Fondamenti di Informatica (Elettronici)

THINK JULIA – Capitolo 2

12 ottobre 2020

2. Variabili, Espressioni e Dichiarazioni[1]

[1] Tratto da https://benlauwens.github.io/ThinkJulia.jl/latest/book.html, disponibile sotto Licenza 'Creative Commons Attribution-NonCommercial 3.0 Unported'.

Section 1

Istruzioni di assegnazione

Istruzioni di assegnazione

Una delle caratteristiche più importanti di un linguaggio di programmazione è la capacità di manipolare le variabili.

Una variabile è un nome che fa riferimento a un valore.

Una dichiarazione di assegnazione crea una nuova variabile e le assegna un valore:

```
julia> message = "And now for something completely different"
"And now for something completely different"
julia> n = 17
17
julia> pi_val = 3.141592653589793
3.141592653589793
```

Diagramma di stato

Il precedente esempio contiene tre assegnazioni.

- La prima assegna una stringa a una nuova variabile denominata `message`;

Diagramma di stato

Il precedente esempio contiene tre assegnazioni.

- La prima assegna una stringa a una nuova variabile denominata `message`;
- la seconda fornisce l'intero 17 alla variabile `n`;

Diagramma di stato

Il precedente esempio contiene tre assegnazioni.

- La prima assegna una stringa a una nuova variabile denominata `message`;
- la seconda fornisce l'intero 17 alla variabile `n`;
- la terza assegna il valore (approssimato) di π a `\pi_val` (\pi TAB).

Diagramma di stato

Il precedente esempio contiene tre assegnazioni.

- La prima assegna una stringa a una nuova variabile denominata `message`;
- la seconda fornisce l'intero 17 alla variabile `n`;
- la terza assegna il valore (approssimato) di π a `\pi_val` (\pi TAB).

Diagramma di stato

Il precedente esempio contiene tre assegnazioni.

- La prima assegna una stringa a una nuova variabile denominata `message`;
- la seconda fornisce l'intero 17 alla variabile `n`;
- la terza assegna il valore (approssimato) di π a `\pi_val` (\pi TAB).

Un modo comune per rappresentare le variabili nel testo è scrivere il nome con una freccia che punta al suo valore.

Questo tipo di figura è chiamato diagramma di stato, perché mostra in quale stato si trova ciascuna delle variabili (pensatela come lo stato mentale della variabile).

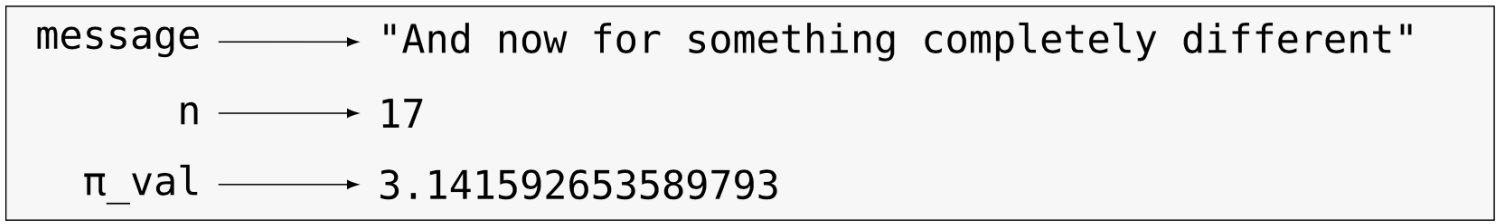


Figure 1. State diagram

Il diagramma di stato mostra il risultato dell'esempio precedente.

Section 2

Nomi di variabili

Sintassi dei nomi delle variabili

I programmatori generalmente scelgono nomi significativi per le loro variabili: documentano come viene utilizzata la variabile.

- I nomi delle variabili possono essere lunghi a piacere.

`your_name` oppure `airspeed_of_unladen_swallow` .

Sintassi dei nomi delle variabili

I programmatori generalmente scelgono nomi significativi per le loro variabili: documentano come viene utilizzata la variabile.

- I nomi delle variabili possono essere lunghi a piacere.
- Possono contenere quasi tutti i caratteri Unicode (vedi `Caratteri`), ma non possono iniziare con un numero.

`your_name` oppure `airspeed_of_unladen_swallow` .

Sintassi dei nomi delle variabili

I programmatori generalmente scelgono nomi significativi per le loro variabili: documentano come viene utilizzata la variabile.

- I nomi delle variabili possono essere lunghi a piacere.
- Possono contenere quasi tutti i caratteri Unicode (vedi `Caratteri`), ma non possono iniziare con un numero.
- È legale usare lettere maiuscole, ma è convenzionale usare solo lettere minuscole per i nomi delle variabili.

`your_name` oppure `airspeed_of_unladen_swallow` .

Sintassi dei nomi delle variabili

I programmatori generalmente scelgono nomi significativi per le loro variabili: documentano come viene utilizzata la variabile.

- I nomi delle variabili possono essere lunghi a piacere.
- Possono contenere quasi tutti i caratteri Unicode (vedi `Caratteri`), ma non possono iniziare con un numero.
- È legale usare lettere maiuscole, ma è convenzionale usare solo lettere minuscole per i nomi delle variabili.
- I caratteri Unicode possono essere inseriti tramite il completamento con tabulazione delle abbreviazioni simili a quelle LaTeX in Julia REPL.

`your_name` oppure `airspeed_of_unladen_swallow` .

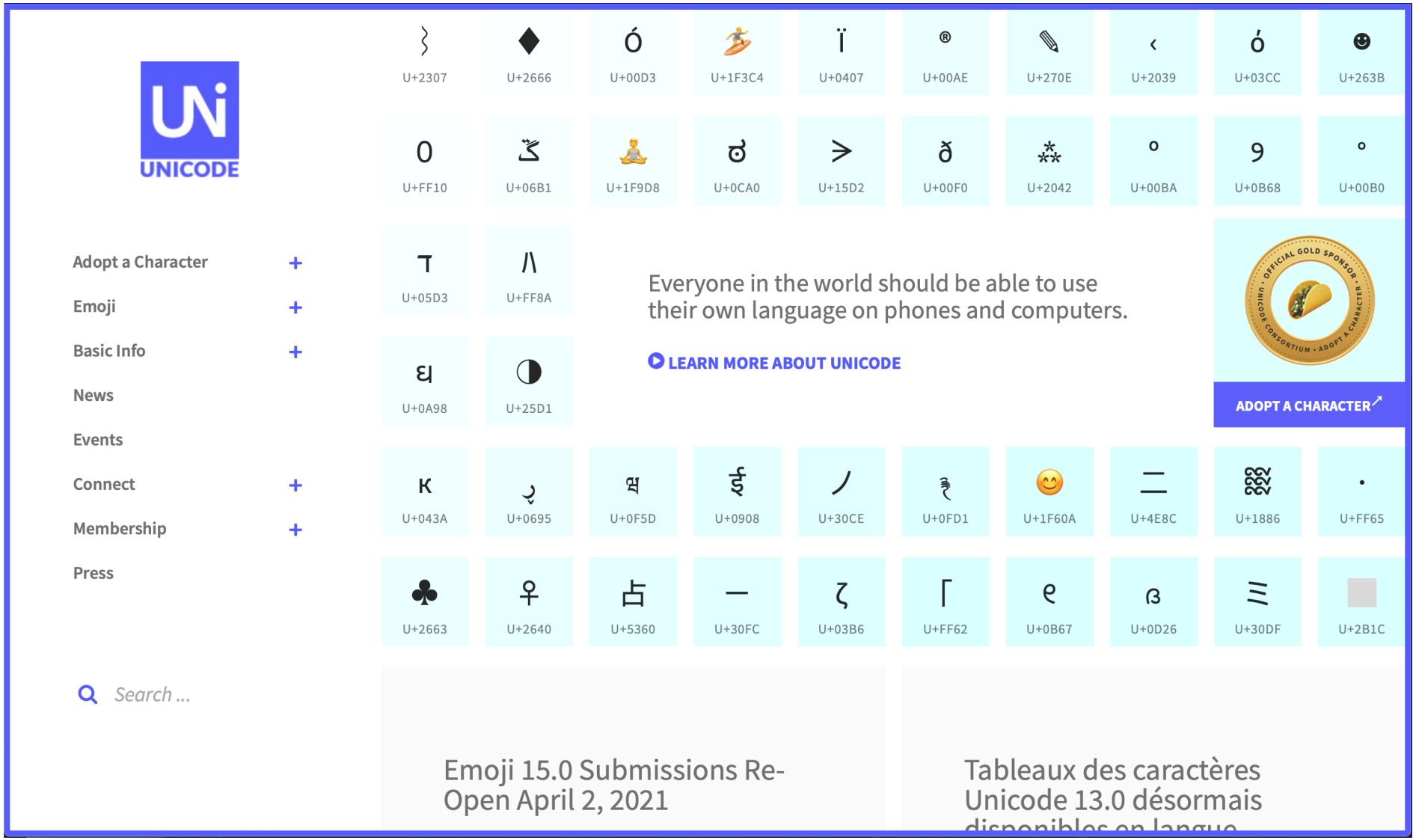
Sintassi dei nomi delle variabili

I programmatori generalmente scelgono nomi significativi per le loro variabili: documentano come viene utilizzata la variabile.

- I nomi delle variabili possono essere lunghi a piacere.
- Possono contenere quasi tutti i caratteri Unicode (vedi `Caratteri`), ma non possono iniziare con un numero.
- È legale usare lettere maiuscole, ma è convenzionale usare solo lettere minuscole per i nomi delle variabili.
- I caratteri Unicode possono essere inseriti tramite il completamento con tabulazione delle abbreviazioni simili a quelle LaTeX in Julia REPL.
- Il carattere di sottolineatura, "_", può apparire in un nome. Viene spesso utilizzato in nomi formati con più parole, come:

`your_name` oppure `airspeed_of_unladen_swallow` .

Caratteri Unicode



Esempi di errori di sintassi

Se si assegna a una variabile un nome illegale, si ottiene un errore di sintassi:

```
julia> 76trombones = "big parade"
ERROR: syntax: "76" is not a valid function argument name around REP
```

```
julia> more@ = 1000000
ERROR: syntax: extra token "@" after end of expression
```

```
julia> struct = "Advanced Theoretical Zymurgy"
ERROR: syntax: unexpected "="
```

`76trombones` è illegale perché inizia con un numero. `more@` è illegale perché contiene un carattere illegale, `@`. Ma cosa c'è di sbagliato in `struct`?

Si scopre che "`struct`" è una delle parole chiave di Julia.

Il REPL utilizza parole chiave (keywords) per riconoscere la struttura del programma, e queste non possono essere usate come nomi di variabili.

Julia keywords

from https://docs.julialang.org/en/v1/base/base/

Questo è l'elenco delle parole chiave riservate in Julia:

`baremodule`, `begin`, `break`, `catch`, `const`, `continue`, `do`, `else`, `elseif`, `end`, `export`, `false`, `finally`, `for`, `function`, `global`, `if`, `import`, `let`, `local`, `macro`, `module`, `quote`, `return`, `struct`, `true`, `try`, `using`, `while`.

Queste parole chiave non possono essere utilizzate come nomi di variabili. Le seguenti sequenze di due parole sono invece riservate:

`abstract type`, `mutable struct`, `primitive type`

Infine, `where` viene analizzato come un operatore infisso per scrivere metodi parametrici e definizioni di tipo.

Inoltre, `in` e `isa` vengono analizzati come operatori infissi.

Section 3

Espressioni e dichiarazioni

Espressioni

Un'espressione è una combinazione di valori, variabili e operatori.

Un valore da solo è considerato una espressione, così come una variabile, quindi le seguenti sono tutte espressioni legali:

```
julia> 42
42
julia> n
17
julia> n + 25
42
```

Quando si digita un'espressione al `prompt`, REPL la valuta, il che significa che trova il valore dell'espressione.

In questo esempio, "`n`" ha il valore "`17`" e "`n + 25`" ha il valore "`42`".

Dichiarazioni (statements)

Una dichiarazione è una unità di codice che ha un effetto, come la creazione di una variabile o la visualizzazione di un valore.

```
julia> n = 17
17
julia> println(n)
17
```

La prima riga è una dichiarazione di assegnazione che fornisce un valore a `n`.
La seconda riga è una istruzione `println` che mostra il valore di `n`.

Dichiarazioni (statements)

Una dichiarazione è una unità di codice che ha un effetto, come la creazione di una variabile o la visualizzazione di un valore.

```
julia> n = 17
17
julia> println(n)
17
```

La prima riga è una dichiarazione di assegnazione che fornisce un valore a `n`. La seconda riga è una istruzione `println` che mostra il valore di `n`.

Esecuzione nella REPL

Quando si digita un'istruzione o un'espressione, `REPL` la esegue, il che significa che esegue tutto ciò che specificano l'istruzione o l'espressione.

Section 4

Modalità script

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Per convenzione, gli script Julia hanno nomi che terminano con `.jl`:

- Apri un file di testo,

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Per convenzione, gli script Julia hanno nomi che terminano con `.jl`:

- Apri un file di testo,
- scrivi lo script nel file, ed infine

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Per convenzione, gli script Julia hanno nomi che terminano con `.jl`:

- Apri un file di testo,
- scrivi lo script nel file, ed infine
- salva con l'estensione `.jl`.

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Per convenzione, gli script Julia hanno nomi che terminano con `.jl`:

- Apri un file di testo,
- scrivi lo script nel file, ed infine
- salva con l'estensione `.jl`.

Esecuzione di un file Julia

Finora abbiamo eseguito Julia in modalità interattiva, il che significa interagire direttamente con `REPL`.

Modalità interattiva è un buon modo per iniziare, ma lavorando con più di poche righe di codice, può diventare inefficiente e goffo.

Modalità script l'alternativa è salvare il codice in un file che definiamo script e quindi eseguire Julia in modalità script per eseguire il contenuto del file.

Per convenzione, gli script Julia hanno nomi che terminano con `.jl`:

- Apri un file di testo,
- scrivi lo script nel file, ed infine
- salva con l'estensione `.jl`.

Lo script può essere eseguito in un terminale con il comando di shell:

```
$ julia name_of_the_script.jl
```

Modalità interattiva vs script

Poiché Julia fornisce entrambe le modalità, è possibile testare porzioni di codice in modalità interattiva prima di inserirli in uno script. Ma ci sono differenze tra la modalità interattiva e la modalità script che possono creare confusione.

Ad esempio, se si sta usando Julia come calcolatrice, si potrebbe digitare

```
julia> miles = 26.2
26.2
julia> miles * 1.61
42.182
```

La prima riga assegna un valore alle miglia e visualizza il valore. La seconda riga è un'espressione, quindi REPL la valuta e visualizza il risultato. Si scopre che una maratona è di circa 42 chilometri.

Ma se digiti lo stesso codice in uno script e lo si esegue, non si ottiene alcun output. In modalità `script`, un'espressione, da sola, non ha alcun effetto visibile.

Modalità script

Julia valuta effettivamente l'espressione, ma** non mostra il valore** a meno che non gli si imponga di farlo:

```
miles = 26.2
println(miles * 1.61)
```

Questo comportamento può creare confusione all'inizio. Uno script al contrario contiene una sequenza di istruzioni. Se è presente più di un'istruzione di stampa, i risultati vengono visualizzati uno alla volta durante l'esecuzione delle istruzioni. Ad esempio, lo script

```
println(1)
x = 2
println(x)
```

produce l'output

```
1
2
```

L'istruzione di assegnazione non produce alcun output.

Section 5

Precedenza degli operatori

Regole di precedenza

Quando un'espressione contiene più di un operatore, l'ordine di valutazione dipende dalla precedenza dell'operatore. Per gli operatori matematici, Julia segue la convenzione matematica. L'acronimo `PEMDAS` è un modo utile per ricordare le regole:

Parentesi hanno la precedenza più alta e possono essere utilizzate per forzare la valutazione di un'espressione nell'ordine desiderato.

Poiché le espressioni tra parentesi vengono valutate per prime, `2 * (3-1)` è `4` e `(1 + 1) ^ (5-2)` è `8`. Si possono anche usare le parentesi per rendere un'espressione più facile da leggere, come in `(minute * 100) / 60`, anche se non cambia il risultato.

Regole di precedenza

Quando un'espressione contiene più di un operatore, l'ordine di valutazione dipende dalla precedenza dell'operatore. Per gli operatori matematici, Julia segue la convenzione matematica. L'acronimo `PEMDAS` è un modo utile per ricordare le regole:

Parentesi hanno la precedenza più alta e possono essere utilizzate per forzare la valutazione di un'espressione nell'ordine desiderato.

Poiché le espressioni tra parentesi vengono valutate per prime, `2 * (3-1)` è `4` e `(1 + 1) ^ (5-2)` è `8`. Si possono anche usare le parentesi per rendere un'espressione più facile da leggere, come in `(minute * 100) / 60`, anche se non cambia il risultato.

Esponenziazione ha la successiva precedenza più alta, quindi `1 + 2 ^ 3` è `9`, non `27` e `2 * 3 ^ 2` è `18`, non `36`.

Regole di precedenza

Quando un'espressione contiene più di un operatore, l'ordine di valutazione dipende dalla precedenza dell'operatore. Per gli operatori matematici, Julia segue la convenzione matematica. L'acronimo `PEMDAS` è un modo utile per ricordare le regole:

Parentesi hanno la precedenza più alta e possono essere utilizzate per forzare la valutazione di un'espressione nell'ordine desiderato.
Poiché le espressioni tra parentesi vengono valutate per prime, `2 * (3-1)` è `4` e `(1 + 1) ^ (5-2)` è `8`. Si possono anche usare le parentesi per rendere un'espressione più facile da leggere, come in `(minute * 100) / 60`, anche se non cambia il risultato.

Esponenziazione ha la successiva precedenza più alta, quindi `1 + 2 ^ 3` è `9`, non `27` e `2 * 3 ^ 2` è `18`, non `36`.

Moltiplicazione e Divisione hanno la precedenza maggiore di Addizione e Sottrazione.
Quindi `2 * 3-1` è `5`, non `4` e `6 + 4/2` è `8`, non `5`.

Regole di precedenza

Quando un'espressione contiene più di un operatore, l'ordine di valutazione dipende dalla precedenza dell'operatore. Per gli operatori matematici, Julia segue la convenzione matematica. L'acronimo `PEMDAS` è un modo utile per ricordare le regole:

Parentesi hanno la precedenza più alta e possono essere utilizzate per forzare la valutazione di un'espressione nell'ordine desiderato.
Poiché le espressioni tra parentesi vengono valutate per prime, `2 * (3-1)` è 4 e `(1 + 1) ^ (5-2)` è 8. Si possono anche usare le parentesi per rendere un'espressione più facile da leggere, come in `(minute * 100) / 60`, anche se non cambia il risultato.

Esponenziazione ha la successiva precedenza più alta, quindi `1 + 2 ^ 3` è 9, non 27 e `2 * 3 ^ 2` è 18, non 36.

Moltiplicazione e Divisione hanno la precedenza maggiore di Addizione e Sottrazione. Quindi `2 * 3-1` è 5, non 4 e `6 + 4/2` è 8, non 5.

Operatori con la stessa precedenza vengono valutati da sinistra a destra (eccetto la esponenziazione). Quindi nell'espressione `gradi / 2 *`π, la divisione avviene per prima e il risultato viene moltiplicato per π. Per dividere per 2, si possono usare le parentesi, oppure scrivendo `gradi/2/`π o `gradi/2`π.

Uso delle parentesi

Suggerimento

Non lavoro molto per ricordare la precedenza degli operatori. Se non riesco a capire guardando l'espressione, uso le parentesi per renderla esplicita.

Section 6

Operazioni su stringhe

Nessuna operazione matematica sulle stringhe

In generale, non è possibile eseguire operazioni matematiche sulle stringhe, anche se le stringhe sembrano numeri, quindi quanto segue è illegale:

```
"2" - "1" "uova" / "facile" "terzo" + "un ciondolo"
```

Ma ci sono due eccezioni, "`*`" e "`^`".

L'operatore `*` esegue la concatenazione di stringhe, il che significa che unisce le stringhe collegandole end-to-end.

Con due eccezioni: * e ^

Per esempio:

```
julia> first_str = "throat"
"throat"
julia> second_str = "warbler"
"warbler"
julia> first_str * second_str
"throatwarbler"
```

L'operatore "^" funziona anche sulle stringhe; esegue la ripetizione.

Ad esempio, `"Spam" ^ 3` è "SpamSpamSpam". Se uno dei valori è una stringa, l'altro deve essere un numero intero.

Questo uso di "*" e "^" ha senso per analogia con la moltiplicazione e l'elevamento a potenza.

Proprio come `4^3` è equivalente a `4*4*4`, ci aspettiamo che `"Spam "^3` sia uguale a "Spam""Spam""Spam", e lo è.

Section 7

Commenti

Commenti di linea

È una buona idea aggiungere note ai programmi per spiegare in linguaggio naturale cosa sta facendo il programma. Queste note sono chiamate commenti e iniziano con il simbolo #:

I commenti sono particolarmente utili quando documentano caratteristiche non ovvie del codice.

È ragionevole presumere che il lettore possa capire cosa fa il codice; è più utile spiegare perché.

```
# compute the percentage of the hour that has elapsed
percentage = (minute * 100) / 60
```

In questo caso, il commento appare su una riga da solo. Si possono anche inserire commenti alla fine di una riga:

```
percentage = (minute * 100) / 60 # percentage of an hour
```

Tutto ciò che va da "#" alla fine della riga viene ignorato – non ha effetto sull'esecuzione del programma

Commenti inutili vs commenti utili

Questo commento è ridondante con il codice e inutile:

```
v = 5 # assign 5 to v
```

Questo commento contiene informazioni utili che non sono nel codice:

```
v = 5 # velocity in meters/second.
```

Commenti inutili vs commenti utili

Questo commento è ridondante con il codice e inutile:

```
v = 5 # assign 5 to v
```

Questo commento contiene informazioni utili che non sono nel codice:

```
v = 5 # velocity in meters/second.
```

AVVERTIMENTO

Nomi di variabili validi possono ridurre la necessità di commenti, ma i nomi lunghi possono rendere difficili da leggere espressioni complesse, quindi occorre un compromesso.

Section 8

Correzione degli errori

Tipi di errori

In un programma possono verificarsi tre tipi di errori:

Errore di sintassi "Sintassi" si riferisce alla struttura di un programma e alle regole su quella struttura. Ad esempio, le parentesi devono trovarsi in coppie corrispondenti, quindi "(1 + 2)" è legale, ma "8)" è un errore di sintassi.

Tipi di errori

In un programma possono verificarsi tre tipi di errori:

Errore di sintassi "Sintassi" si riferisce alla struttura di un programma e alle regole su quella struttura. Ad esempio, le parentesi devono trovarsi in coppie corrispondenti, quindi "(1 + 2)" è legale, ma "8)" è un errore di sintassi.

Errore a `runtime` Il secondo tipo di errore è un errore di runtime, così chiamato perché l'errore non viene visualizzato fino a quando il programma non è stato avviato.

Tipi di errori

In un programma possono verificarsi tre tipi di errori:

Errore di sintassi "Sintassi" si riferisce alla struttura di un programma e alle regole su quella struttura. Ad esempio, le parentesi devono trovarsi in coppie corrispondenti, quindi "(1 + 2)" è legale, ma "8)" è un errore di sintassi.

Errore a `runtime` Il secondo tipo di errore è un errore di runtime, così chiamato perché l'errore non viene visualizzato fino a quando il programma non è stato avviato.

Errore semantico significa errore relativo al significato. Se c'è un errore semantico nel tuo programma, lo farà eseguire senza generare messaggi di errore, ma non farà la cosa giusta. (Sono gli errori peggiorie con le peggiori conseguenze)

Section 9

Glossario

Glossario I

variabile Un nome che fa riferimento a un valore.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

parola chiave Una parola riservata utilizzata per analizzare un programma; non è possibile utilizzare parole chiave come `if`, `function` e `while` come nomi di variabili.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

parola chiave Una parola riservata utilizzata per analizzare un programma; non è possibile utilizzare parole chiave come `if`, `function` e `while` come nomi di variabili.

operando Uno dei valori su cui opera un operatore.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

parola chiave Una parola riservata utilizzata per analizzare un programma; non è possibile utilizzare parole chiave come `if`, `function` e `while` come nomi di variabili.

operando Uno dei valori su cui opera un operatore.

espressione Una combinazione di variabili, operatori e valori che fornisce un singolo risultato.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

parola chiave Una parola riservata utilizzata per analizzare un programma; non è possibile utilizzare parole chiave come `if`, `function` e `while` come nomi di variabili.

operando Uno dei valori su cui opera un operatore.

espressione Una combinazione di variabili, operatori e valori che fornisce un singolo risultato.

valutare Per semplificare un'espressione eseguendo le operazioni in modo da ottenere un unico valore.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

parola chiave Una parola riservata utilizzata per analizzare un programma; non è possibile utilizzare parole chiave come `if`, `function` e `while` come nomi di variabili.

operando Uno dei valori su cui opera un operatore.

espressione Una combinazione di variabili, operatori e valori che fornisce un singolo risultato.

valutare Per semplificare un'espressione eseguendo le operazioni in modo da ottenere un unico valore.

dichiarazione Una sezione di codice che rappresenta un comando o un'azione. Finora, le affermazioni che abbiamo visto sono assegnazioni e dichiarazioni di stampa.

Glossario I

variabile Un nome che fa riferimento a un valore.

assegnazione Un'istruzione che assegna un valore a una variabile.

diagramma di stato Una rappresentazione grafica di un insieme di variabili e dei valori a cui si riferiscono.

parola chiave Una parola riservata utilizzata per analizzare un programma; non è possibile utilizzare parole chiave come `if`, `function` e `while` come nomi di variabili.

operando Uno dei valori su cui opera un operatore.

espressione Una combinazione di variabili, operatori e valori che fornisce un singolo risultato.

valutare Per semplificare un'espressione eseguendo le operazioni in modo da ottenere un unico valore.

dichiarazione Una sezione di codice che rappresenta un comando o un'azione. Finora, le affermazioni che abbiamo visto sono assegnazioni e dichiarazioni di stampa.

eseguire Eseguire una istruzione è compiere ciò che essa significa (fare ciò che dice).

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

concatenare Unire due stringhe end-to-end.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

concatenare Unire due stringhe end-to-end.

commento Informazioni in un programma, destinate ad altri programmatori (o a chiunque legga il codice sorgente). Non hanno alcun effetto sull'esecuzione del programma.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

concatenare Unire due stringhe end-to-end.

commento Informazioni in un programma, destinate ad altri programmatori (o a chiunque legga il codice sorgente). Non hanno alcun effetto sull'esecuzione del programma.

Errore di sintassi Un errore in un programma che ne rende impossibile l'analisi (e quindi impossibile da interpretare).

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

concatenare Unire due stringhe end-to-end.

commento Informazioni in un programma, destinate ad altri programmatori (o a chiunque legga il codice sorgente). Non hanno alcun effetto sull'esecuzione del programma.

Errore di sintassi Un errore in un programma che ne rende impossibile l'analisi (e quindi impossibile da interpretare).

errore o eccezione a runtime Errore rilevato durante l'esecuzione del programma.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

concatenare Unire due stringhe end-to-end.

commento Informazioni in un programma, destinate ad altri programmatori (o a chiunque legga il codice sorgente). Non hanno alcun effetto sull'esecuzione del programma.

Errore di sintassi Un errore in un programma che ne rende impossibile l'analisi (e quindi impossibile da interpretare).

errore o eccezione a runtime Errore rilevato durante l'esecuzione del programma.

semantica Il significato di un programma.

Glossario II

modalità interattiva Un modo per utilizzare Julia REPL digitando il codice al prompt.

modalità script Un modo per utilizzare Julia per leggere il codice da uno** script su file** ed eseguirlo.

script Un programma memorizzato in un file.

precedenza degli operatori Regole che regolano l'ordinein cui vengono valutate le sotto-espressioni che coinvolgono più operatori matematici e operandi.

concatenare Unire due stringhe end-to-end.

commento Informazioni in un programma, destinate ad altri programmatori (o a chiunque legga il codice sorgente). Non hanno alcun effetto sull'esecuzione del programma.

Errore di sintassi Un errore in un programma che ne rende impossibile l'analisi (e quindi impossibile da interpretare).

errore o eccezione a runtime Errore rilevato durante l'esecuzione del programma.

semantica Il significato di un programma.

errore semantico Un errore in un programma che gli fa eseguire qualcosa di diverso da quello che intendeva il programmatore.

Section 10

Esercizi

aaaa

aaaa