3.10.3 Letters and Letter-like Forms

Greek Letters

form	full name	aliases	form	full name	aliases
α	\[Alpha]	:a:, :alpha:	A	\[CapitalAlpha]	:A:, :Alpha:
β	\[Beta]	:b:, :beta:	B	\[CapitalBeta]	:B:, :Beta:
γ	\[Gamma]	:g:, :gamma:	Γ	\[CapitalGamma]	:G:, :Gamma:
δ	\[Delta]	:d:, :delta:	Δ	\[CapitalDelta]	:D:, :Delta:
ϵ	\[Epsilon]	:e:, :epsilon:	E	\[CapitalEpsilon]	:E:, :Epsilon:
ε	\[CurlyEpsilon]	:ce:, :cepsilon:			
ζ	\[Zeta]	:z:, :zeta:	Z	\[CapitalZeta]	:Z:, :Zeta:
η	\[Eta]	:h:, :et:, :eta:	H	\[CapitalEta]	:H:, :Et:, :Eta:
θ	\[Theta]	:q:, :th:, :theta:	Θ	\[CapitalTheta]	:Q:, :Th:, :Theta:
ϑ	\[CurlyTheta]	:cq:, :cth:, :ctheta:			
ι	\[Iota]	:i:, :iota:	I	\[CapitalIota]	:I:, :Iota:
κ	\[Kappa]	:k:, :kappa:	K	\[CapitalKappa]	:K:, :Kappa:
$\varkappa$	\[CurlyKappa]	:ck:, :ckappa:			
λ	\[Lambda]	:l:, :lambda:	Λ	\[CapitalLambda]	:L:, :Lambda:
μ	\[Mu]	:m:, :mu:	M	\[CapitalMu]	:M:, :Mu:
ν	\[Nu]	:n:, :nu:	N	\[CapitalNu]	:N:, :Nu:
ξ	\[Xi]	:x:, :xi:	Ξ	\[CapitalXi]	:X:, :Xi:
o	\[Omicron]	:om:, :omicron:	O	\[CapitalOmicron]	:Om:, :Omicron:
π	\[Pi]	:p:, :pi:	Π	\[CapitalPi]	:P:, :Pi:
ϖ	\[CurlyPi]	:cp:, :cpi:			
ρ	\[Rho]	:r:, :rho:	P	\[CapitalRho]	:R:, :Rho:
ϱ	\[CurlyRho]	:cr:, :crho:			
σ	\[Sigma]	:s:, :sigma:	Σ	\[CapitalSigma]	:S:, :Sigma:
ς	\[FinalSigma]	:fs:			
τ	\[Tau]	:t:, :tau:	T	\[CapitalTau]	:T:, :Tau:
υ	\[Upsilon]	:u:, :upsilon:	Υ	\[CapitalUpsilon]	:U:, :Upsilon:
			$\varUpsilon$	\[CurlyCapitalUpsilon]	:cU:, :cUpsilon:
ϕ	\[Phi]	:f:, :ph:, :phi:	Φ	\[CapitalPhi]	:F:, :Ph:, :Phi:
φ	\[CurlyPhi]	:j:, :cph:, :cphi:			
χ	\[Chi]	:c:, :ch:, :chi:	X	\[CapitalChi]	:C:, :Ch:, :Chi:
ψ	\[Psi]	:y:, :ps:, :psi:	Ψ	\[CapitalPsi]	:Y:, :Ps:, :Psi:
ω	\[Omega]	:o:, :w:, :omega:	Ω	\[CapitalOmega]	:O:, :W:, :Omega:
$\digamma$	\[Digamma]	:di:, :digamma:	F	\[CapitalDigamma]	:Di:, :Digamma:
$\varkappa$	\[Koppa]	:ko:, :koppa:	$\Koppa$	\[CapitalKoppa]	:Ko:, :Koppa:
ς	\[Stigma]	:sti:, :stigma:	$\Stigma$	\[CapitalStigma]	:Sti:, :Stigma:
$\san$	\[Sampi]	:sa:, :sampi:	$\San$	\[CapitalSampi]	:Sa:, :Sampi:

The complete collection of Greek letters in *Mathematica*.

You can use Greek letters as the names of symbols. The only Greek letter with a built-in meaning in `StandardForm` is π, which *Mathematica* takes to stand for the symbol `Pi`.

Note that even though π on its own is assigned a built-in meaning, combinations such as $\pi 2$ or $x\pi$ have no built-in meanings.

The Greek letters Σ and Π look very much like the operators for sum and product. But as discussed above, these operators are different characters, entered as `\[Sum]` and `\[Product]` respectively.

Similarly, ϵ is different from the $\in$ operator `\[Element]`, and μ is different from µ or `\[Micro]`.

Some capital Greek letters such as `\[CapitalAlpha]` look essentially the same as capital English letters. *Mathematica* however treats them as different characters, and in `TraditionalForm` it uses `\[CapitalBeta]`, for example, to denote the built-in function `Beta`.

Following common convention, lower-case Greek letters are rendered slightly slanted in the standard fonts provided with *Mathematica*, while capital Greek letters are unslanted.

Almost all Greek letters that do not look similar to English letters are widely used in science and mathematics. The **capital xi** Ξ is rare, though it is used to denote the cascade hyperon particles, the grand canonical partition function and regular language complexity. The **capital upsilon** Υ is also rare, though it is used to denote $b\bar{b}$ particles, as well as the vernal equinox.

Curly Greek letters are often assumed to have different meanings from their ordinary counterparts. Indeed, in pure mathematics a single formula can sometimes contain both curly and ordinary forms of a particular letter. The curly pi ϖ is rare, except in astronomy.

The **final sigma** ς is used for sigmas that appear at the ends of words in written Greek; it is not commonly used in technical notation.

The **digamma** $\digamma$, **koppa** $\varkappa$, **stigma** ς and **sampi** π are archaic Greek letters. These letters provide a convenient extension to the usual set of Greek letters. They are sometimes needed in making correspondences with English letters. The digamma corresponds to an English w, and koppa to an English q. Digamma is occasionally used to denote the digamma function `PolyGamma[`x`]`.

Variants of English Letters

form	full name	alias	form	full name	alias
l	\[ScriptL]	:scl:	$\mathbb{C}$	\[DoubleStruckCapitalC]	:dsC:
$\mathcal{E}$	\[ScriptCapitalE]	:scE:	$\mathbb{R}$	\[DoubleStruckCapitalR]	:dsR:
$\mathcal{H}$	\[ScriptCapitalH]	:scH:	$\mathbb{Q}$	\[DoubleStruckCapitalQ]	:dsQ:
$\mathcal{L}$	\[ScriptCapitalL]	:scL:	$\mathbb{Z}$	\[DoubleStruckCapitalZ]	:dsZ:
$\mathfrak{C}$	\[GothicCapitalC]	:goC:	$\mathbb{N}$	\[DoubleStruckCapitalN]	:dsN:
$\mathfrak{H}$	\[GothicCapitalH]	:goH:	$\imath$	\[DotlessI]	
$\mathfrak{I}$	\[GothicCapitalI]	:goI:	$\jmath$	\[DotlessJ]	
$\mathfrak{R}$	\[GothicCapitalR]	:goR:	$\wp$	\[WeierstrassP]	:wp:

Some commonly used variants of English letters.

By using menu items in the notebook front end, or explicit `StyleBox` objects, you can make changes in the font and style of ordinary text. However, such changes are usually discarded whenever you send input to the *Mathematica* kernel.

Script, gothic and double-struck characters are however treated as fundamentally different from their ordinary forms. This means that even though a `C` that is italic or a different size will be considered equivalent to an ordinary `C` when fed to the kernel, a double-struck `ℂ` will not.

Different styles and sizes of C are treated as the same by the kernel. But gothic and double-struck characters are treated as different.

```
In[1]:= C + C + C + C + ℂ
Out[1]= 3 C + C + ℂ
```

In standard mathematical notation, capital script and gothic letters are sometimes used interchangeably. The double-struck letters, sometimes called blackboard or openface letters, are conventionally used to denote specific sets. Thus, for example, $\mathbb{C}$ conventionally denotes the set of complex numbers, and $\mathbb{Z}$ the set of integers.

Dotless i and j are not usually taken to be different in meaning from ordinary i and j; they are simply used when overscripts are being placed on the ordinary characters.

`\[WeierstrassP]` is a notation specifically used for the Weierstrass P function `WeierstrassP`.

full names	aliases	
\[ScriptA] – \[ScriptZ]	:sca: – :scz:	lower-case script letters
\[ScriptCapitalA] – \[ScriptCapitalZ]	:scA: – :scZ:	upper-case script letters
\[GothicA] – \[GothicZ]	:goa: – :goz:	lower-case gothic letters
\[GothicCapitalA] – \[GothicCapitalZ]	:goA: – :goZ:	upper-case gothic letters
\[DoubleStruckA] – \[DoubleStruckZ]	:dsa: – :dsz:	lower-case double-struck letters
\[DoubleStruckCapitalA] – \[DoubleStruckCapitalZ]	:dsA: – :dsZ:	upper-case double-struck letters

Complete alphabets of variant English letters.

Hebrew Letters

form	full name	alias	form	full name
ℵ	\[Aleph]	:al:	ג	\[Gimel]
ב	\[Bet]		ד	\[Dalet]

Hebrew characters.

Hebrew characters are used in mathematics in the theory of transfinite sets; $\aleph_0$ is for example used to denote the total number of integers.

Units and Letter-like Mathematical Symbols

form	full name	alias	form	full name	alias
μ	\[Micro]	:mi:	°	\[Degree]	:deg:
℧	\[Mho]	:mho:	∅	\[EmptySet]	:es:
Å	\[Angstrom]	:Ang:	∞	\[Infinity]	:inf:
ℏ	\[HBar]	:hb:	ⅇ	\[ExponentialE]	:ee:
¢	\[Cent]	:cent:	ⅈ	\[ImaginaryI]	:ii:
£	\[Sterling]		ⅉ	\[ImaginaryJ]	:jj:
€	\[Euro]		π	\[DoubledPi]	:pp:
¥	\[Yen]		ℽ	\[DoubledGamma]	:gg:

Units and letter-like mathematical symbols.

Mathematica treats ° or `\[Degree]` as the symbol `Degree`, so that, for example, `30°` is equivalent to `30 Degree`.

Note that μ, Å and ∅ are all distinct from the ordinary letters μ (`\[Mu]`), Å (`\[CapitalARing]`) and Ø (`\[CapitalOSlash]`).

Mathematica interprets ∞ as `Infinity`, ⅇ as `E`, and both ⅈ and ⅉ as `I`. The characters ⅇ, ⅈ and ⅉ are provided as alternatives to the usual upper-case letters `E` and `I`.

π and γ are not by default assigned meanings in `StandardForm`. You can therefore use π to represent a pi that will not automatically be treated as `Pi`. In `TraditionalForm` γ is interpreted as `EulerGamma`.

form	full name	alias	form	full name	alias
∂	\[PartialD]	:pd:	∇	\[Del]	:del:
ⅆ	\[DifferentialD]	:dd:	∑	\[Sum]	:sum:
ⅅ	\[CapitalDifferentialD]	:DD:	∏	\[Product]	:prod:

Operators that look like letters.

∇ is an operator while ℏ, ° and ¥ are ordinary symbols.

```
In[1]:= {∇ f, ℏ^2, 45°, 5000¥} // FullForm
Out[1]//FullForm= List[Del[f], Power[\[HBar], 2],
                    Times[45, Degree], Times[5000, \[Yen]]]
```

Shapes, Icons and Geometrical Constructs

form	full name	alias	form	full name	alias
▪	\[FilledVerySmallSquare]	:fvssq:	○	\[EmptySmallCircle]	:esci:
▫	\[EmptySmallSquare]	:essq:	●	\[FilledSmallCircle]	:fsci:
▪	\[FilledSmallSquare]	:fssq:	○	\[EmptyCircle]	:eci:
□	\[EmptySquare]	:esq:	◍	\[GrayCircle]	:gci:
▩	\[GraySquare]	:gsq:	●	\[FilledCircle]	:fci:
■	\[FilledSquare]	:fsq:	△	\[EmptyUpTriangle]	
⬚	\[DottedSquare]		▲	\[FilledUpTriangle]	
▯	\[EmptyRectangle]		▽	\[EmptyDownTriangle]	
▮	\[FilledRectangle]		▼	\[FilledDownTriangle]	
◇	\[EmptyDiamond]		★	\[FivePointedStar]	:*5:
◆	\[FilledDiamond]		✱	\[SixPointedStar]	:*6:

Shapes.

Shapes are most often used as "dingbats" to emphasize pieces of text. But *Mathematica* treats them as letter-like forms, and also allows them to appear in the names of symbols.

In addition to shapes such as \[EmptySquare], there are characters such as \[Square] which are treated by *Mathematica* as operators rather than letter-like forms.

form	full name	alias	form	full name	aliases
✲	\[MathematicaIcon]	:math:	☺	\[HappySmiley]	:):, :-):
✲	\[KernelIcon]		😐	\[NeutralSmiley]	:-|:
💡	\[LightBulb]		☹	\[SadSmiley]	:-(:
⚠	\[WarningSign]		😵	\[FreakedSmiley]	:-@:
⌚	\[WatchIcon]		🐺	\[Wolf]	:wf:, :wolf:

Icons.

You can use icon characters just like any other letter-like forms.

```
In[1]:= Expand[(☺ + 🐺)^4]
Out[1]= ☺^4 + 4 ☺^3 🐺 + 6 ☺^2 🐺^2 + 4 ☺ 🐺^3 + 🐺^4
```

form	full name	form	full name
∠	\[Angle]	⧼	\[SphericalAngle]
⦜	\[RightAngle]	△	\[EmptyUpTriangle]
∡	\[MeasuredAngle]	∅	\[Diameter]

Notation for geometrical constructs.

Since *Mathematica* treats characters like ∠ as letter-like forms, constructs like ∠BC are treated in *Mathematica* as single symbols.

Textual Elements

form	full name	alias	form	full name	alias
-	\[Dash]	:-:	′	\[Prime]	:':
—	\[LongDash]	:--:	″	\[DoublePrime]	:'':
•	\[Bullet]	:bu:	‵	\[ReversePrime]	:`:
¶	\[Paragraph]		‶	\[ReverseDoublePrime]	:``:
§	\[Section]		«	\[LeftGuillemet]	:g<<:
¿	\[DownQuestion]	:d?:	»	\[RightGuillemet]	:g>>:
¡	\[DownExclamation]	:d!:	…	\[Ellipsis]	:...:

Characters used for punctuation and annotation.

form	full name	form	full name	alias
©	\[Copyright]	†	\[Dagger]	:dg:
®	\[RegisteredTrademark]	‡	\[DoubleDagger]	:ddg:
™	\[Trademark]	♣	\[ClubSuit]	
♭	\[Flat]	♢	\[DiamondSuit]	
♮	\[Natural]	♡	\[HeartSuit]	
♯	\[Sharp]	♠	\[SpadeSuit]	

Other characters used in text.

form	full name	alias	form	full name	alias
-	\[HorizontalLine]	:hline:	⏝	\[UnderParenthesis]	:u(:
'	\[VerticalLine]	:vline:	⏜	\[OverParenthesis]	:o(:
…	\[Ellipsis]	:...:	⎵	\[UnderBracket]	:u[:
⋯	\[CenterEllipsis]		⎴	\[OverBracket]	:o[:
⋮	\[VerticalEllipsis]		⏟	\[UnderBrace]	:u{:
⋰	\[AscendingEllipsis]		⏞	\[OverBrace]	:o{:
⋱	\[DescendingEllipsis]				

Characters used in building sequences and arrays.

The under and over braces grow to enclose the whole expression.

```
In[1]:= Underoverscript[Expand[(1 + x)^4],
                        \[UnderBrace], \[OverBrace]]
```

$$\text{Out[1]=}\ \overbrace{\underbrace{1+4x+6x^2+4x^3+x^4}}$$

Extended Latin Letters

Mathematica supports all the characters commonly used in Western European languages based on Latin scripts.

form	full name	alias
à	\[AGrave]	:a`:
á	\[AAcute]	:a':
â	\[AHat]	:a^:
ã	\[ATilde]	:a~:
ä	\[ADoubleDot]	:a":
å	\[ARing]	:ao:
ā	\[ABar]	:a-:
ă	\[ACup]	:au:
æ	\[AE]	:ae:
ć	\[CAcute]	:c':
ç	\[CCedilla]	:c,:
č	\[CHacek]	:cv:
è	\[EGrave]	:e`:
é	\[EAcute]	:e':
ē	\[EBar]	:e-:
ê	\[EHat]	:e^:
ë	\[EDoubleDot]	:e":
ě	\[ECup]	:eu:
ì	\[IGrave]	:i`:
í	\[IAcute]	:i':
î	\[IHat]	:i^:
ï	\[IDoubleDot]	:i":
ĭ	\[ICup]	:iu:
ð	\[Eth]	:d-:
ł	\[LSlash]	:l/:
ñ	\[NTilde]	:n~:
ò	\[OGrave]	:o`:
ó	\[OAcute]	:o':
ô	\[OHat]	:o^:
õ	\[OTilde]	:o~:
ö	\[ODoubleDot]	:o":
ő	\[ODoubleAcute]	:o'':
ø	\[OSlash]	:o/:
š	\[SHacek]	:sv:
ù	\[UGrave]	:u`:
ú	\[UAcute]	:u':
û	\[UHat]	:u^:
ü	\[UDoubleDot]	:u":
ű	\[UDoubleAcute]	:u'':
ý	\[YAcute]	:y':
þ	\[Thorn]	:thn:
ß	\[SZ]	:sz:, :ss:

form	full name	alias
À	\[CapitalAGrave]	:A`:
Á	\[CapitalAAcute]	:A':
Â	\[CapitalAHat]	:A^:
Ã	\[CapitalATilde]	:A~:
Ä	\[CapitalADoubleDot]	:A":
Å	\[CapitalARing]	:Ao:
Ā	\[CapitalABar]	:A-:
Ă	\[CapitalACup]	:Au:
Æ	\[CapitalAE]	:AE:
Ć	\[CapitalCAcute]	:C':
Ç	\[CapitalCCedilla]	:C,:
Č	\[CapitalCHacek]	:Cv:
È	\[CapitalEGrave]	:E`:
É	\[CapitalEAcute]	:E':
Ē	\[CapitalEBar]	:E-:
Ê	\[CapitalEHat]	:E^:
Ë	\[CapitalEDoubleDot]	:E":
Ě	\[CapitalECup]	:Eu:
Ì	\[CapitalIGrave]	:I`:
Í	\[CapitalIAcute]	:I':
Î	\[CapitalIHat]	:I^:
Ï	\[CapitalIDoubleDot]	:I":
Ĭ	\[CapitalICup]	:Iu:
Đ	\[CapitalEth]	:D-:
Ł	\[CapitalLSlash]	:L/:
Ñ	\[CapitalNTilde]	:N~:
Ò	\[CapitalOGrave]	:O`:
Ó	\[CapitalOAcute]	:O':
Ô	\[CapitalOHat]	:O^:
Õ	\[CapitalOTilde]	:O~:
Ö	\[CapitalODoubleDot]	:O":
Ő	\[CapitalODoubleAcute]	:O'':
Ø	\[CapitalOSlash]	:O/:
Š	\[CapitalSHacek]	:Sv:
Ù	\[CapitalUGrave]	:U`:
Ú	\[CapitalUAcute]	:U':
Û	\[CapitalUHat]	:U^:
Ü	\[CapitalUDoubleDot]	:U":
Ű	\[CapitalUDoubleAcute]	:U'':
Ý	\[CapitalYAcute]	:Y':
Þ	\[CapitalThorn]	:Thn:

Variants of English letters.

Most of the characters shown are formed by adding diacritical marks to ordinary English letters. Exceptions include \[SZ] ß, used in German, and \[Thorn] þ and \[Eth] ð, used primarily in Old English.

You can make additional characters by explicitly adding diacritical marks yourself.

char CTRL-[&] *mark* CTRL-[␣] or \(*char*\&*mark*\)	add a mark above a character
char CTRL-[+] *mark* CTRL-[␣] or \(*char*\+*mark*\)	add a mark below a character

Adding marks above and below characters.

form	*alias*	*full name*	
'	(keyboard character)	\[RawQuote]	acute accent
´	:':	\[Prime]	acute accent
`	(keyboard character)	\[RawBackquote]	grave accent
‵	:`:	\[ReversePrime]	grave accent
¨	(keyboard characters)		umlaut or diaeresis
^	(keyboard character)	\[RawWedge]	circumflex or hat
°	:esci:	\[EmptySmallCircle]	ring
˙	(keyboard character)	\[RawDot]	dot
~	(keyboard character)	\[RawTilde]	tilde
¯	(keyboard character)	\[RawUnderscore]	bar or macron
ˇ	:hc:	\[Hacek]	hacek or check
˘	:bv:	\[Breve]	breve
‿	:dbv:	\[DownBreve]	tie accent
″	:'':	\[DoublePrime]	long umlaut
¸	:cd:	\[Cedilla]	cedilla

Diacritical marks to add to characters.

3.10.4 Operators

Basic Mathematical Operators

form	full name	alias	form	full name	alias
×	\[Times]	:*:	×	\[Cross]	:cross:
÷	\[Divide]	:div:	±	\[PlusMinus]	:+-:
√	\[Sqrt]	:sqrt:	∓	\[MinusPlus]	:-+:

Some operators used in basic arithmetic and algebra.

Note that the × for \[Cross] is distinguished by being drawn slightly smaller than the × for \[Times].

$x \times y$	Times[x, y]	multiplication
$x \div y$	Divide[x, y]	division
$\sqrt{x}$	Sqrt[x]	square root
$x \times y$	Cross[x, y]	vector cross product
$\pm x$	PlusMinus[x]	(no built-in meaning)
$x \pm y$	PlusMinus[x, y]	(no built-in meaning)
$\mp x$	MinusPlus[x]	(no built-in meaning)
$x \mp y$	MinusPlus[x, y]	(no built-in meaning)

Interpretation of some operators in basic arithmetic and algebra.

Operators in Calculus

form	full name	alias	form	full name	alias
∇	\[Del]	:del:	$\int$	\[Integral]	:int:
∂	\[PartialD]	:pd:	$\oint$	\[ContourIntegral]	:cint:
d	\[DifferentialD]	:dd:	$\oiint$	\[DoubleContourIntegral]	
$\sum$	\[Sum]	:sum:	$\oint$	\[CounterClockwiseContourIntegral]	:cccint:
$\prod$	\[Product]	:prod:	$\oint$	\[ClockwiseContourIntegral]	:ccint:

Operators used in calculus.

Logical and Other Connectives

form	full name	aliases
∧	\[And]	:&&:, :and:
∨	\[Or]	:||:, :or:
¬	\[Not]	:!:, :not:
∈	\[Element]	:el:
∀	\[ForAll]	:fa:
∃	\[Exists]	:ex:
∄	\[NotExists]	:!ex:
⊻	\[Xor]	:xor:
⊼	\[Nand]	:nand:
⊽	\[Nor]	:nor:

form	full name	alias
⇒	\[Implies]	:=>:
⊃	\[RoundImplies]	
∴	\[Therefore]	:tf:
∵	\[Because]	
⊢	\[RightTee]	
⊣	\[LeftTee]	
⊨	\[DoubleRightTee]	
⫤	\[DoubleLeftTee]	
∋	\[SuchThat]	:st:
∣	\[VerticalSeparator]	:|:
:	\[Colon]	:::

Operators used as logical connectives.

The operators ∧, ∨ and ¬ are interpreted as corresponding to the built-in functions `And`, `Or` and `Not`, and are equivalent to the keyboard operators `&&`, `||` and `!`. The operators ⊻, ⊼ and ⊽ correspond to the built-in functions `Xor`, `Nand` and `Nor`. Note that ¬ is a prefix operator.

x⇒*y* and *x*⇒*y* are both taken to give the built-in function `Implies[`*x*`, `*y*`]`. *x*∈*y* gives the built-in function `Element[`*x*`, `*y*`]`.

This is interpreted using the built-in functions `And` and `Implies`.

```
In[1]:= 3 < 4 ∧ x > 5 ⇒ y < 7
Out[1]= Implies[x > 5, y < 7]
```

Mathematica supports most of the standard syntax used in mathematical logic. In *Mathematica*, however, the variables that appear in the quantifiers ∀, ∃ and ∄ must appear as subscripts. If they appeared directly after the quantifier symbols then there could be a conflict with multiplication operations.

∀ and ∃ are essentially prefix operators like ∂.

```
In[2]:= ∀x ∃y ϕ[x, y] // FullForm
Out[2]//FullForm= ForAll[x, Exists[y, \[Phi][x, y]]]
```