Package ‘rebus.unicode’

October 14, 2022

Type Package

Title Unicode Extensions for the 'rebus' Package

Version 0.0-2

Date 2017-01-02

Author Richard Cotton [aut, cre]

Maintainer Richard Cotton <richierocks@gmail.com>

Description Build regular expressions piece by piece using human readable code. This package contains Unicode functionality, and is primarily intended to be used by package developers.

Depends R (>= 3.1.0)

Imports rebus.base (>= 0.0-2)

Suggests stringi

License Unlimited

LazyLoad yes

LazyData yes

Acknowledgments Development of this package was partially funded by the Proteomics Core at Weill Cornell Medical College in Qatar <http://qatar-weill.cornell.edu>. The Core is supported by 'Biomedical Research Program' funds, a program funded by Qatar Foundation.

Collate 'imports.R' 'unicode-classes.R' 'unicode-constants.R' 'unicode-general-category-classes.R' 'unicode-general-category-constants.R' 'unicode-operators.R'

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-01-03 07:42:23

R topics documented:

`ugc_cased_letter` *Unicode General Categories*

Description

Match a Unicode General Category.

Usage

```
ugc_cased_letter(lo, hi, char_class = TRUE)

ugc_close_punctuation(lo, hi, char_class = TRUE)

ugc_connector_punctuation(lo, hi, char_class = TRUE)

ugc_control(lo, hi, char_class = TRUE)

ugc_currency_symbol(lo, hi, char_class = TRUE)

ugc_dash_punctuation(lo, hi, char_class = TRUE)

ugc_decimal_number(lo, hi, char_class = TRUE)

ugc_enclosing_mark(lo, hi, char_class = TRUE)

ugc_final_punctuation(lo, hi, char_class = TRUE)

ugc_format_control(lo, hi, char_class = TRUE)

ugc_initial_punctuation(lo, hi, char_class = TRUE)

ugc_letter(lo, hi, char_class = TRUE)

ugc_letter_number(lo, hi, char_class = TRUE)

ugc_line_separator(lo, hi, char_class = TRUE)

ugc_lowercase_letter(lo, hi, char_class = TRUE)
```

```
ugc_mark(lo, hi, char_class = TRUE)

ugc_math_symbol(lo, hi, char_class = TRUE)

ugc_modifier_letter(lo, hi, char_class = TRUE)

ugc_modifier_symbol(lo, hi, char_class = TRUE)

ugc_nonspacing_mark(lo, hi, char_class = TRUE)

ugc_number(lo, hi, char_class = TRUE)

ugc_open_punctuation(lo, hi, char_class = TRUE)

ugc_other(lo, hi, char_class = TRUE)

ugc_other_letter(lo, hi, char_class = TRUE)

ugc_other_number(lo, hi, char_class = TRUE)

ugc_other_punctuation(lo, hi, char_class = TRUE)

ugc_other_symbol(lo, hi, char_class = TRUE)

ugc_paragraph_separator(lo, hi, char_class = TRUE)

ugc_private_use_control(lo, hi, char_class = TRUE)

ugc_punctuation(lo, hi, char_class = TRUE)

ugc_separator(lo, hi, char_class = TRUE)

ugc_space_separator(lo, hi, char_class = TRUE)

ugc_spacing_mark(lo, hi, char_class = TRUE)

ugc_surrogate_control(lo, hi, char_class = TRUE)

ugc_symbol(lo, hi, char_class = TRUE)

ugc_titlecase_letter(lo, hi, char_class = TRUE)

ugc_unassigned_control(lo, hi, char_class = TRUE)

ugc_uppercase_letter(lo, hi, char_class = TRUE)

UGC_UPPERCASE_LETTER
```

`UGC_LOWERCASE_LETTER`

`UGC_TITLECASE_LETTER`

`UGC_CASED_LETTER`

`UGC_MODIFIER_LETTER`

`UGC_OTHER_LETTER`

`UGC_LETTER`

`UGC_NONSPACING_MARK`

`UGC_SPACING_MARK`

`UGC_ENCLOSING_MARK`

`UGC_MARK`

`UGC_DECIMAL_NUMBER`

`UGC_LETTER_NUMBER`

`UGC_OTHER_NUMBER`

`UGC_NUMBER`

`UGC_CONNECTOR_PUNCTUATION`

`UGC_DASH_PUNCTUATION`

`UGC_OPEN_PUNCTUATION`

`UGC_CLOSE_PUNCTUATION`

`UGC_INITIAL_PUNCTUATION`

`UGC_FINAL_PUNCTUATION`

`UGC_OTHER_PUNCTUATION`

`UGC_PUNCTUATION`

`UGC_MATH_SYMBOL`

`UGC_CURRENCY_SYMBOL`

`UGC_MODIFIER_SYMBOL`

`UGC_OTHER_SYMBOL`

`UGC_SYMBOL`

`UGC_SPACE_SEPARATOR`

`UGC_LINE_SEPARATOR`

`UGC_PARAGRAPH_SEPARATOR`

`UGC_SEPARATOR`

`UGC_CONTROL`

`UGC_FORMAT_CONTROL`

`UGC_SURROGATE_CONTROL`

`UGC_PRIVATE_USE_CONTROL`

`UGC_UNASSIGNED_CONTROL`

`UGC_OTHER`

Arguments

`lo` A non-negative integer. Minimum number of repeats, when grouped.

`hi` positive integer. Maximum number of repeats, when grouped.

`char_class` `TRUE` or `FALSE`. Should the values be wrapped into a character class?

Format

An object of class `regex` (inherits from `character`) of length 1.

Value

A character vector representing part or all of a regular expression.

References

Table 12 of the Unicode Standard Annex #44 defines the Unicode General Categories. http://www.unicode.org/reports/tr44

You can see which characters are contained in a category by visiting, e.g., http://www.fileformat.info/info/unicode/category/Nd/list.htm

See Also

unicode_property, Unicode

Examples

```
# Classes
ugc_lowercase_letter()
ugc_decimal_number()
ugc_paragraph_separator()
ugc_currency_symbol()

# With repetition
ugc_nonspacing_mark(3, 6)
ugc_separator(1, Inf)
ugc_dash_punctuation(0, Inf)

# Without a class wrapper
ugc_titlecase_letter(char_class = FALSE)

# Constants
UGC_UPPERCASE_LETTER
UGC_LETTER_NUMBER
UGC_MATH_SYMBOL
UGC_FORMAT_CONTROL

## Not run:
# All the Unicode general categories.
# Not run, since it generates lots of output
ls("package:rebus.unicode", pattern = "^ugc")

## End(Not run)

# Usage
library(rebus.base)
x <- "I exchanged $1000 for \u20ac665.41 and \u00a3243.13."
(rx <- capture(ugc_currency_symbol()) %R%
  capture(
    ugc_decimal_number(1, Inf) %R%
    optional(group("." %R% ugc_decimal_number(2)))
  )
)
stringi::stri_match_all_regex(x, rx)
```

Unicode *Unicode classes*

Description

Match ranges of unicode characters. In particular, you can match characters from a particular language.

Usage

```
armenian(lo, hi, char_class = TRUE)

armenian_ligatures(lo, hi, char_class = TRUE)

caucasian_albanian(lo, hi, char_class = TRUE)

cypriot_syllabary(lo, hi, char_class = TRUE)

cyrillic(lo, hi, char_class = TRUE)

cyrillic_supplement(lo, hi, char_class = TRUE)

cyrillic_extended_a(lo, hi, char_class = TRUE)

cyrillic_extended_b(lo, hi, char_class = TRUE)

elbasan(lo, hi, char_class = TRUE)

georgian(lo, hi, char_class = TRUE)

georgian_supplement(lo, hi, char_class = TRUE)

glagolitic(lo, hi, char_class = TRUE)

gothic(lo, hi, char_class = TRUE)

greek_and_coptic(lo, hi, char_class = TRUE)

greek_extended(lo, hi, char_class = TRUE)

latin(lo, hi, char_class = TRUE)

latin_1_supplement(lo, hi, char_class = TRUE)

latin_extended_a(lo, hi, char_class = TRUE)

latin_extended_b(lo, hi, char_class = TRUE)

latin_extended_c(lo, hi, char_class = TRUE)

latin_extended_d(lo, hi, char_class = TRUE)

latin_extended_e(lo, hi, char_class = TRUE)

latin_extended_additional(lo, hi, char_class = TRUE)

latin_ligatures(lo, hi, char_class = TRUE)
```

```
linear_a(lo, hi, char_class = TRUE)

linear_b_syllabary(lo, hi, char_class = TRUE)

linear_b_ideograms(lo, hi, char_class = TRUE)

ogham(lo, hi, char_class = TRUE)

old_italic(lo, hi, char_class = TRUE)

old_permic(lo, hi, char_class = TRUE)

phaistos_disc(lo, hi, char_class = TRUE)

runic(lo, hi, char_class = TRUE)

shavian(lo, hi, char_class = TRUE)

duployan(lo, hi, char_class = TRUE)

shorthand_format_controls(lo, hi, char_class = TRUE)

ipa_extensions(lo, hi, char_class = TRUE)

phonetic_extensions(lo, hi, char_class = TRUE)

phonetic_extensions_supplement(lo, hi, char_class = TRUE)

modifier_tone_letters(lo, hi, char_class = TRUE)

spacing_modifier_letters(lo, hi, char_class = TRUE)

superscripts_and_subscripts(lo, hi, char_class = TRUE)

combining_diacritic_marks(lo, hi, char_class = TRUE)

combining_diacritic_supplement(lo, hi, char_class = TRUE)

combining_diacritic_extended(lo, hi, char_class = TRUE)

combining_half_marks(lo, hi, char_class = TRUE)

bamun(lo, hi, char_class = TRUE)

bamun_supplement(lo, hi, char_class = TRUE)

bassa_vah(lo, hi, char_class = TRUE)
```

```
coptic(lo, hi, char_class = TRUE)

coptic_epact_numbers(lo, hi, char_class = TRUE)

egyptian_hieroglyphs(lo, hi, char_class = TRUE)

ethiopic(lo, hi, char_class = TRUE)

ethiopic_supplement(lo, hi, char_class = TRUE)

ethiopic_extended(lo, hi, char_class = TRUE)

ethiopic_extended_a(lo, hi, char_class = TRUE)

mende_kikakui(lo, hi, char_class = TRUE)

meroitic_cursive(lo, hi, char_class = TRUE)

meroitic_hieroglyphs(lo, hi, char_class = TRUE)

nko(lo, hi, char_class = TRUE)

osmanya(lo, hi, char_class = TRUE)

tifinagh(lo, hi, char_class = TRUE)

vai(lo, hi, char_class = TRUE)

arabic(lo, hi, char_class = TRUE)

arabic_supplement(lo, hi, char_class = TRUE)

arabic_extended_a(lo, hi, char_class = TRUE)

arabic_presentation_forms_a(lo, hi, char_class = TRUE)

arabic_presentation_forms_b(lo, hi, char_class = TRUE)

imperial_aramaic(lo, hi, char_class = TRUE)

avestan(lo, hi, char_class = TRUE)

carian(lo, hi, char_class = TRUE)

cuneiform(lo, hi, char_class = TRUE)

cuneiform_numbers_and_punctuation(lo, hi, char_class = TRUE)
```

```
old_persian(lo, hi, char_class = TRUE)

ugaritic(lo, hi, char_class = TRUE)

hebrew(lo, hi, char_class = TRUE)

lycian(lo, hi, char_class = TRUE)

lydian(lo, hi, char_class = TRUE)

mandaic(lo, hi, char_class = TRUE)

nabataean(lo, hi, char_class = TRUE)

old_north_arabian(lo, hi, char_class = TRUE)

old_south_arabian(lo, hi, char_class = TRUE)

pahlavi_inscriptional(lo, hi, char_class = TRUE)

pahlavi_psalter(lo, hi, char_class = TRUE)

palmyrene(lo, hi, char_class = TRUE)

phoenician(lo, hi, char_class = TRUE)

samaritan(lo, hi, char_class = TRUE)

syriac(lo, hi, char_class = TRUE)

manichaean(lo, hi, char_class = TRUE)

mongolian(lo, hi, char_class = TRUE)

old_turkic(lo, hi, char_class = TRUE)

phags_pa(lo, hi, char_class = TRUE)

tibetan(lo, hi, char_class = TRUE)

bengali_and_assamese(lo, hi, char_class = TRUE)

brahmi(lo, hi, char_class = TRUE)

chakma(lo, hi, char_class = TRUE)

devanagari(lo, hi, char_class = TRUE)
```

```
devanagari_extended(lo, hi, char_class = TRUE)

grantha(lo, hi, char_class = TRUE)

gujarati(lo, hi, char_class = TRUE)

gurmukhi(lo, hi, char_class = TRUE)

kaithi(lo, hi, char_class = TRUE)

kannada(lo, hi, char_class = TRUE)

kharoshthi(lo, hi, char_class = TRUE)

khojki(lo, hi, char_class = TRUE)

khudawadi(lo, hi, char_class = TRUE)

lepcha(lo, hi, char_class = TRUE)

limbu(lo, hi, char_class = TRUE)

mahajani(lo, hi, char_class = TRUE)

malayalam(lo, hi, char_class = TRUE)

meetei_mayek(lo, hi, char_class = TRUE)

meetei_mayek_extensions(lo, hi, char_class = TRUE)

modi(lo, hi, char_class = TRUE)

mro(lo, hi, char_class = TRUE)

ol_chiki(lo, hi, char_class = TRUE)

oriya(lo, hi, char_class = TRUE)

saurashtra(lo, hi, char_class = TRUE)

sharada(lo, hi, char_class = TRUE)

siddham(lo, hi, char_class = TRUE)

sinhala(lo, hi, char_class = TRUE)

sinhala_archaic_numbers(lo, hi, char_class = TRUE)
```

```
sora_sompeng(lo, hi, char_class = TRUE)

syloti_nagri(lo, hi, char_class = TRUE)

takri(lo, hi, char_class = TRUE)

tamil(lo, hi, char_class = TRUE)

telugu(lo, hi, char_class = TRUE)

thaana(lo, hi, char_class = TRUE)

tirhuta(lo, hi, char_class = TRUE)

vedic_extensions(lo, hi, char_class = TRUE)

warang_citi(lo, hi, char_class = TRUE)

cham(lo, hi, char_class = TRUE)

kayah_li(lo, hi, char_class = TRUE)

khmer(lo, hi, char_class = TRUE)

khmer_symbols(lo, hi, char_class = TRUE)

lao(lo, hi, char_class = TRUE)

myanmar(lo, hi, char_class = TRUE)

myanmar_extended_a(lo, hi, char_class = TRUE)

myanmar_extended_b(lo, hi, char_class = TRUE)

new_tai_lue(lo, hi, char_class = TRUE)

pahawh_hmong(lo, hi, char_class = TRUE)

pau_cin_hau(lo, hi, char_class = TRUE)

tai_le(lo, hi, char_class = TRUE)

tai_tham(lo, hi, char_class = TRUE)

tai_viet(lo, hi, char_class = TRUE)

thai(lo, hi, char_class = TRUE)
```

```
balinese(lo, hi, char_class = TRUE)

batak(lo, hi, char_class = TRUE)

buginese(lo, hi, char_class = TRUE)

buhid(lo, hi, char_class = TRUE)

hanunoo(lo, hi, char_class = TRUE)

javanese(lo, hi, char_class = TRUE)

rejang(lo, hi, char_class = TRUE)

sundanese(lo, hi, char_class = TRUE)

sundanese_supplement(lo, hi, char_class = TRUE)

tagalog(lo, hi, char_class = TRUE)

tagbanwa(lo, hi, char_class = TRUE)

bopomofo(lo, hi, char_class = TRUE)

bopomofo_extended(lo, hi, char_class = TRUE)

cjk_unified_ideographs(lo, hi, char_class = TRUE)

cjk_unified_ideographs_extension_a(lo, hi, char_class = TRUE)

cjk_unified_ideographs_extension_b(lo, hi, char_class = TRUE)

cjk_unified_ideographs_extension_c(lo, hi, char_class = TRUE)

cjk_unified_ideographs_extension_d(lo, hi, char_class = TRUE)

cjk_compatibility_ideographs(lo, hi, char_class = TRUE)

cjk_compatibility_ideographs_supplement(lo, hi, char_class = TRUE)

kangxi_radicals(lo, hi, char_class = TRUE)

kangxi_radicals_supplement(lo, hi, char_class = TRUE)

cjk_strokes(lo, hi, char_class = TRUE)

cjk_ideographic_description_characters(lo, hi, char_class = TRUE)
```

```
hangul_jamo(lo, hi, char_class = TRUE)

hangul_jamo_extended_a(lo, hi, char_class = TRUE)

hangul_jamo_extended_b(lo, hi, char_class = TRUE)

hangul_compatibility_jamo(lo, hi, char_class = TRUE)

hangul_syllables(lo, hi, char_class = TRUE)

hiragana(lo, hi, char_class = TRUE)

katakana(lo, hi, char_class = TRUE)

katakana_phonetic_extensions(lo, hi, char_class = TRUE)

kana_supplement(lo, hi, char_class = TRUE)

kanbun(lo, hi, char_class = TRUE)

lisu(lo, hi, char_class = TRUE)

miao(lo, hi, char_class = TRUE)

yi_syllables(lo, hi, char_class = TRUE)

yi_radicals(lo, hi, char_class = TRUE)

cherokee(lo, hi, char_class = TRUE)

deseret(lo, hi, char_class = TRUE)

unified_canadian_aboriginal_syllabics(lo, hi, char_class = TRUE)

unified_canadian_aboriginal_syllabics_extended(lo, hi, char_class = TRUE)

alphabetic_presentation_forms(lo, hi, char_class = TRUE)

halfwidth_and_fullwidth_forms(lo, hi, char_class = TRUE)

general_punctuation(lo, hi, char_class = TRUE)

latin_1_punctuation(lo, hi, char_class = TRUE)

small_form_variants(lo, hi, char_class = TRUE)

supplemental_punctuation(lo, hi, char_class = TRUE)
```

```
cjk_symbols_and_punctuation(lo, hi, char_class = TRUE)

cjk_compatibility_forms(lo, hi, char_class = TRUE)

fullwidth_ascii_punctuation(lo, hi, char_class = TRUE)

vertical_forms(lo, hi, char_class = TRUE)

letterlike_symbols(lo, hi, char_class = TRUE)

ancient_symbols(lo, hi, char_class = TRUE)

mathematical_alphanumeric_symbols(lo, hi, char_class = TRUE)

arabic_mathematical_alphanumeric_symbols(lo, hi, char_class = TRUE)

enclosed_alphanumerics(lo, hi, char_class = TRUE)

enclosed_alphanumeric_supplement(lo, hi, char_class = TRUE)

enclosed_cjk_letters_and_months(lo, hi, char_class = TRUE)

enclosed_ideographic_supplement(lo, hi, char_class = TRUE)

cjk_compatibility(lo, hi, char_class = TRUE)

miscellaneous_technical(lo, hi, char_class = TRUE)

control_pictures(lo, hi, char_class = TRUE)

optical_character_recognition(lo, hi, char_class = TRUE)

combining_diacritic_marks_for_symbols(lo, hi, char_class = TRUE)

aegean_numbers(lo, hi, char_class = TRUE)

ancient_greek_numbers(lo, hi, char_class = TRUE)

fullwidth_ascii_digits(lo, hi, char_class = TRUE)

common_indic_number_forms(lo, hi, char_class = TRUE)

coptic_epact_numbers(lo, hi, char_class = TRUE)

counting_rod_numerals(lo, hi, char_class = TRUE)

number_forms(lo, hi, char_class = TRUE)
```

```
rumi_numeral_symbols(lo, hi, char_class = TRUE)

sinhala_archaic_numbers(lo, hi, char_class = TRUE)

math_arrows(lo, hi, char_class = TRUE)

supplemental_arrows_a(lo, hi, char_class = TRUE)

supplemental_arrows_a(lo, hi, char_class = TRUE)

supplemental_arrows_a(lo, hi, char_class = TRUE)

additional_arrows(lo, hi, char_class = TRUE)

supplemental_mathematical_operators(lo, hi, char_class = TRUE)

miscellaneous_mathematical_symbols_a(lo, hi, char_class = TRUE)

miscellaneous_mathematical_symbols_b(lo, hi, char_class = TRUE)

floors_and_ceilings(lo, hi, char_class = TRUE)

invisible_operators(lo, hi, char_class = TRUE)

geometric_shapes(lo, hi, char_class = TRUE)

box_drawing(lo, hi, char_class = TRUE)

block_elements(lo, hi, char_class = TRUE)

geometric_shapes_extended(lo, hi, char_class = TRUE)

alchemical_symbols(lo, hi, char_class = TRUE)

braille_patterns(lo, hi, char_class = TRUE)

currency_symbols(lo, hi, char_class = TRUE)

dingbats(lo, hi, char_class = TRUE)

ornamental_dingbats(lo, hi, char_class = TRUE)

emoticons(lo, hi, char_class = TRUE)

chess_checkers_draughts(lo, hi, char_class = TRUE)

domino_tiles(lo, hi, char_class = TRUE)
```

```
japanese_chess(lo, hi, char_class = TRUE)

mahjong_tiles(lo, hi, char_class = TRUE)

playing_cards(lo, hi, char_class = TRUE)

card_suits(lo, hi, char_class = TRUE)

miscellaneous_symbols_and_pictographs(lo, hi, char_class = TRUE)

musical_symbols(lo, hi, char_class = TRUE)

ancient_greek_musical_notation(lo, hi, char_class = TRUE)

byzantine_musical_symbols(lo, hi, char_class = TRUE)

transport_and_map_symbols(lo, hi, char_class = TRUE)

yijing_mono_di_and_trigrams(lo, hi, char_class = TRUE)

yijing_hexagram_symbols(lo, hi, char_class = TRUE)

tai_xuan_jing_symbols(lo, hi, char_class = TRUE)

specials(lo, hi, char_class = TRUE)

tags(lo, hi, char_class = TRUE)

variation_selectors(lo, hi, char_class = TRUE)

variation_selectors_supplement(lo, hi, char_class = TRUE)

private_use_area(lo, hi, char_class = TRUE)

supplementary_private_use_area_a(lo, hi, char_class = TRUE)

supplementary_private_use_area_b(lo, hi, char_class = TRUE)

ARMENIAN

ARMENIAN_LIGATURES

CAUCASIAN_ALBANIAN

CYPRIOT_SYLLABARY

CYRILLIC
```

CYRILLIC_SUPPLEMENT

CYRILLIC_EXTENDED_A

CYRILLIC_EXTENDED_B

ELBASAN

GEORGIAN

GEORGIAN_SUPPLEMENT

GLAGOLITIC

GOTHIC

GREEK_AND_COPTIC

GREEK_EXTENDED

LATIN

LATIN_1_SUPPLEMENT

LATIN_EXTENDED_A

LATIN_EXTENDED_B

LATIN_EXTENDED_C

LATIN_EXTENDED_D

LATIN_EXTENDED_E

LATIN_EXTENDED_ADDITIONAL

LATIN_LIGATURES

LINEAR_A

LINEAR_B_SYLLABARY

LINEAR_B_IDEOGRAMS

OGHAM

OLD_ITALIC

```
OLD_PERMIC

PHAISTOS_DISC

RUNIC

SHAVIAN

DUPLOYAN

SHORTHAND_FORMAT_CONTROLS

IPA_EXTENSIONS

PHONETIC_EXTENSIONS

PHONETIC_EXTENSIONS_SUPPLEMENT

MODIFIER_TONE_LETTERS

SPACING_MODIFIER_LETTERS

SUPERSCRIPTS_AND_SUBSCRIPTS

COMBINING_DIACRITIC_MARKS

COMBINING_DIACRITIC_SUPPLEMENT

COMBINING_DIACRITIC_EXTENDED

COMBINING_HALF_MARKS

BAMUN

BAMUN_SUPPLEMENT

BASSA_VAH

COPTIC

COPTIC_EPACT_NUMBERS

EGYPTIAN_HIEROGLYPHS

ETHIOPIC

ETHIOPIC_SUPPLEMENT
```

```
ETHIOPIC_EXTENDED

ETHIOPIC_EXTENDED_A

MENDE_KIKAKUI

MEROITIC_CURSIVE

MEROITIC_HIEROGLYPHS

NKO

OSMANYA

TIFINAGH

VAI

ARABIC

ARABIC_SUPPLEMENT

ARABIC_EXTENDED_A

ARABIC_PRESENTATION_FORMS_A

ARABIC_PRESENTATION_FORMS_B

IMPERIAL_ARAMAIC

AVESTAN

CARIAN

CUNEIFORM

CUNEIFORM_NUMBERS_AND_PUNCTUATION

OLD_PERSIAN

UGARITIC

HEBREW

LYCIAN

LYDIAN
```

MANDAIC

NABATAEAN

OLD_NORTH_ARABIAN

OLD_SOUTH_ARABIAN

PAHLAVI_INSCRIPTIONAL

PAHLAVI_PSALTER

PALMYRENE

PHOENICIAN

SAMARITAN

SYRIAC

MANICHAEAN

MONGOLIAN

OLD_TURKIC

PHAGS_PA

TIBETAN

BENGALI_AND_ASSAMESE

BRAHMI

CHAKMA

DEVANAGARI

DEVANAGARI_EXTENDED

GRANTHA

GUJARATI

GURMUKHI

KAITHI

KANNADA

KHAROSHTHI

KHOJKI

KHUDAWADI

LEPCHA

LIMBU

MAHAJANI

MALAYALAM

MEETEI_MAYEK

MEETEI_MAYEK_EXTENSIONS

MODI

MRO

OL_CHIKI

ORIYA

SAURASHTRA

SHARADA

SIDDHAM

SINHALA

SINHALA_ARCHAIC_NUMBERS

SORA_SOMPENG

SYLOTI_NAGRI

TAKRI

TAMIL

TELUGU

```
THAANA

TIRHUTA

VEDIC_EXTENSIONS

WARANG_CITI

CHAM

KAYAH_LI

KHMER

KHMER_SYMBOLS

LAO

MYANMAR

MYANMAR_EXTENDED_A

MYANMAR_EXTENDED_B

NEW_TAI_LUE

PAHAWH_HMONG

PAU_CIN_HAU

TAI_LE

TAI_THAM

TAI_VIET

THAI

BALINESE

BATAK

BUGINESE

BUHID

HANUNOO
```

JAVANESE

REJANG

SUNDANESE

SUNDANESE_SUPPLEMENT

TAGALOG

TAGBANWA

BOPOMOFO

BOPOMOFO_EXTENDED

CJK_UNIFIED_IDEOGRAPHS

CJK_UNIFIED_IDEOGRAPHS_EXTENSION_A

CJK_UNIFIED_IDEOGRAPHS_EXTENSION_B

CJK_UNIFIED_IDEOGRAPHS_EXTENSION_C

CJK_UNIFIED_IDEOGRAPHS_EXTENSION_D

CJK_COMPATIBILITY_IDEOGRAPHS

CJK_COMPATIBILITY_IDEOGRAPHS_SUPPLEMENT

KANGXI_RADICALS

KANGXI_RADICALS_SUPPLEMENT

CJK_STROKES

CJK_IDEOGRAPHIC_DESCRIPTION_CHARACTERS

HANGUL_JAMO

HANGUL_JAMO_EXTENDED_A

HANGUL_JAMO_EXTENDED_B

HANGUL_COMPATIBILITY_JAMO

HANGUL_SYLLABLES

```
HIRAGANA

KATAKANA

KATAKANA_PHONETIC_EXTENSIONS

KANA_SUPPLEMENT

KANBUN

LISU

MIAO

YI_SYLLABLES

YI_RADICALS

CHEROKEE

DESERET

UNIFIED_CANADIAN_ABORIGINAL_SYLLABICS

UNIFIED_CANADIAN_ABORIGINAL_SYLLABICS_EXTENDED

ALPHABETIC_PRESENTATION_FORMS

HALFWIDTH_AND_FULLWIDTH_FORMS

GENERAL_PUNCTUATION

LATIN_1_PUNCTUATION

SMALL_FORM_VARIANTS

SUPPLEMENTAL_PUNCTUATION

CJK_SYMBOLS_AND_PUNCTUATION

CJK_COMPATIBILITY_FORMS

FULLWIDTH_ASCII_PUNCTUATION

VERTICAL_FORMS

LETTERLIKE_SYMBOLS
```

ANCIENT_SYMBOLS

MATHEMATICAL_ALPHANUMERIC_SYMBOLS

ARABIC_MATHEMATICAL_ALPHANUMERIC_SYMBOLS

ENCLOSED_ALPHANUMERICS

ENCLOSED_ALPHANUMERIC_SUPPLEMENT

ENCLOSED_CJK_LETTERS_AND_MONTHS

ENCLOSED_IDEOGRAPHIC_SUPPLEMENT

CJK_COMPATIBILITY

MISCELLANEOUS_TECHNICAL

CONTROL_PICTURES

OPTICAL_CHARACTER_RECOGNITION

COMBINING_DIACRITIC_MARKS_FOR_SYMBOLS

AEGEAN_NUMBERS

ANCIENT_GREEK_NUMBERS

FULLWIDTH_ASCII_DIGITS

COMMON_INDIC_NUMBER_FORMS

COPTIC_EPACT_NUMBERS

COUNTING_ROD_NUMERALS

NUMBER_FORMS

RUMI_NUMERAL_SYMBOLS

SINHALA_ARCHAIC_NUMBERS

MATH_ARROWS

SUPPLEMENTAL_ARROWS_A

SUPPLEMENTAL_ARROWS_A

```
SUPPLEMENTAL_ARROWS_A

ADDITIONAL_ARROWS

SUPPLEMENTAL_MATHEMATICAL_OPERATORS

MISCELLANEOUS_MATHEMATICAL_SYMBOLS_A

MISCELLANEOUS_MATHEMATICAL_SYMBOLS_B

FLOORS_AND_CEILINGS

INVISIBLE_OPERATORS

GEOMETRIC_SHAPES

BOX_DRAWING

BLOCK_ELEMENTS

GEOMETRIC_SHAPES_EXTENDED

ALCHEMICAL_SYMBOLS

BRAILLE_PATTERNS

CURRENCY_SYMBOLS

DINGBATS

ORNAMENTAL_DINGBATS

EMOTICONS

CHESS_CHECKERS_DRAUGHTS

DOMINO_TILES

JAPANESE_CHESS

MAHJONG_TILES

PLAYING_CARDS

CARD_SUITS

MISCELLANEOUS_SYMBOLS_AND_PICTOGRAPHS
```

MUSICAL_SYMBOLS

ANCIENT_GREEK_MUSICAL_NOTATION

BYZANTINE_MUSICAL_SYMBOLS

TRANSPORT_AND_MAP_SYMBOLS

YIJING_MONO_DI_AND_TRIGRAMS

YIJING_HEXAGRAM_SYMBOLS

TAI_XUAN_JING_SYMBOLS

SPECIALS

TAGS

VARIATION_SELECTORS

VARIATION_SELECTORS_SUPPLEMENT

PRIVATE_USE_AREA

SUPPLEMENTARY_PRIVATE_USE_AREA_A

SUPPLEMENTARY_PRIVATE_USE_AREA_B

Arguments

`lo`	A non-negative integer. Minimum number of repeats, when grouped.
`hi`	positive integer. Maximum number of repeats, when grouped.
`char_class`	`TRUE` or `FALSE`. Should the values be wrapped into a character class?

Format

An object of class `regex` (inherits from `character`) of length 1.

Value

A character vector representing part or all of a regular expression.

Note

Windows currently doesn't handle Unicode points with more than four digits correctly. See `https://bugs.r-project.org/bugzilla3/show_bug.cgi?id=16098`

References

http://www.unicode.org/charts

See Also

ClassGroups

Examples

```
# Classes
latin()
greek_and_coptic()
cyrillic()
arabic()

# With repetition
hebrew(3, 6)
hiragana(1, Inf)
katakana(0, Inf)

# Without a class wrapper
cjk_unified_ideographs(char_class = FALSE)

# Constants
ARMENIAN
LINEAR_B_IDEOGRAMS
DUPLOYAN
OSMANYA

## Not run:
# All the Unicode characer classes
# Not run, since it generates lots of output
setdiff(
  ls("package:rebus.unicode", pattern = lower()),
  ls(
    "package:rebus.unicode",
    pattern = START %R% case_insensitive(or("up", "ugc", "unicode")))
)

## End(Not run)

# Usage
pythag <- "\u03b1^2 + \u03b2^2 = \u03b3^2"
stringi::stri_extract_all_regex(pythag, greek_and_coptic())
```

UnicodeOperators *Unicode Pattern Operators*

Description

Manipulate and combine Unicode Properties.

Usage

```
unicode_inverse(x, char_class = TRUE)

unicode_union(..., char_class = TRUE)

unicode_intersect(x, y, char_class = TRUE)

unicode_setdiff(x, y, char_class = TRUE)
```

Arguments

x	A character vector containing Unicode General Category or Unicode Properties. Use the functional forms (ugc_*()) not the constants.
char_class	TRUE or FALSE. Should the values be wrapped into a character class?
...	Character vectors containing Unicode General Category or Unicode Properties. Use the functional forms (ugc_*()) not the constants.
y	A character vector containing Unicode General Category or Unicode Properties. Use the functional forms (ugc_*()) not the constants.

Note

Use these with ICU-based regular expression engines (stringi and stringr).

References

http://userguide.icu-project.org/strings/unicodeset

Examples

```
# POSIX [:punct:] is more or less equivalent to the union of
# Unicode punctuation and symbol general categories
unicode_union(ugc_punctuation(), ugc_symbol())

# Everything except "A" to "Z" (including punctuation, control chars etc.)
unicode_inverse("[A-Z]")

# Uppercase letters, except "A" to "Z"
unicode_setdiff(ugc_uppercase_letter(), "[A-Z]")

# "A" to "F" (in upper or lower case)
unicode_intersect(ugc_letter(), up_ascii_hex_digit())

# Usage
x <- c(letters, LETTERS)
rx <- unicode_intersect(ugc_letter(), up_ascii_hex_digit())
stringi::stri_extract_first_regex(x, rx)
```

`up_alphabetic` *Unicode Properties*

Description

Match a Unicode Property.

Usage

```
up_alphabetic(lo, hi, char_class = TRUE)

up_ascii_hex_digit(lo, hi, char_class = TRUE)

up_bidi_control(lo, hi, char_class = TRUE)

up_bidi_mirrored(lo, hi, char_class = TRUE)

up_case_ignorable(lo, hi, char_class = TRUE)

up_case_sensitive(lo, hi, char_class = TRUE)

up_cased(lo, hi, char_class = TRUE)

up_changes_when_casefolded(lo, hi, char_class = TRUE)

up_changes_when_casemapped(lo, hi, char_class = TRUE)

up_changes_when_lowercased(lo, hi, char_class = TRUE)

up_changes_when_nfkc_casefolded(lo, hi, char_class = TRUE)

up_changes_when_titlecased(lo, hi, char_class = TRUE)

up_changes_when_uppercased(lo, hi, char_class = TRUE)

up_dash(lo, hi, char_class = TRUE)

up_default_ignorable_code_point(lo, hi, char_class = TRUE)

up_deprecated(lo, hi, char_class = TRUE)

up_diacritic(lo, hi, char_class = TRUE)

up_extender(lo, hi, char_class = TRUE)

up_hex_digit(lo, hi, char_class = TRUE)
```

```
up_hyphen(lo, hi, char_class = TRUE)

up_id_continue(lo, hi, char_class = TRUE)

up_id_start(lo, hi, char_class = TRUE)

up_ideographic(lo, hi, char_class = TRUE)

up_lowercase(lo, hi, char_class = TRUE)

up_math(lo, hi, char_class = TRUE)

up_noncharacter_code_point(lo, hi, char_class = TRUE)

up_posix_alnum(lo, hi, char_class = TRUE)

up_posix_blank(lo, hi, char_class = TRUE)

up_posix_graph(lo, hi, char_class = TRUE)

up_posix_print(lo, hi, char_class = TRUE)

up_posix_xdigit(lo, hi, char_class = TRUE)

up_quotation_mark(lo, hi, char_class = TRUE)

up_soft_dotted(lo, hi, char_class = TRUE)

up_terminal_punctuation(lo, hi, char_class = TRUE)

up_uppercase(lo, hi, char_class = TRUE)

up_white_space(lo, hi, char_class = TRUE)

UP_ALPHABETIC

UP_ASCII_HEX_DIGIT

UP_BIDI_CONTROL

UP_BIDI_MIRRORED

UP_DASH

UP_DEFAULT_IGNORABLE_CODE_POINT

UP_DEPRECATED
```

UP_DIACRITIC

UP_EXTENDER

UP_HEX_DIGIT

UP_HYPHEN

UP_ID_CONTINUE

UP_ID_START

UP_IDEOGRAPHIC

UP_LOWERCASE

UP_MATH

UP_NONCHARACTER_CODE_POINT

UP_QUOTATION_MARK

UP_SOFT_DOTTED

UP_TERMINAL_PUNCTUATION

UP_UPPERCASE

UP_WHITE_SPACE

UP_CASE_SENSITIVE

UP_POSIX_ALNUM

UP_POSIX_BLANK

UP_POSIX_GRAPH

UP_POSIX_PRINT

UP_POSIX_XDIGIT

UP_CASED

UP_CASE_IGNORABLE

UP_CHANGES_WHEN_LOWERCASED

UP_CHANGES_WHEN_UPPERCASED

UP_CHANGES_WHEN_TITLECASED

UP_CHANGES_WHEN_CASEFOLDED

UP_CHANGES_WHEN_CASEMAPPED

UP_CHANGES_WHEN_NFKC_CASEFOLDED

Arguments

lo	A non-negative integer. Minimum number of repeats, when grouped.
hi	positive integer. Maximum number of repeats, when grouped.
char_class	TRUE or FALSE. Should the values be wrapped into a character class?

Format

An object of class regex (inherits from character) of length 1.

Value

A character vector representing part or all of a regular expression.

References

Table 12 of the Unicode Standard Annex #44 defines the Unicode General Categories. http://www.unicode.org/reports/tr44/

You can see which characters are contained in a category by visiting, e.g., http://www.fileformat.info/info/unicode/category/Nd/list.htm

See Also

unicode_general_category, Unicode, stringi-search-charclass

Examples

```
# Classes
up_math()
up_posix_alnum()
up_changes_when_uppercased()
up_diacritic()

# With repetition
ugc_nonspacing_mark(3, 6)
up_quotation_mark(1, Inf)
up_posix_xdigit(0, Inf)

# Without a class wrapper
up_hyphen(char_class = FALSE)
```

```
# Constants
UP_ALPHABETIC
UP_DASH
UP_POSIX_ALNUM
UP_CHANGES_WHEN_LOWERCASED

## Not run:
# All the Unicode properties.
# Not run, since it generates lots of output
ls("package:rebus.unicode", pattern = "^up")

## End(Not run)

# Usage
# Hello in Samoan, Serbian, Persian, Simplified Chinese
hello <- "t\u101lofa, \u437\u434\u440\u430\u432\u43e, \u633\u644\u627\u645, \u4f60\u597d"
stringi::stri_extract_all_regex(hello, up_alphabetic(1, Inf))
stringi::stri_extract_all_regex(hello, up_case_sensitive(1, Inf))
```

Index