

Esercizio guidato

Settimana 7
15/11/2022

Si ringrazia il Dott. Giacomo Baruzzo per il materiale

Lab 6 – Es 4

Scrivere il modulo **myinput.py** che contenga le seguenti funzioni di utilità per l'acquisizione di dati dall'utente:

- **inputYesNo(message, yes, no)** restituisce True se l'utente ha scritto la stringa fornita come secondo argomento, restituisce False se l'utente ha scritto la stringa fornita come terzo argomento; ignora la differenza tra maiuscole e minuscole (es. di utilizzo: **if inputYesNo("Vuoi continuare? (S/N) ", "S", "N") : # vuole continuare**)
- **inputStringStartingWith(message, startingString)** restituisce una stringa che inizia con la stringa **startingString**; se **startingString** è la stringa vuota, viene restituita la prima stringa digitata dall'utente (anche se è la stringa vuota)
- **inputStringEndingWith(message, endingString)** restituisce una stringa che termina con la stringa **endingString**; se **endingString** è la stringa vuota, viene restituita la prima stringa digitata dall'utente (anche se è la stringa vuota)
- **inputStringContaining(message, substring)** restituisce una stringa che contiene la stringa **substring**; se **substring** è la stringa vuota, viene restituita la prima stringa digitata dall'utente (anche se è la stringa vuota)
- **isDecimalInteger(s)** restituisce True se e solo se la stringa s contiene un numero intero decimale, che ha questo formato: zero o più spazi iniziali, un eventuale segno meno, una o più cifre decimali (da 0 a 9), zero o più spazi finali (quindi, ad esempio, non ci può essere uno spazio tra il segno meno e la prima cifra del numero)
- **inputDecimalInteger(message)** restituisce un numero intero; la funzione deve utilizzare in modo opportuno la funzione **isDecimalInteger**
- **inputPositiveDecimalInteger(message)** restituisce un numero intero positivo; la funzione deve utilizzare in modo opportuno la funzione **inputDecimalInteger**
- **inputNegativeDecimalInteger(message)** restituisce un numero intero negativo; la funzione deve utilizzare in modo opportuno la funzione **inputDecimalInteger**
- **inputNonPositiveDecimalInteger(message)** restituisce un numero intero non positivo; la funzione deve utilizzare in modo opportuno la funzione **inputDecimalInteger**
- **inputNonNegativeDecimalInteger(message)** restituisce un numero intero non negativo; la funzione deve utilizzare in modo opportuno la funzione **inputDecimalInteger**
- **isFloating(s)** restituisce True se e solo se la stringa s contiene un numero decimale in virgola mobile, che ha questo formato: zero o più spazi iniziali, un eventuale segno meno, una o più cifre decimali (da 0 a 9), un eventuale separatore decimale (il carattere "punto") seguito da una o più cifre decimali, un'eventuale lettera "e" (maiuscola o minuscola) seguita da un eventuale segno meno e da una o più cifre decimali, zero o più spazi finali
- **inputFloating(message)** restituisce un numero in virgola mobile; la funzione deve utilizzare in modo opportuno la funzione **isFloating**

Esercizio

Scrivere il modulo **myinput.py** che contenga le seguenti funzioni di utilità per l'acquisizione di dati dall'utente:

- **inputYesNo(message, yes, no)**
- **inputStringStartingWith(message, startingString)**
- **inputStringEndingWith(message, endingString)**
- **inputStringContaining(message, substring)**
- **isDecimalInteger(s)**
- **inputDecimalInteger(message)**
- **inputPositiveDecimalInteger(message)**
- **inputNegativeDecimalInteger(message)**
- **inputNonPositiveDecimalInteger(message)**
- **inputNonNegativeDecimalInteger(message)**
- **isFloating(s)**
- **inputFloating(message)**

Tutte le funzioni di tipo **input...** devono chiedere ripetutamente il dato all'utente finché questo non rispetta le specifiche della funzione, riproponendo il messaggio **message** (senza visualizzare messaggi d'errore). Tutte le funzioni di tipo **is...**, invece, non devono avere alcuna interazione con l'utente (né in input né in output).

Esercizio

Scrivere il modulo **myinput.py** che contenga le seguenti funzioni di utilità per l'acquisizione di dati dall'utente:

- **inputYesNo(message, yes, no)**
- **inputStringStartingWith(message, startingString)**
- **inputStringEndingWith(message, endingString)**
- **inputStringContaining(message, substring)**
- **isDecimalInteger(s)**
- **inputDecimalInteger(message)**
- **inputPositiveDecimalInteger(message)**
- **inputNegativeDecimalInteger(message)**
- **inputNonPositiveDecimalInteger(message)**
- **inputNonNegativeDecimalInteger(message)**
- **isFloating(s)**
- **inputFloating(message)**

Tutte le funzioni di tipo **input...** devono chiedere ripetutamente il dato all'utente finché questo non rispetta le specifiche della funzione, riproponendo il messaggio **message** (senza visualizzare messaggi d'errore). Tutte le funzioni di tipo **is...**, invece, non devono avere alcuna interazione con l'utente (né in input né in output).

Il problema

isDecimalInteger(s) restituisce *True* se e solo se la stringa *s* contiene un numero intero decimale, che ha questo formato:

- zero o più spazi iniziali
- un eventuale segno meno
- una o più cifre decimali (da 0 a 9)
- zero o più spazi finali

Se vogliamo vedere *s* come blocchi di caratteri:

Num. spazi ≥ 0

Segno “-” (opzionale)

Num. cifre ≥ 1

Num. spazi ≥ 0

Esempi

- Stringhe considerate valide:

“1024”, “ 1024”, “1024 ”, “ 1024 ”,
“ -1024”, “-1024 ”

- Stringhe considerate NON valide:

- “- 1024” : spazio tra il segno e il numero
- “+1024” : carattere “non cifra”
- “10 24”, : spazio tra le cifre
- “102b4” : carattere “non cifra”
- “1024 a” : carattere “non cifra”

Idea

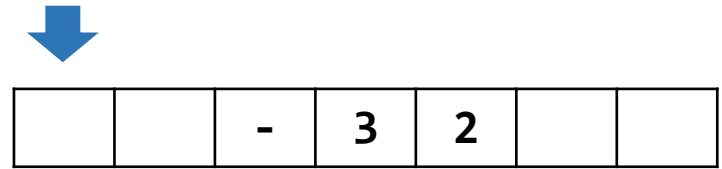
- s è una stringa
- Devo analizzare i singoli caratteri di s ...
- (quindi non conviene convertire s in un numero!)
- ...e controllare che rispettino le regole imposte

Algoritmo

- **Controllare la presenza di zero o più spazi iniziali:** scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- **Controllare la presenza di un eventuale segno meno**
- **Controllare la presenza di una o più cifre decimali (da 0 a 9):** scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- **Controllare la presenza di zero o più spazi finali:** se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> **ritornare *True***
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> **ritornare *False***

Algoritmo

Esempio 1



- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*


2 spazi iniziali



		-	3	2		
--	--	---	---	---	--	--

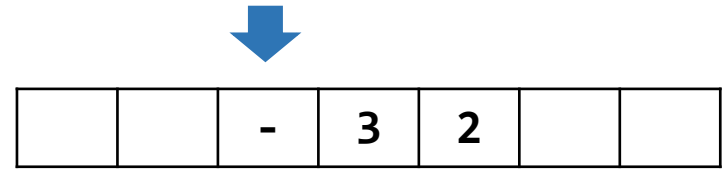
Algoritmo


Esempio 1

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi 
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*

Algoritmo

Esempio 1



- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno 
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!) *(this item is faded)*
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa *(this item is faded)*
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*


2 cifre decimali



		-	3	2		
--	--	---	---	---	--	--

Algoritmo

Esempio 1

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno 
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*





Algoritmo

Esempio 1

		-	3	2		
--	--	---	---	---	--	--

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno ✓
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa ✓
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*

sono in fondo alla stringa



		-	3	2		
--	--	---	---	---	--	--

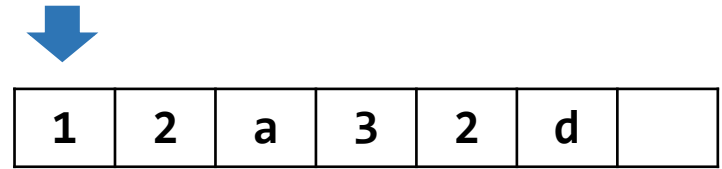
Algoritmo

Esempio 1

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno ✓
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True* ✓
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*

Algoritmo

Esempio 2



- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*



1	2	a	3	2	d	
---	---	---	---	---	---	--

Algoritmo

Esempio 2

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*



1	2	a	3	2	d	
---	---	---	---	---	---	--

Algoritmo

Esempio 2

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*



1	2	a	3	2	d	
---	---	---	---	---	---	--

Algoritmo

Esempio 2

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*



carattere diverso dallo spazio dopo le cifre!



1	2	a	3	2	d	
---	---	---	---	---	---	--

Algoritmo

Esempio 2

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa finché trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!)
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False*





non sono in fondo alla stringa



1	2	a	3	2	d	
---	---	---	---	---	---	--

Algoritmo

Esempio 2

- Controllare la presenza di zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi
- Controllare la presenza di un eventuale segno meno
- Controllare la presenza di una o più cifre decimali (da 0 a 9): scorro la stringa fin trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un "carattere cifra"!) 
- Controllare la presenza di zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa
- Se sono arrivato all'ultimo carattere della stringa, allora la stringa rappresenta un numero decimale -> ritornare *True*
- Se NON sono arrivato all'ultimo carattere della stringa, allora la stringa NON rappresenta un numero decimale -> ritornare *False* 

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Zero o più spazi iniziali: scorro la stringa dall'inizio e mi posiziono sul primo carattere dopo l'eventuale sequenza di spazi

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

il valore finale di i corrisponde al primo carattere della stringa che NON è uno spazio (eventualmente i = 0)

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Eventuale segno meno

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Una o più cifre decimali: scorro la stringa fintanto che trovo caratteri corrispondenti a cifre (NB: deve esserci almeno un “carattere cifra”!)

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Zero o più spazi finali: se dopo le (eventuali) cifre ho caratteri diversi dallo spazio, NON scorro più la stringa

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Se $i == \text{len}(s)$ vuol dire che sono riuscito a scandire tutta la stringa rispettando il formato.

Devo solo controllare che ci sia almeno una cifra numerica (potrebbero essere tutti spazi)

Dall'algoritmo al codice

```
def isDecimalInteger(s) :  
    atLeastOneDigit = False  
    i = 0  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i < len(s) and s[i] == '-' :  
        i += 1  
    while i < len(s) :  
        if not s[i].isdigit() :  
            break  
        atLeastOneDigit = True  
        i += 1  
    while i < len(s) :  
        if s[i] != ' ' :  
            break  
        i += 1  
    if i == len(s) and atLeastOneDigit:  
        return True  
    else:  
        return False
```

Se sono arrivato all'ultimo carattere della stringa -> ritornare *True*
Se NON sono arrivato all'ultimo carattere della stringa -> ritornare *False*

Soluzione 1

```
def isDecimalInteger(s) :
    atLeastOneDigit = False
    i = 0
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    if i < len(s) and s[i] == '-' :
        i += 1
    while i < len(s) :
        if not s[i].isdigit() :
            break
        atLeastOneDigit = True
        i += 1
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    if i == len(s) and atLeastOneDigit:
        return True
    else:
        return False
```

Soluzione 2

```
def isDecimalInteger(s) :
    atLeastOneDigit = False
    i = 0
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    if i < len(s) and s[i] == '-' :
        i += 1
    while i < len(s) :
        if not s[i].isdigit() :
            break
        atLeastOneDigit = True
        i += 1
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    return i == len(s) and atLeastOneDigit
```

Soluzione 2: la novità

```
...  
if i == len(s) and atLeastOneDigit:  
    return True  
else:  
    return False
```

```
...  
return i == len(s) and atLeastOneDigit
```

```
if condizione :  
    return True  
else:  
    return False
```

Si può semplicemente scrivere:

```
return condizione
```

Soluzione 2

```
def isDecimalInteger(s) :
    atLeastOneDigit = False
    i = 0
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    if i < len(s) and s[i] == '-' :
        i += 1
    while i < len(s) :
        if not s[i].isdigit() :
            break
        atLeastOneDigit = True
        i += 1
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    return i == len(s) and atLeastOneDigit
```

Soluzione 3

```
def isDecimalInteger(s) :
    atLeastOneDigit = False
    i = 0
    i = checkSpaces(i, s)
    if i < len(s) and s[i] == '-' :
        i += 1
    while i < len(s) :
        if not s[i].isdigit() :
            break
        atLeastOneDigit = True
        i += 1
    i = checkSpaces(i, s)
    return i == len(s) and atLeastOneDigit

def checkSpaces(i, s):
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    return i
```

Soluzione 3

```
def isDecimalInteger(s) :
    atLeastOneDigit = False
    i = 0
    i = checkSpaces(i, s)
    if i < len(s) and s[i] == '-' :
        i += 1
    while i < len(s) :
        if not s[i].isdigit() :
            break
        atLeastOneDigit = True
        i += 1
    i = checkSpaces(i, s)
    return i == len(s) and atLeastOneDigit

def checkSpaces(i, s):
    while i < len(s) :
        if s[i] != ' ' :
            break
        i += 1
    return i
```

Soluzione 4

```
def isDecimalInteger(s) :
    atLeastOneDigit = False
    i = 0
    i = checkSpaces(i, s)
    if i < len(s) and s[i] == '-' :
        i += 1
    while i < len(s) :
        if not s[i].isdigit() :
            break
        atLeastOneDigit = True
        i += 1
    i = checkSpaces(i, s)
    return i == len(s) and atLeastOneDigit

def checkSpaces(i, s):
    import re
    while i < len(s) :
        if re.search("\S", s[i]):
            break
        i += 1
    return i
```