filter by|edit) labels
key(m)	(filter by|edit) milestones
key(a)	(filter by|edit) assignee
key(r)	reply
key(ctrl-enter)	(comment submit | submit comment)
key(ctrl-shift-p)	(comment preview | preview comment)
key(ctrl-shift-l)	git hub full screen
key(escape)	(form close | close form)
key(p)	parent commit
key(o)	other parent commit
key(y)	mark as read
key(shift-m)	(thread mute | mute thread)
key(o)	(issue open | open issue)

outlook

Output	Say
key(n)	new message
key(alt-s)	send [this] message
key(r)	reply [to] [this] message
key(ctrl-shift-r)	reply all [to] [this] message
key(ctrl-shift-f)	forward [this] message
key(ctrl-s)	save [draft]
key(esc)	discard [draft]
key(ctrl-k)	insert [a] [hyper] link
key(ctrl-space)	(select|unselect) [this] message
key(ctrl-a)	select all [messages]
key(esc)	clear all [messages]
key(home)	select first [message]
key(and)	select last [message]
key(o)	open [this] message
key(shift-enter)	open [this] message [in] [a] new window
key(esc)	close [this] message
key(ctrl-.)	[open] [the] next (item|message)
key(ctrl-,)	[open] [the] (prev|previous) item
key(.)	next reading [pane] (item|message)
key(,)	(prev|previous) [pane] (item|message)
key(x)	(expand|collapse) [conversation]
key(ctrl-shift-1)	go [to] mail
key(ctrl-shift-2)	go [to] calendar
key(ctrl-shift-3)	go [to] people
key(ctrl-shift-4)	go [to] to do
key(g) key(i)	go [to] inbox
key(g) key(d)	go to drafts
key(g) key(s)	go to sent
key(alt-q)	search [email]
key(?)	show help
key(ctrl-z)	undo [last] [action]
key(delete)	delete [this] [message]
key(shift+delete)	(perm|permanently) delete [this] [message]
key(shift-e)	new folder
key(q)	mark [this] [(item|message)] as read
key(u)	mark [this] [(item|message)] as unread
key(insert)	flag [this] [(item|message)]
key(e)	archive
key(j)	mark [this] [message] [as] junk

twitter

Output	Say
key(?)	(show shortcuts|shortcuts help)
key(j)	next tweet
key(k)	previous tweet
key(space)	page down
key(.)	load new tweet
insert("gh")	go home
insert("ge")	go explore
insert("gn")	go notifications
insert("gr")	go mentions
insert("gp")	go profile
insert("gl")	go likes
insert("gi")	go lists
insert("gm")	go direct messages
insert("gs")	go settings
insert("gb")	go book marks
insert("gu")	go to user
insert("gd")	display settings
key(n)	new tweet
key(ctrl-enter)	send tweet
key(m)	new direct message
key(/)	search
key(l)	like message
key(r)	reply message
key(t)	re tweet [message]
key(s)	share tweet
key(b)	bookmark
key(urge)	mute account
key(x)	block account
key(enter)	open details
key(o)	expand photo

draft editor

Output	Say
draft_editor_open()	draft this
edit.select_all() user.draft_editor_open()	draft all
edit.select_line() user.draft_editor_open()	draft line

draft editor open

Output	Say
draft_editor_submit()	draft submit
draft_editor_discard()	draft discard

emoji

Output	Say
"{emoticon}"	emoticon {user.emoticon}
paste(emoji)	emoji {user.emoji}
paste(kaomoji)	kaomoji {user.kaomoji}

abbreviate

Output	Say
"{abbreviation}"	(abbreviate|abreviate|brie {user.abbreviation}

cancel

Output	Say
skip()	cancel cancel
app.notify("Command ignored")	ignore [<phrase>]

chapters

Output	Say
chapter_next()	chapter next
chapter_previous()	chapter last
chapter_jump(number)	go chapter <number>
chapter_final()	go chapter final

datetimeinsert

Output	Say
insert(user.time_format("%Y-%m-%d"))	date insert
insert(user.time_format_utc("%Y-%m-%d"))	date insert UTC
insert(user.time_format("%Y-%m-%d %H:%M:%S"))	timestamp insert
insert(user.time_format("%Y-%m-%d %H:%M:%S.%f"))	timestamp insert high resolution
insert(user.time_format_utc("%Y-%m-%d %H:%M:%S"))	timestamp insert UTC
insert(user.time_format_utc("%Y-%m-%d %H:%M:%S.%f"))	timestamp insert UTC high resolution

desktops

Output	Say
desktop(number_small)	desk <number_small>
desktop_next()	desk next
desktop_last()	desk last
desktop_show()	desk show
window_move_desktop(number)	window move desk <number>
window_move_desktop_left()	window move desk left
window_move_desktop_right()	window move desk right

extensions

Output	Say
"{file_extension}"	{user.file_extension}

formatters

Output	Say
insert_formatted(text, "NOOP")	phrase <user.text>
insert_formatted(text, "NOOP")	phrase <user.text> over
insert_formatted(prose, prose_formatter)	{user.prose_formatter} <user.prose>
insert_formatted(prose, prose_formatter)	{user.prose_formatter} <user.prose> over
insert_many(format_text_list)	<user.format_text>+
insert_many(format_text_list)	<user.format_text>+ over
formatters_reformat_selection(user.formatters)	<user.formatters> that
insert_formatted(user.word, "NOOP")	word <user.word>
toggle_phrase_history()	recent list
phrase_history_hide()	recent close
insert(user.get_recent_phrase(number_small))	recent repeat <number_small>
clip.set_text(user.get_recent_phrase(number_small))	recent copy <number_small>
select_last_phrase()	select that
before_last_phrase()	before that
clear_last_phrase()	nope that | scratch that
formatters_reformat_last(formatters)	nope that was <user.formatters>

git

Output	Say
"git add . -p\n"	git add patch
"git add "	git add
"git add -u\n"	git add everything
"git bisect "	git bisect
"git blame "	git blame
"git branch "	git branch
"git branch –remote\n"	git remote branches
"git branch {text}"	git branch <user.text>
"git checkout "	git checkout
"git checkout master\n"	git checkout master
"git checkout main\n"	git checkout main
"git checkout {text}"	git checkout <user.text>
"git cherry-pick "	git cherry pick
"git cherry-pick –continue "	git cherry pick continue
"git cherry-pick –abort "	git cherry pick abort
"git cherry-pick –skip "	git cherry pick skip
"git clone "	git clone
"git clean -dfx"	git clean everything
"git commit -m '{text}'"	git commit message <user.text>
"git commit\n"	git commit
"git commit –amend\n"	git commit amend
"git commit -a\n"	git commit existing
"git diff –color-words "	git diff (colour|color) words
"git diff "	git diff
"git diff –cached\n"	git diff cached
"git fetch\n"	git fetch
"git fetch –all\n"	git fetch all
"git fetch {text}"	git fetch <user.text>
"git fetch –prune\n"	git fetch prune
"git init\n"	git in it
"git log\n"	git log all
"git log -c\n"	git log all changes
"git log "	git log
"git log -c "	git log changes
"git merge "	git merge
"git merge {text}"	git merge <user.text>

git add patch

Output	Say
key(y) key(enter)	yank
key(n) key(enter)	near
key(q) key(enter)	quench
key(d) key(enter)	drum
key(a) key(enter)	air

help

Output	Say
help_list("user.letter")	help alphabet
help_list("user.symbol_key")	help symbols
help_list("user.number_key")	help numbers
help_list("user.punctuation")	help punctuation
help_list("user.modifier_key")	help modifier
help_list("user.special_key")	help special keys
help_list("user.function_key")	help function keys
help_list("user.arrow_key")	help arrows
help_formatters(user.get_formatters_words())	(help formatters | help format | format help)
help_context()	help context
help_context_enabled()	help active
help_search(text)	help search <user.text>
help_selected_context(help_contexts)	help context {user.help_contexts}
help_search("help")	help help
help_scope_toggle()	help scope

help open

Output	Say
help_next()	help next
help_previous()	help previous
help_select_index(number - 1)	help <number>
help_return()	help return
help_refresh()	help refresh
help_hide()	help close

help scope open

Output	Say
help_scope_toggle()	scope (hide | close)

history

Output	Say
history_toggle()	command history
history_disable()	command history close
history_clear()	command history clear
history_less()	command history less
history_more()	command history more

keys

Output	Say
move_cursor(arrow_keys)	go <user.arrow_keys>
key(letter)	<user.letter>
insert_formatted(letters, "ALL_CAPS")	(ship | uppercase) <user.letters> [(lowercase | sunk)]
key(symbol_key)	<user.symbol_key>
key(function_key)	<user.function_key>
key(special_key)	<user.special_key>
key("{modifiers}-{unmodified_key}")	<user.modifiers> <user.unmodified_key>
key(modifiers)	press <user.modifiers>
key(keys)	press <user.keys>

macro

Output	Say
macro_record()	macro record
macro_stop()	macro stop
macro_play()	macro play

media

Output	Say
key(volup)	volume up
key(voldown)	volume down
media_set_volume(number)	set volume <number>
key(mute)	(volume|media) mute
key(next)	[media] play next
key(prev)	[media] play previous
play_pause()	media (play | pause)

messaging

Output	Say
messaging_workspace_previous()	previous (workspace | server)
messaging_workspace_next()	next (workspace | server)
messaging_open_channel_picker()	channel
messaging_open_channel_picker() insert(user.formatted_text(user.text, "ALL_LOWERCASE"))	channel <user.text>
messaging_channel_previous()	channel up
messaging_channel_next()	channel down
messaging_unread_previous()	([channel] unread last | gopreev)
messaging_unread_next()	([channel] unread next | goneck)
messaging_open_search()	go (find | search)
messaging_mark_workspace_read()	mark (all | workspace | server) read
messaging_mark_channel_read()	mark channel read
messaging_upload_file()	upload file

microphone selection

Output	Say
microphone_selection_toggle()	microphone show
microphone_selection_hide()	microphone close
microphone_select(number_small)	microphone pick <number_small>

mouse cursor

Output	Say
mouse_show_cursor()	curse yes
mouse_hide_cursor()	curse no

multiple cursors

Output	Say
multi_cursor_enable()	cursor multiple
multi_cursor_disable()	cursor stop
multi_cursor_add_above()	cursor up
multi_cursor_add_below()	cursor down
multi_cursor_select_fewer_occurrences()	cursor less
multi_cursor_select_more_occurrences()	cursor more
multi_cursor_skip_occurrence()	cursor skip
multi_cursor_select_all_occurrences()	cursor all
multi_cursor_add_to_line_ends()	cursor lines

pages

Output	Say
page_next()	page next
page_previous()	page last
page_jump(number)	go page <number>
page_final()	go page final

repeater

Output	Say
core.repeat_command(ordinals-1)	<user.ordinals>
core.repeat_command(number_small-1)	<number_small> times
core.repeat_command(1)	(repeat that|twice)
core.repeat_command(number_small)	repeat that <number_small> [times]

screens

Output	Say
screens_show_numbering()	screen numbers

screenshot

Output	Say
screenshot()	grab screen
screenshot(number_small)	grab screen <number_small>
screenshot_window()	grab window
screenshot_selection()	grab selection
screenshot_clipboard()	grab screen clip
screenshot_clipboard(number_small)	grab screen <number_small> clip
screenshot_window_clipboard()	grab window clip

talon helpers

Output	Say
menu.check_for_updates()	talon check updates
menu.open_log()	talon open log
menu.open_repl()	talon open rebel
menu.open_talon_home()	talon home
talon_add_context_clipboard_python()	talon copy context pie
talon_add_context_clipboard()	talon copy context
name = app.name() clip.set_text(name)	talon copy name
executable = app.executable() clip.set_text(executable)	talon copy executable
bundle = app.bundle() clip.set_text(bundle)	talon copy bundle
title = win.title() clip.set_text(title)	talon copy title
result = user.talon_version_info() print(result)	talon dump version
result = user.talon_version_info() user.paste(result)	talon insert version
result = user.talon_get_active_context() print(result)	talon dump context
phrase = user.history_get(1) user.talon_sim_phrase(phrase)	talon test last
phrase = user.history_get(number_small) user.talon_sim_phrase(phrase)	talon test numb <number_small>
talon_sim_phrase(phrase)	talon test <phrase>
talon_action_find("{user.talon_actions}")	talon debug action {user.talon_actions}
talon_debug_list(talon_lists)	talon debug list {user.talon_lists}
talon_copy_list(talon_lists)	talon copy list {user.talon_lists}
talon_debug_tags()	talon debug tags
talon_debug_modes()	talon debug modes
talon_debug_scope(talon_scopes)	talon debug scope {user.talon_scopes}
talon_debug_setting(talon_settings)	talon debug setting {user.talon_settings}
talon_debug_all_settings()	talon debug all settings
result = user.talon_get_active_application_info() print("**** Dumping active	talon debug active app

websites and search engines

Output	Say
open_url(website)	open {user.website}
search_with_search_engine(search_engine, user.text)	{user.search_engine} hunt <user.text>
text = edit.selected_text() user.search_with_search_engine(search_engine, text)	{user.search_engine} (that|this)

window management

Output	Say
app.window_open()	window (new|open)
app.window_next()	window next
app.window_previous()	window last
app.window_close()	window close
switcher_focus(running_applications)	focus <user.running_applications>
switcher_menu()	focus
switcher_toggle_running()	running list
switcher_hide_running()	running close
switcher_launch(launch_applications)	launch <user.launch_applications>
snap_window(window_snap_position)	snap <user.window_snap_position>
move_window_next_screen()	snap next [screen]
move_window_previous_screen()	snap last [screen]
move_window_to_screen(number)	snap screen <number>
snap_app(running_applications, window_snap_position)	snap <user.running_applications> <user.window_snap_position>
move_app_to_screen(running_applications, number)	snap <user.running_applications> [screen] <number>

language modes

Output	Say
code_set_language_mode("csharp")	force see sharp
code_set_language_mode("cplusplus")	force see plus plus
code_set_language_mode("go")	force go (lang|language)
code_set_language_mode("java")	force java
code_set_language_mode("javascript")	force java script
code_set_language_mode("typescript")	force type script
code_set_language_mode("markdown")	force markdown
code_set_language_mode("python")	force python
code_set_language_mode("terraform")	force terraform
code_set_language_mode("r")	force are language
code_set_language_mode("talon")	force talon [language]
code_clear_language_mode()	clear language modes
mode.enable("user.gdb")	[enable] debug mode
mode.disable("user.gdb")	disable debug mode

modes

Output	Say
mode.disable("sleep") mode.disable("command") mode.enable("dictation") user.code_clear_language_mode() mode.disable("user.gdb")	dictation mode
mode.disable("sleep") mode.disable("dictation") mode.enable("command")	command mode

mouse grid

Output	Say
app.notify("please use the voice command 'mouse grid' instead of 'm grid'") user.grid_select_screen(1) user.grid_activate()	M grid

mouse grid always

Output	Say
grid_select_screen(1) user.grid_activate()	mouse grid
grid_place_window() user.grid_activate()	grid win
grid_activate() user.grid_narrow_list(number_key_list)	grid <user.number_key>+
grid_select_screen(number or 1) user.grid_activate()	grid screen [<number>]

mouse grid open

Output	Say
grid_narrow(number_key)	<user.number_key>
grid_close()	grid off
grid_reset()	grid reset
grid_go_back()	grid back

draft global

Output	Say

# Do this toggle so we can have focus when saying 'draft show' 	
user.draft_hide()
user.draft_show()
	draft show

# Do this toggle so we can have focus when saying 'draft show' 	
user.draft_hide()
user.draft_show()
user.draft_named_move(draft_window_position)
	draft show <user.draft_window_position>

# Do this toggle so we can have focus when saying 'draft show' 	
user.draft_hide()
user.draft_show()
user.draft_resize(600, 200)
	draft show small

# Do this toggle so we can have focus when saying 'draft show' 	
user.draft_hide()
user.draft_show()
user.draft_resize(800, 500)
	draft show large

draft_show("")	draft empty
text = edit.selected_text() key(backspace) user.draft_show(text)	draft edit
edit.select_all() text = edit.selected_text() key(backspace) user.draft_show(text)	draft edit all

find and replace

Output	Say
find("")	hunt this
find(text)	hunt this <user.text>
find_everywhere("")	hunt all
find_everywhere(text)	hunt all <user.text>
find_toggle_match_by_case()	hunt case
find_toggle_match_by_word()	hunt word
find_toggle_match_by_regex()	hunt expression
find_next()	hunt next
find_previous()	hunt previous
replace(text or "")	replace this [<user.text>]
replace_everywhere("")	replace all
replace_everywhere(text)	replace <user.text> all
replace_confirm()	replace confirm that
replace_confirm_all()	replace confirm all
select_previous_occurrence(text) sleep(100ms) edit.delete()	clear last <user.text> [over]
select_next_occurrence(text) sleep(100ms) edit.delete()	clear next <user.text> [over]
select_previous_occurrence(clip.text()) edit.delete()	clear last clip
select_next_occurrence(clip.text()) sleep(100ms) edit.delete()	clear next clip
select_previous_occurrence(text) sleep(100ms) code.toggle_comment()	comment last <user.text> [over]
select_previous_occurrence(clip.text()) sleep(100ms) code.toggle_comment()	comment last clip
select_next_occurrence(text) sleep(100ms) code.toggle_comment()	comment next <user.text> [over]
select_next_occurrence(clip.text()) sleep(100ms) code.toggle_comment()	comment next clip
select_previous_occurrence(text) sleep(100ms) edit.right()	go last <user.text> [over]
select_previous_occurrence(clip.text()) sleep(100ms) edit.right()	go last clip

homophones

Output	Say
homophones_show(homophones_canonical)	phones <user.homophones_canonical>
homophones_show_selection()	phones that
homophones_force_show(homophones_canonical)	phones force <user.homophones_canonical>
homophones_force_show_selection()	phones force
homophones_hide()	phones hide
edit.select_word() user.homophones_show_selection()	phones word
n = ordinals or 1 user.words_left(n - 1) edit.extend_word_left() user.homophones_show_selection()	phones [<user.ordinals>] word left
n = ordinals or 1 user.words_right(n - 1) edit.extend_word_right() user.homophones_show_selection()	phones [<user.ordinals>] word right

homophones open

Output	Say
result = user.homophones_select(number_small) insert(result) user.homophones_hide()	choose <number_small>
result = user.homophones_select(number_small) insert(user.formatted_text(result, formatters)) user.homophones_hide()	choose <user.formatters> <number_small>

line commands

Output	Say
edit.line_end()	lend
edit.line_start()	bend
edit.jump_line(number)	go <number>
edit.jump_line(number) edit.line_end()	go <number> end
select_range(number, number) code.toggle_comment()	comment [line] <number>
select_range(number_1, number_2) code.toggle_comment()	comment <number> until <number>
edit.jump_line(number) user.select_range(number, number) edit.delete()	clear [line] <number>
select_range(number_1, number_2) edit.delete()	clear <number> until <number>
select_range(number, number) edit.copy()	copy [line] <number>
select_range(number_1, number_2) edit.copy()	copy <number> until <number>
select_range(number, number) edit.cut()	cut [line] <number>
select_range(number_1, number_2) edit.cut()	cut [line] <number> until <number>
select_range(number_1, number_2) edit.paste()	(paste | replace) <number> until <number>
select_range(number, number)	(select | cell | sell) [line] <number>
select_range(number_1, number_2)	(select | cell | sell) <number> until <number>
edit.indent_more()	tab that
edit.jump_line(number) edit.indent_more()	tab [line] <number>
select_range(number_1, number_2) edit.indent_more()	tab <number> until <number>
edit.indent_less()	retab that
select_range(number, number)	retab [line] <number>
edit.indent_less()	

numbers

Output	Say
"{number_string}"	<user.number_string>

symbols

Output	Say
"?"	question [mark]
"_"	(downscore | underscore)
"–"	double dash
"{"	(bracket | brack | left bracket)
"}"	(rbrack | are bracket | right bracket)
"""	triple quote
insert("```")	(triple grave | triple back tick | gravy)
".."	(dot dot | dotdot)
"..."	ellipses
", "	(comma and | spamma)
"+"	plus
"->"	arrow
"=>"	dub arrow
"\\n"	new line
"\\r"	carriage return
"\\r\\n"	line feed
'""' key(left)	empty dubstring
'\\"\\"' key(left) key(left)	empty escaped (dubstring|dub quotes)
"" key(left)	empty string
"\\'\\'" key(left) key(left)	empty escaped string
insert("()") key(left)	(inside parens | args)
insert("[]") key(left)	inside (squares | square brackets | list)
insert("{}") key(left)	inside (bracket | braces)
insert("%%") key(left)	inside percent
insert("") key(left)	inside (quotes | string)
insert('""') key(left)	inside (double quotes | dubquotes)
insert("``") key(left)	inside (graves | back ticks)
text = edit.selected_text() user.paste("<{text}>")	angle that
text = edit.selected_text() user.paste("[{text}]")	(square | square bracket) that
text = edit.selected_text() user.paste("{{{text}}}")	(bracket | brace) that
text = edit.selected_text() user.paste("({text})")	(parens | args) that
text = edit.selected_text() user.paste("%{text}%")	percent that
text = edit.selected_text() user.paste("{text}")	quote that

edge

Output	Say
key(ctrl-shift-u)	read aloud

powertoys

Output	Say
key(win-space)	(search tool)

talon control

Output	Say
mouse_wake() user.history_enable() user.talon_mode()	welcome back
speech.disable()	drowsy [<phrase>]
speech.enable()	talon wake

toolkit

Output	Say
hud_toolkit_options()	toolkit options
hud_toolkit_debug_options()	toolkit debugging
hud_toolkit_scope()	toolkit scope
hud_toolkit_speech()	toolkit speech
hud_toolkit_lists()	toolkit lists
hud_toolkit_history()	toolkit history
show_microphone_options()	toolkit microphones
hud_show_documentation()	toolkit documentation
hud_show_walkthroughs()	toolkit walk throughs
hud_add_focus_toggle()	toolkit focus [show]
hud_remove_focus_toggle()	toolkit focus hide

widget quick choices

Output	Say
hud_activate_choice(talon_hud_quick_choices)	{user.talon_hud_quick_choices}

batch

Output	Say
"exit /B 1\n"	soft exit
"exit 1\n"	hard exit
"echo "	echo
"@echo off\n"	echo off
"call "	call
"call cmd \\c "	call shell
"if errorlevel 1 "	if error
"goto "	go to
"SETLOCAL EnableDelayedExpansion\n"	delayed expansion
"%{number_small}"	arg <number_small>

javascript

Output	Say
" === "	(op | is) strict equal
" !== "	(op | is) strict not equal
"const "	state const
"let "	state let
"var "	state var
"export "	state export
"async "	state async
"await "	state await
insert(".map()") key(left)	state map
insert(".filter()") key(left)	state filter
insert(".reduce()") key(left)	state reduce
"..."	state spread
insert(' from ""') key("left")	from import

markdown

Output	Say
"# "	level one
"## "	level two
"### "	level three
"#### "	level four
"##### "	level five
"###### "	level six
"```{markdown_code_block_language}" key(enter:2) "```" key(up)	{user.markdown_code_block_language} block

python

Output	Say
"__init__"	dunder in it
"def "	state (def | deaf | deft)
"try:\n"	state try
"except "	state except
"raise "	state raise
"self."	self taught
"pytest"	pie test
"pass"	state past
insert_cursor("raise {python_exception}([])")	raise {user.python_exception}
"except {python_exception}:"	except {user.python_exception}
code_comment_documentation()	dock string
insert("{python_docstring_fields}") edit.left()	dock {user.python_docstring_fields}
insert_cursor(":type [|]: {code_type}")	dock type {user.code_type}
insert_cursor(":rtype [|]: {code_type}")	dock returns type {user.code_type}
code_toggle_libraries()	toggle imports
code_insert_library(code_libraries, "") key(end enter)	import <user.code_libraries>
insert('from import ') key('left') edit.word_left() key('space') edit.left()	from import

comment documentation

Output	Say
code_comment_documentation()	dock comment

comment line

Output	Say
code_comment_line_prefix()	comment
#todo: this should probably be a single function once #.talon supports implementing actions with parameters? edit.line_start() user.code_comment_line_prefix() #adds comment to the start of the line	comment line
#todo: this should probably be a single function once #.talon supports implementing actions with parameters? edit.line_start() user.code_comment_line_prefix() insert(user.text) insert(" ")	comment line <user.text> over
#todo: this should probably be a single function once #.talon supports implementing actions with parameters? user.code_comment_line_prefix() insert(user.text)	comment <user.text> over
#todo: this should probably be a single function once #.talon supports implementing actions with parameters? user.code_comment_line_prefix() insert(user.text)	comment <user.text>
#todo: this should probably be a single function once #.talon supports implementing actions with parameters? edit.line_end() user.code_com-	(line | inline) comment <user.text> over

data bool

Output	Say
code_insert_true()	state true
code_insert_false()	state false

data null

Output	Say
code_insert_null()	state (no | nil | null)
code_insert_is_not_null()	is not (none|null)
code_insert_is_null()	is (none|null)

functions

Output	Say
code_default_function(text)	funky <user.text>
code_protected_function(text)	pro funky <user.text>
code_public_function(text)	pub funky <user.text>
code_private_static_function(text)	static funky <user.text>
code_protected_static_function(text)	pro static funky <user.text>
code_public_static_function(text)	pub static funky <user.text>
code_insert_type_annotation(code_type)	is type {user.code_type}
code_insert_return_type(code_type)	returns [type] {user.code_type}
insert("{code_type}")	type {user.code_type}

functions gui

Output	Say
code_toggle_functions()	toggle funk
code_insert_function(code_functions, "")	funk <user.code_functions>
code_select_function(number - 1, "")	funk cell <number>
code_insert_function(code_functions, edit.selected_text())	funk wrap <user.code_functions>
code_select_function(number - 1, edit.selected_text())	funk wrap <number>

functions gui open

Output	Say
code_toggle_functions()	toggle funk

imperative

Output	Say
code_block()	block
code_state_if()	state if
code_state_else_if()	state else if
code_state_else()	state else
code_state_while()	state while
code_state_for()	state for
code_state_for_each()	state for in
code_state_switch()	state switch
code_state_case()	state case
code_state_do()	state do
code_state_go_to()	state goto
code_state_return()	state return
code_break()	state break
code_next()	state next

libraries

Output	Say
code_import()	state import

library gui open

Output	Say
code_toggle_libraries()	toggle libraries close

object oriented

Output	Say
code_self() user.code_operator_object_accessor()	self dot
code_self()	state self
code_define_class()	state class

operators array

Output	Say
code_operator_subscript()	op subscript

operators assignment

Output	Say
code_operator_assignment()	op (equals | assign)
code_operator_subtraction_assignment()	op (minus | subtract) equals
code_operator_addition_assignment()	op (plus | add) equals
code_operator_multiplication_assignment()	op (times | multiply) equals
code_operator_division_assignment()	op divide equals
code_operator_modulo_assignment()	op mod equals
code_operator_increment()	[op] increment
code_operator_bitwise_exclusive_or_equals()	(op | logical | bitwise) (ex | exclusive) or equals
code_operator_bitwise_left_shift_equals()	[(op | logical | bitwise)] (left shift | shift left) equals
code_operator_bitwise_right_shift_equals()	[(op | logical | bitwise)] (left right | shift right) equals

operators bitwise

Output	Say
code_operator_bitwise_and()	[op] bitwise and
code_operator_bitwise_or()	[op] bitwise or
code_operator_bitwise_exclusive_or()	(op | logical | bitwise) (ex | exclusive) or
code_operator_bitwise_left_shift()	(op | logical | bitwise) (left shift | shift left)
code_operator_bitwise_right_shift()	(op | logical | bitwise) (right shift | shift right)

operators lambda

Output	Say
code_operator_lambda()	op lambda

operators math

Output	Say
code_operator_subtraction()	op (minus | subtract)
code_operator_addition()	op (plus | add)
code_operator_multiplication()	op (times | multiply)
code_operator_division()	op divide
code_operator_modulo()	op mod
code_operator_exponent()	(op (power | exponent) | to the power [of])
code_operator_equal()	(op | is) equal
code_operator_not_equal()	(op | is) not equal
code_operator_greater_than()	(op | is) (greater | more)
code_operator_less_than()	(op | is) (less | below) [than]
code_operator_greater_than_or_equal_to()	(op | is) greater [than] or equal
code_operator_less_than_or_equal_to()	(op | is) less [than] or equal
code_operator_and()	(op | logical) and
code_operator_or()	(op | logical) or
" : "	(op | pad) colon

operators pointer

Output	Say
code_operator_indirection()	op dereference
code_operator_address_of()	op address of
code_operator_structure_dereference()	op arrow