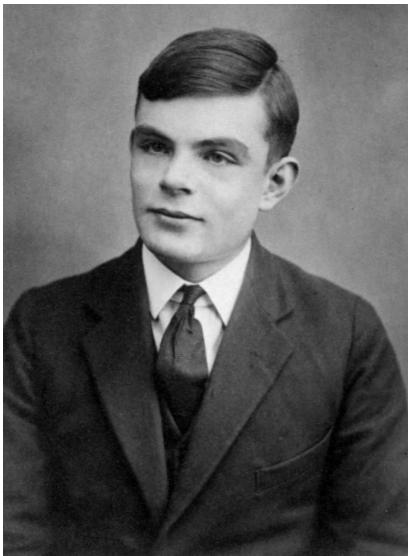




**Text Processing**  
Chris Piech and Mehran Sahami



# Decryption



Alan Turing



Joan Clarke





# Translation

*The spirit is willing but the flesh is weak.*



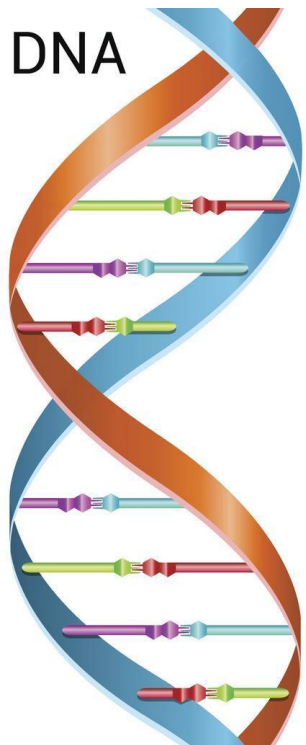
(Russian)



*The vodka is good but the meat is rotten.*



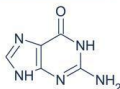
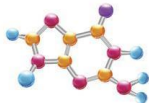
# DNA Analysis



## NITROGENOUS BASES



Adenine



Guanine



Thymine



Cytosine



# 23andMe

# A Problem That Needs Solving



mPedigree

Bringing Quality to Life









424220128



mPedigree

Bringing Quality to Life



Manufacturer

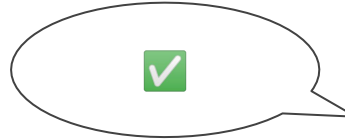


424220128



mPedigree

Bringing Quality to Life





Counterfeiter

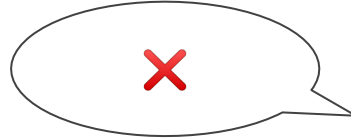


424320128



mPedigree

Bringing Quality to Life





424220128



### Valid numbers

97620000

54980001

.

.

.

424220128

.

.

.

28675959

15895960





# How do we generate these numbers?

Valid numbers

97620000

54980001

.

.

.

424220128

.

.

.

28675959

15895960

# Lists Review



```
def main():  
    lst = [1, 2, 3]  
    print(lst)
```



# Text in Python





Text in Python is represented as a  
**String**



# Revisiting Strings

name = “Brahm”



# Revisiting Strings

name = “Brahm”





# Revisiting Strings

```
name = input("Name: ")
```



# Revisiting Strings

```
name = input("Name: ")
```

```
$ python my_program.py  
Name:
```



# Revisiting Strings

```
name = input("Name: ")
```

```
$ python my_program.py  
Name: Brahm
```

name  → "Brahm"



# Revisiting Strings

```
print("Hi, " + name + "!")
```

```
$ python my_program.py  
Name: Brahm  
Hi, Brahm!
```

name  → "Brahm"



## Revisiting Strings

num\_str = "42"

num = 42

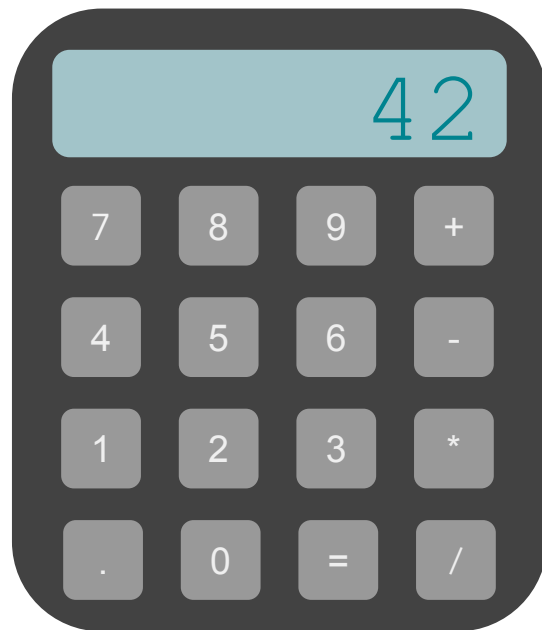
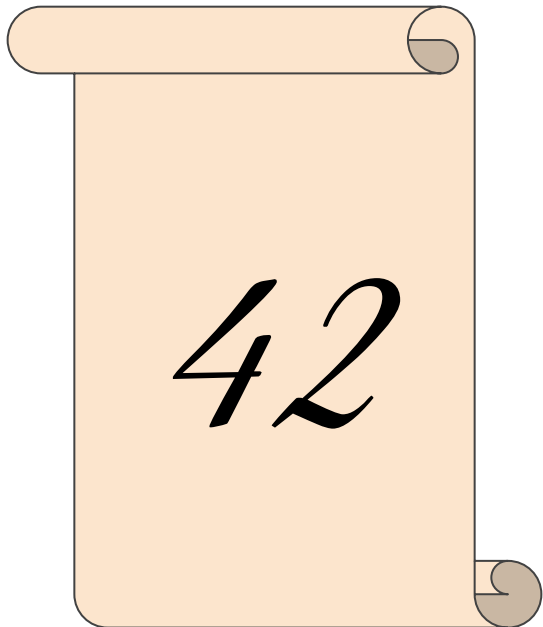




## Revisiting Strings

`num_str = "42"`

`num = 42`

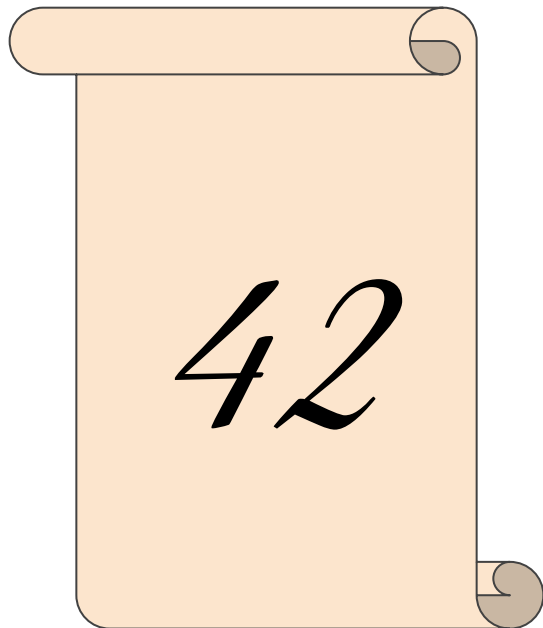




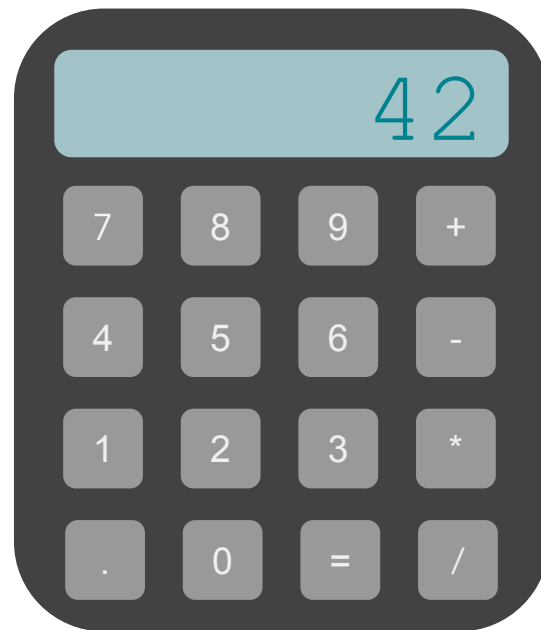
# Revisiting Strings

num\_str = "42"

num = 42



`int(num_str)`

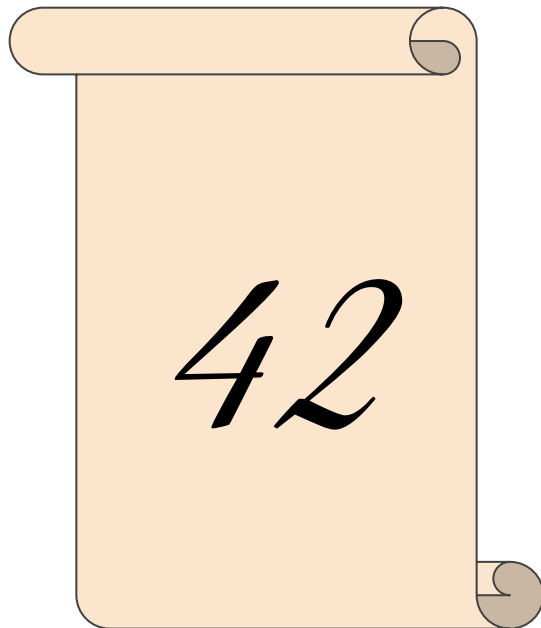




# Revisiting Strings

num\_str = "42"

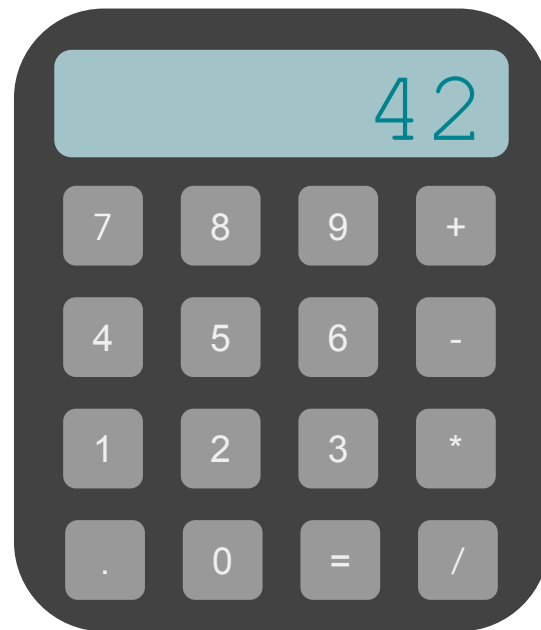
num = 42



`int(num_str)`



`str(num)`



**We've been shortchanging strings a little bit.**



## Strings are *sequences*

S = “lightsaber”



# Strings are *sequences*

S = “lightsaber”

0 1 2 3 4 5 6 7 8 9





# Strings are *sequences*

S = “lightsaber”

0 1 2 3 4 5 6 7 8 9



ch = s[2]





# Strings are *sequences*

`s = "lightsaber"`

0 1 2 3 4 5 6 7 8 9



`ch = s[len(s) - 1]`



# Strings are *sequences*

`s = "lightsaber"`

0 1 2 3 4 5 6 7 8 9



`ch = s[-1]`



## Strings are *sequences*

`s` = “lightsaber”

0 1 2 3 4 5 6 7 8 9

A diagram illustrating string indexing for the string "lightsaber". The characters are arranged in a row, with indices 0 through 9 written below them. The characters 'i', 'g', 'h', 't', 's', and 'a' are highlighted in green. A blue horizontal line with small vertical end caps is drawn below the indices 1 through 6, indicating the slice s[1:7].

`part` = `s[1:7]`



## Strings are *sequences*

`s` = “`l i g h t s a b e r`”

0 1 2 3 4 5 6 7 8 9



`part` = `s[1:]`



# Strings are *sequences*

`S` = “`l i g h t s a b e r`”


0 1 2 3 4 5 6 7 8 9



`part` = `s[:7]`




```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

A teal arrow pointing to the right, indicating the start of the code block.

```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

example  → "Python"

```
$ python3 strings.py
```




```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

```
example  → "Python"  
length  → 6
```

```
$ python3 strings.py
```






```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

example  → "Python"  
length  → 6


```
$ python3 strings.py  
6
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

```
example  → "Python"  
length  → 6  
first_char  → "p"
```

```
$ python3 strings.py  
6
```



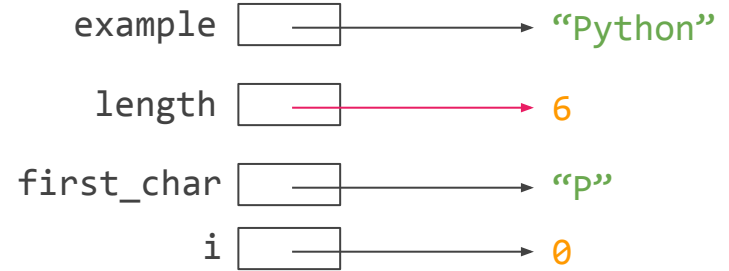
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

```
example [ ] → "Python"  
length [ ] → 6  
first_char [ ] → "p"
```

```
$ python3 strings.py  
6  
P
```



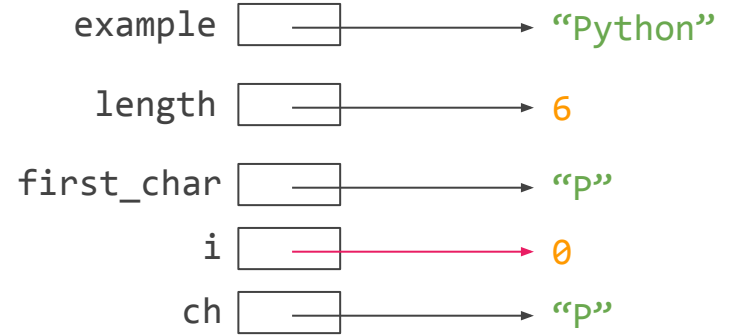
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P
```



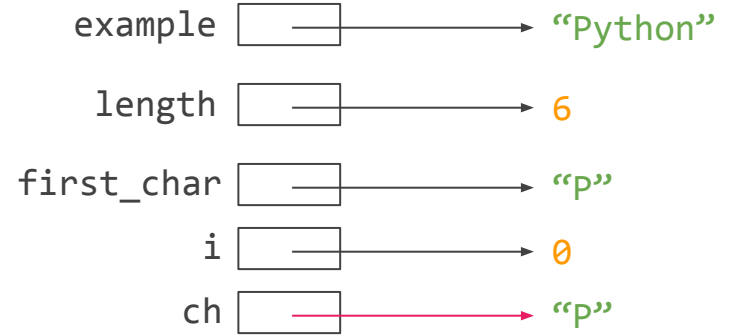
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

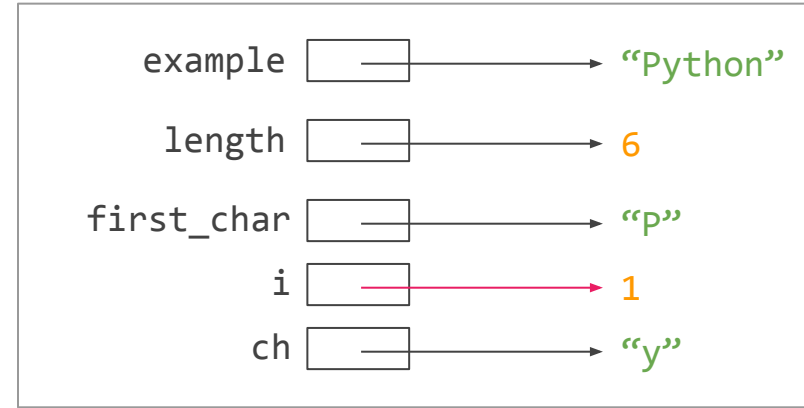


example  → "Python"  
length  → 6  
first\_char  → "p"  
i  → 1  
ch  → "p"

```
$ python3 strings.py  
6  
P  
P
```




```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

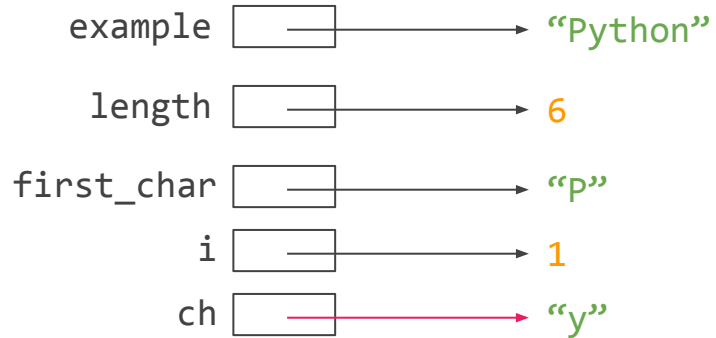


```
$ python3 strings.py  
6  
P  
P
```





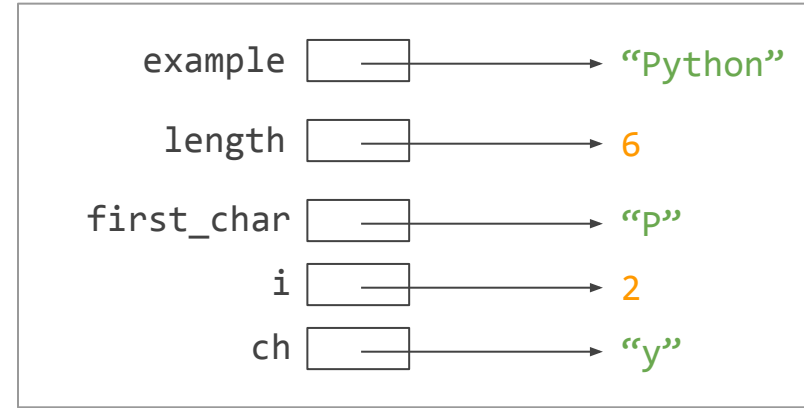
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```




```
$ python3 strings.py  
6  
P  
P  
y
```



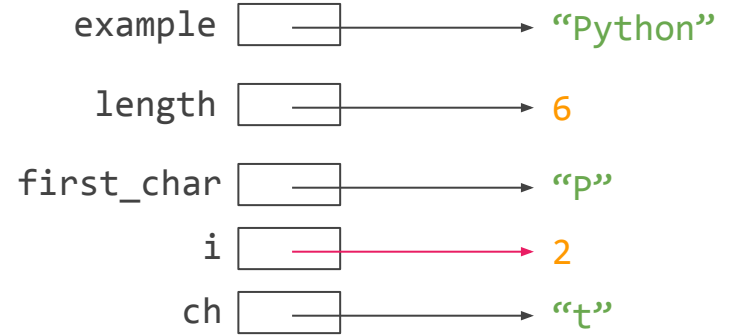
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```




```
$ python3 strings.py  
6  
P  
P  
y
```



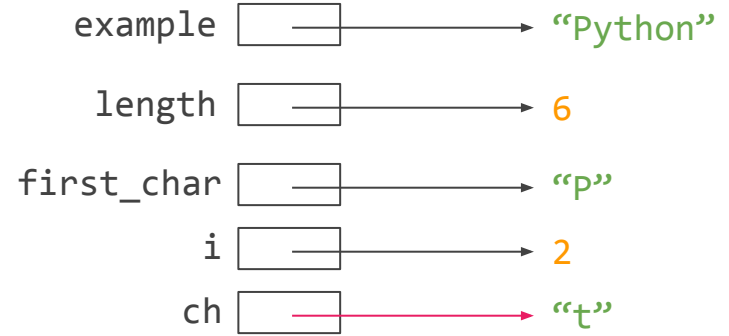
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
y
```



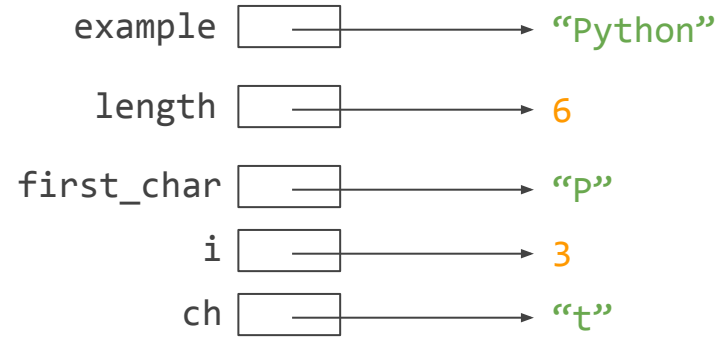
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
y  
t
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
y  
t
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



example  → "Python"

length  → 6

first\_char  → "p"

i  → 3

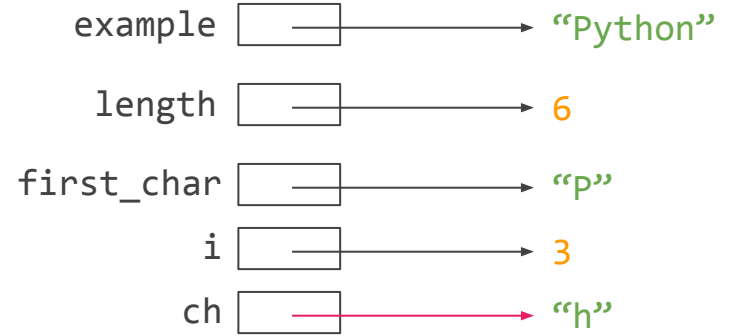
ch  → "h"

```
$ python3 strings.py
```

```
6  
P  
P  
y  
t
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
y  
t  
h
```




```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



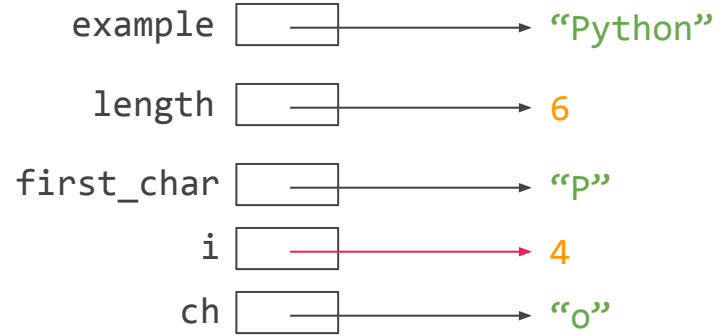
example  → "Python"  
length  → 6  
first\_char  → "p"  
i  → 4  
ch  → "h"

```
$ python3 strings.py  
6  
P  
P  
Y  
t  
h
```






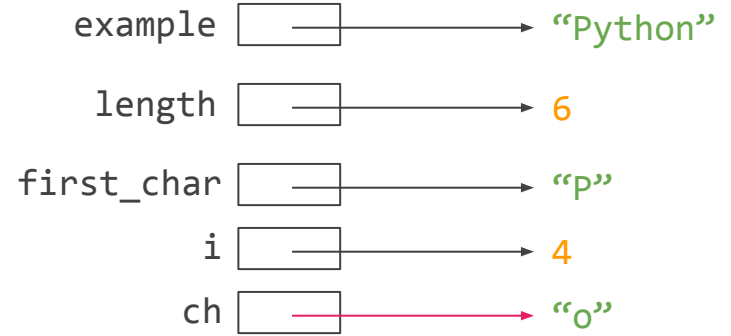
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```




```
$ python3 strings.py  
6  
P  
P  
y  
t  
h
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
y  
t  
h  
o
```



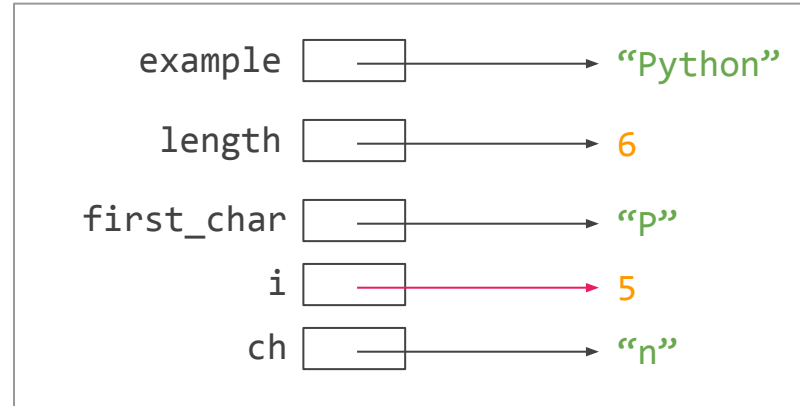
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

```
example  → "Python"  
length  → 6  
first_char  → "p"  
i  → 5  
ch  → "h"
```


```
$ python3 strings.py  
6  
P  
P  
Y  
T  
H  
O
```



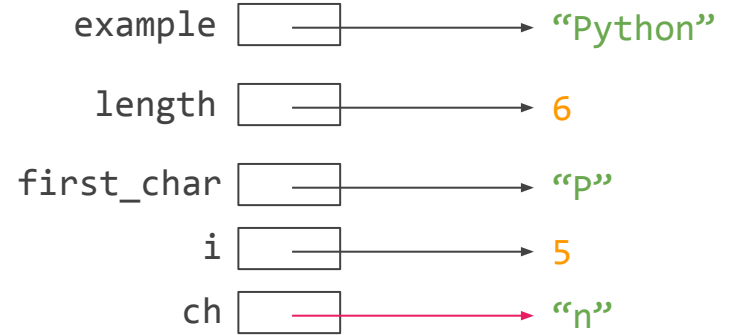
```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
y  
t  
h  
o
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```



```
$ python3 strings.py  
6  
P  
P  
Y  
t  
h  
o  
n
```



```
def main():  
    example = "Python"  
  
    length = len(example)  
    print(length)  
  
    first_char = example[0]  
    print(first_char)  
  
    for i in range(length):  
        ch = example[i]  
        print(ch)
```

```
$ python3 strings.py  
6  
P  
P  
y  
t  
h  
o  
n  
  
$
```

# Under The Hood



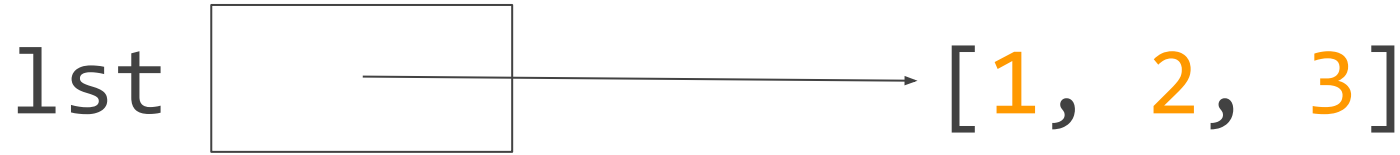
# Mutation and reassignment







# Mutation and reassignment



Mutation

```
lst.append(4)
```





# Mutation and reassignment



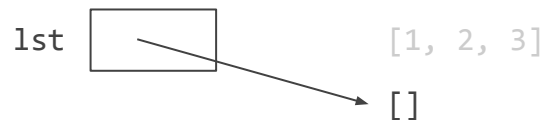
Mutation

```
lst.append(4)
```



Reassignment

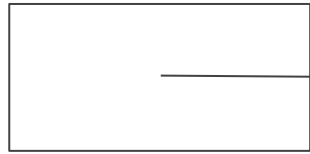
```
lst = []
```





# Mutation and reassignment

S



“brahm”



# Mutation and reassignment

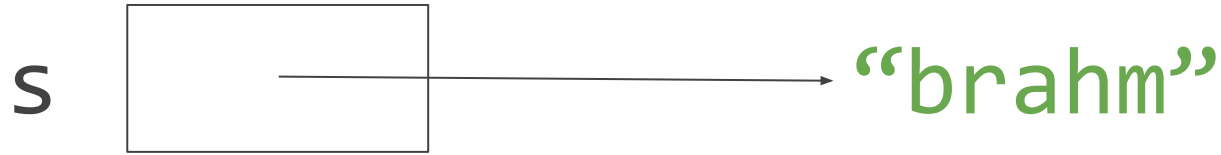


Mutation

`s[0] = "B"`



# Mutation and reassignment



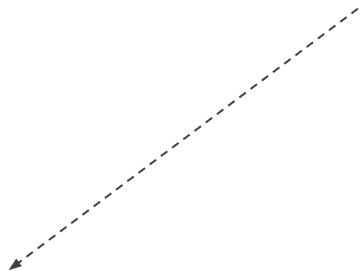
Mutation

```
s[0] = "B"
```

```
TypeError: 'str' object does not support item  
assignment
```



# Mutation and reassignment



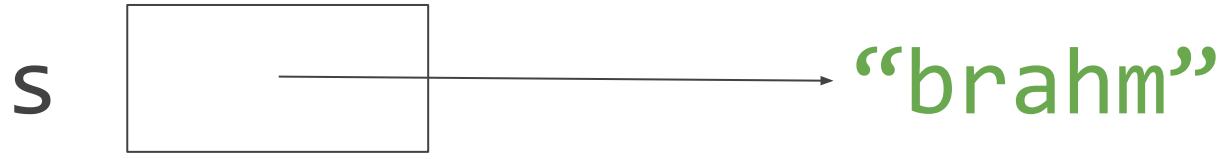
Mutation

s[0] = “B”





# Mutation and reassignment

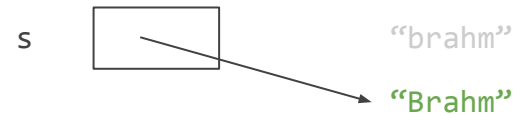


Mutation

`s[0] = "B"`

Reassignment

`s = "B" + s[1:]`





## Mutation and reassignment

Strings **can't be mutated**





# Mutation and reassignment

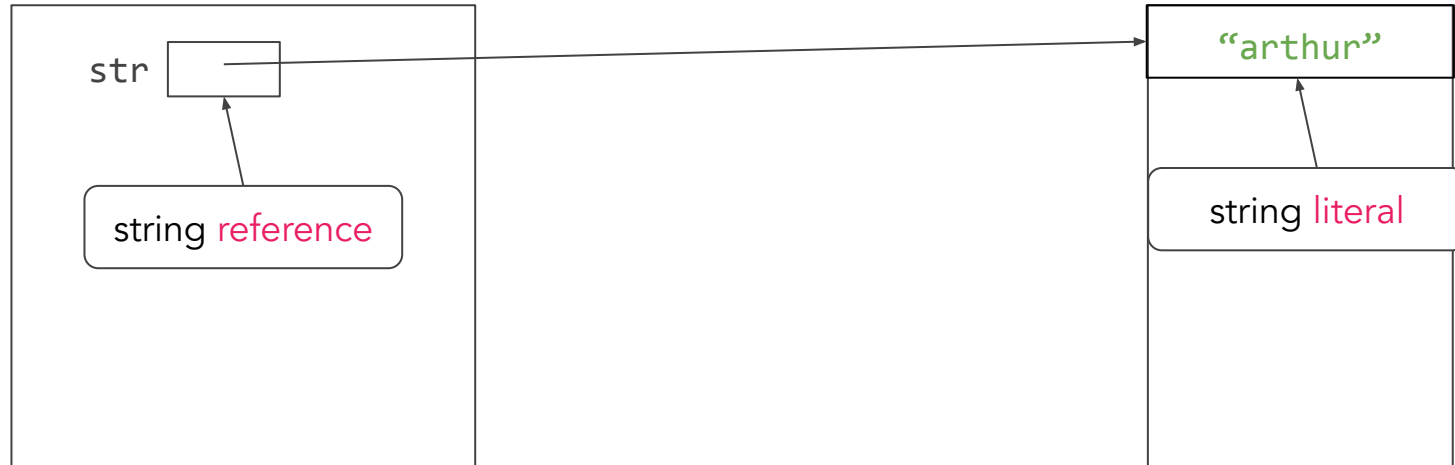
Strings are **immutable** 



# Changing Strings

An important nuance: string **literals** are immutable

```
str = "arthur"
```

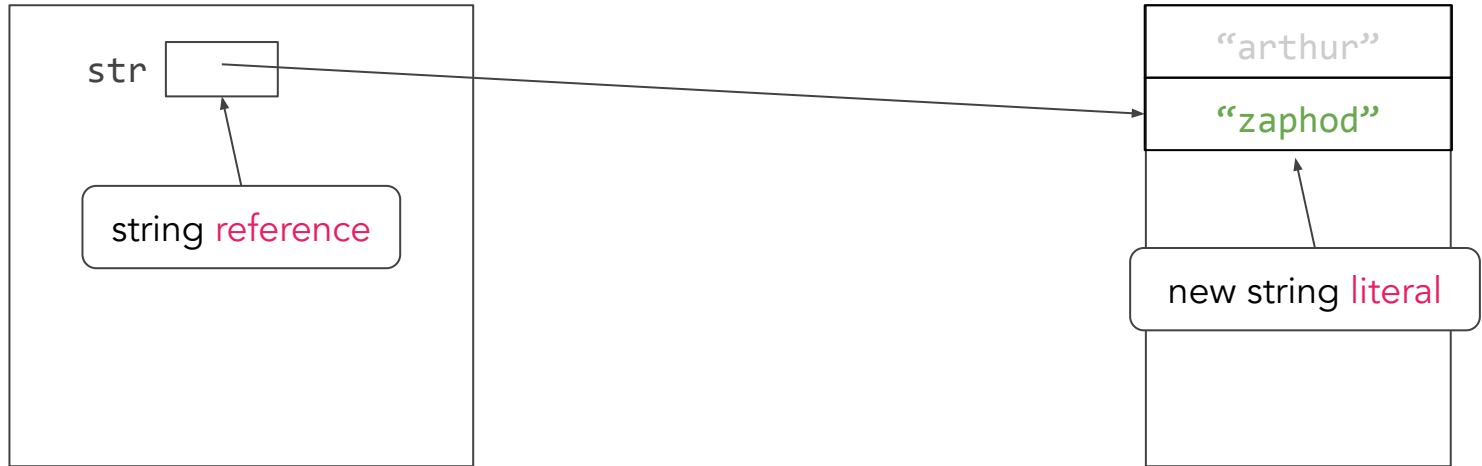




# Changing Strings

...but references aren't!

```
str = "zaphod"
```

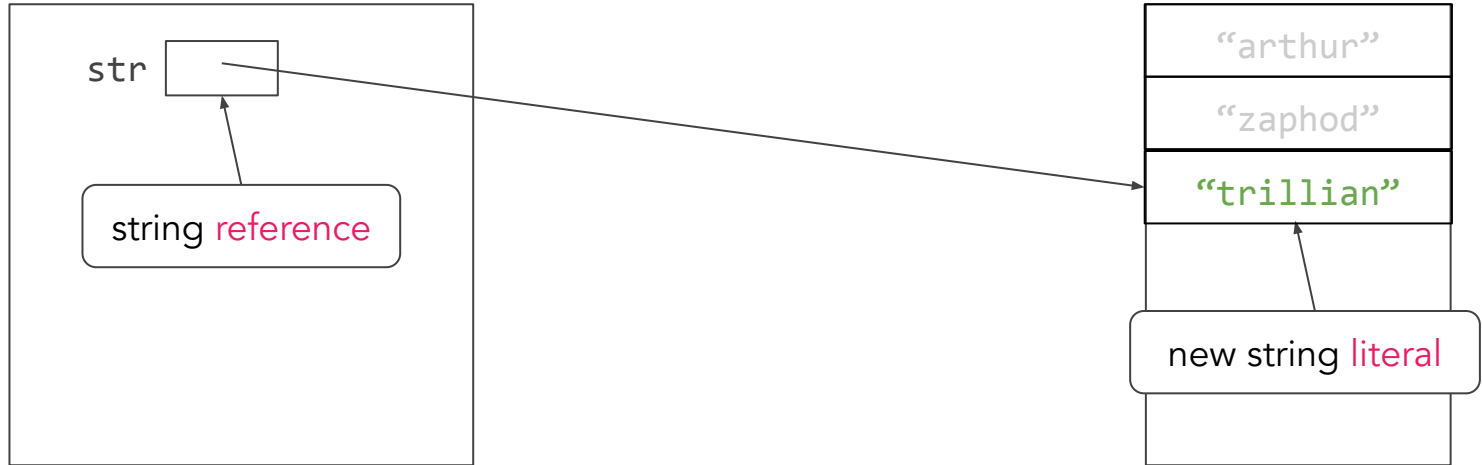




# Changing Strings

...but references aren't!

```
str = "trillian"
```

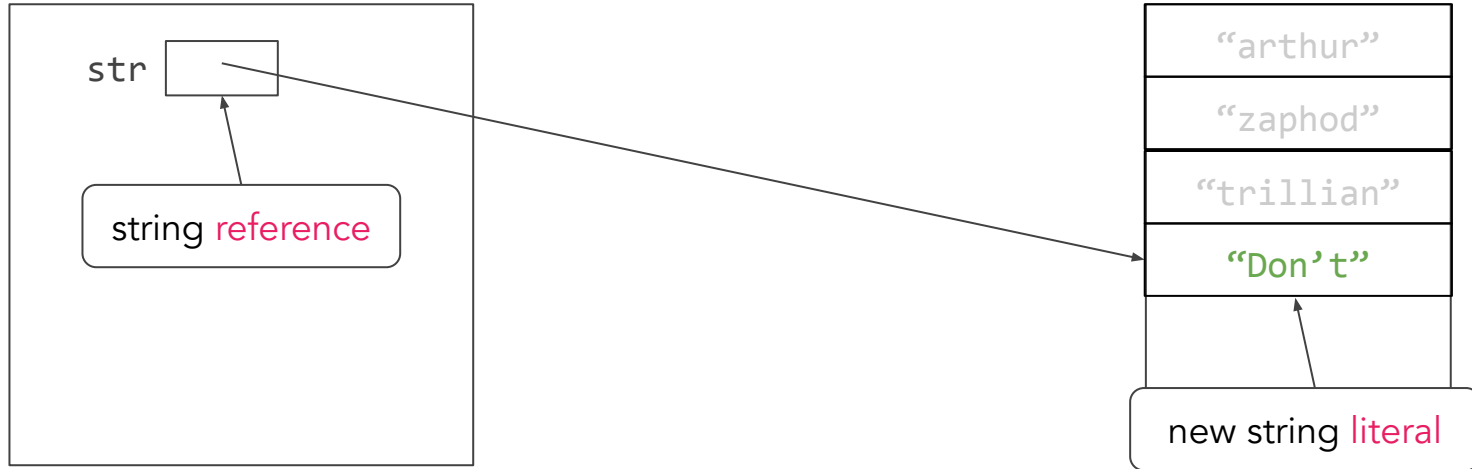




# Changing Strings

...but references aren't!

```
str = "Don't"
```

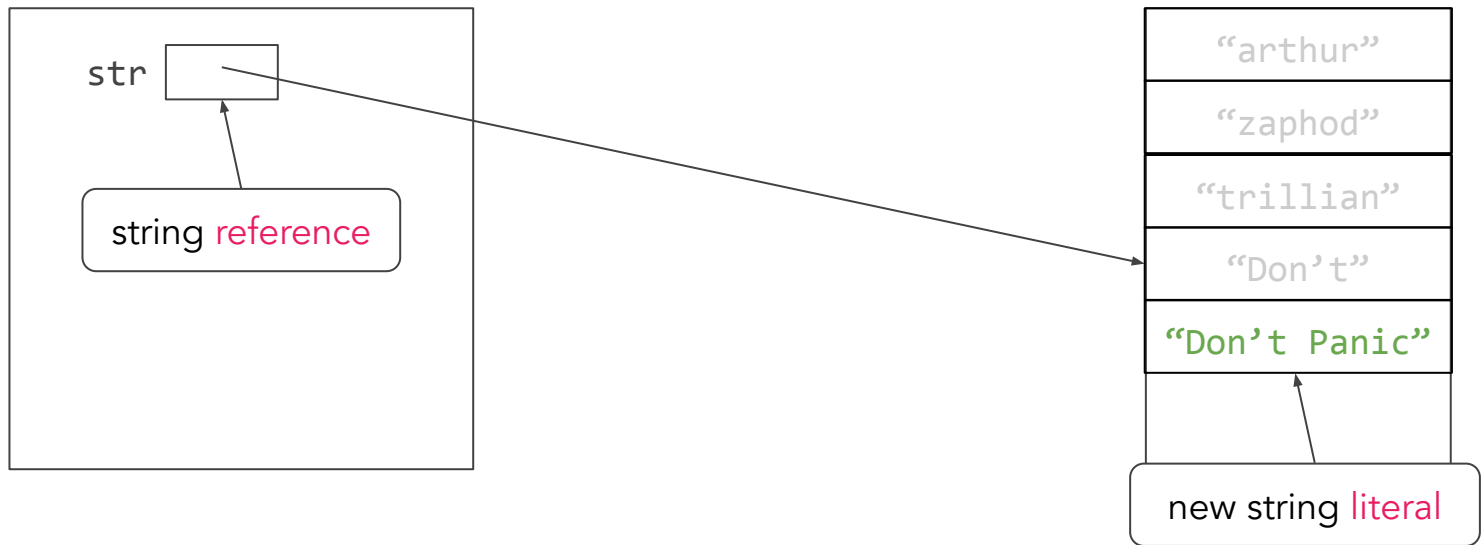




# Changing Strings

## ...but references aren't!

```
str = str + " Panic"
```





## Changing Strings

The only way to change a string variable is to **reassign** it



## Changing Strings

The only way to change a string variable is to make a **new string**



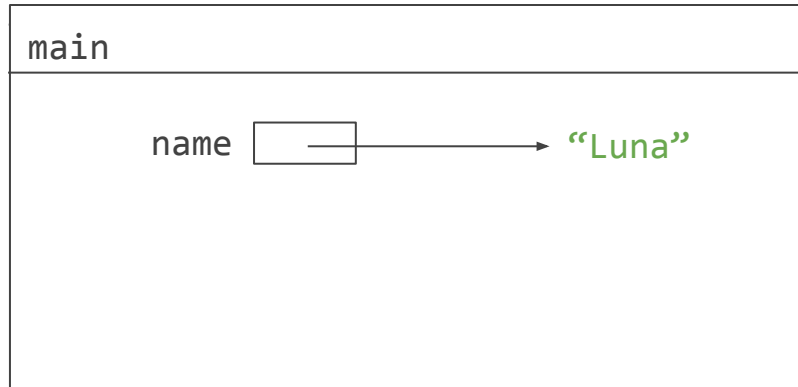




```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```



```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```





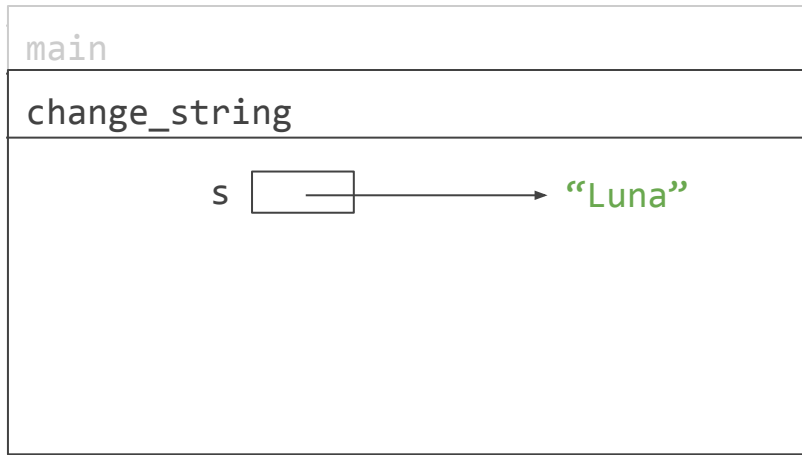
```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```



main
change_string



```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```



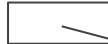


```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```

main

change\_string

s



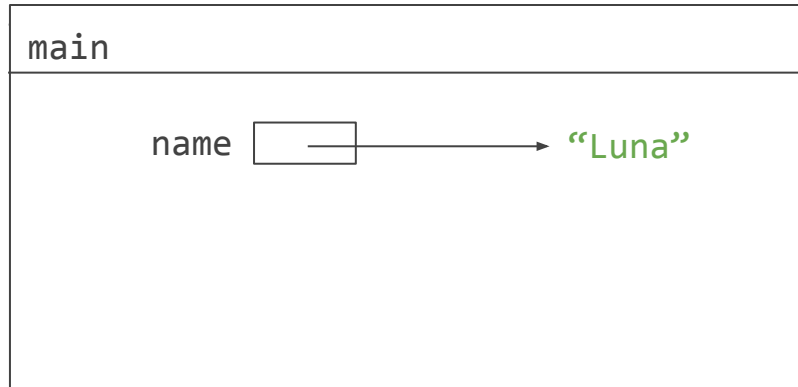
"Luna"

"Hagrid"



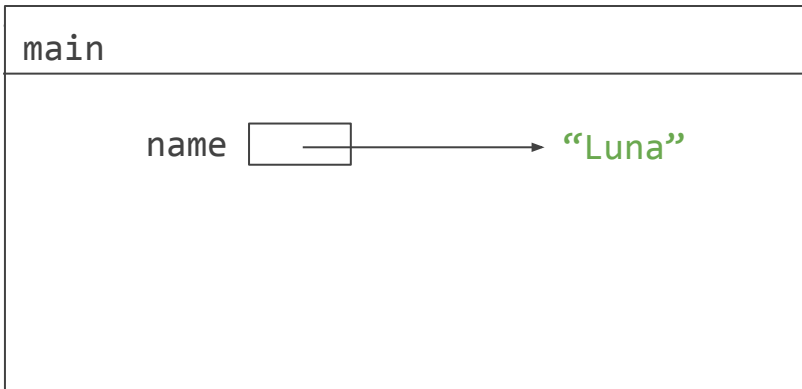


```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```





```
def change_string(s):  
    s = "Hagrid"  
  
def main():  
    name = "Luna"  
    change_string(name)
```



Immutability guarantees that string parameters **won't change**

# String Utilities





lst.append

lst.insert

lst.extend

lst.index

lst.copy

lst.clear

lst.pop

lst.remove



Functions like **append**, **extend**,  
**copy** and **pop** represent  
**behaviours** of a list, or things that  
a list knows how to do.



Strings also have **behaviours**  
(things they know how to do)  
which are represented by  
functions



# ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]





# String Functions

s = “ So long and thanks for all the fish ”



# String Functions

s = “ So long and thanks for all the fish ”

```
>>> s.upper()
```

```
“ SO LONG AND THANKS FOR ALL THE FISH ”
```

```
>>> s.lower()
```

```
“ so long and thanks for all the fish ”
```



# String Functions

s = “ So long and thanks for all the fish ”

```
>>> s.replace(“a”, “e”)
“ So long end thenks for ell the fish ”
```

```
>>> s.replace(“s”, “”)
“ o long and thank for all the fih ”
```





# String Functions

s = “ So long and thanks for all the fish ”

```
>>> s.find("n")
```

```
6
```

```
>>> s.find("x")
```

```
-1
```



# String Functions

s = “So long and thanks for all the fish”

```
>>> s.strip()
```

```
“So long and thanks for all the fish”
```



# String Functions

s = “ So long and thanks for all the fish ”

```
>>> s.split()  
[“So”, “long”, “and”, “thanks”, “for”,  
“all”, “the”, “fish”]
```



# String Functions

names = "Bruce,Diana,Victor,Barry,Clark,Arthur,Hal"

```
>>> names.split(",")  
["Bruce", "Diana", "Victor", "Barry",  
"Clark", "Arthur", "Hal"]
```



```
>>> chant = "wakanda forever"  
>>> stop_balrog = "YOU SHALL NOT PASS"  
>>> spaces = " "  
>>> number = "42"
```

```
>>> chant.startswith("wak")
```

```
True
```

```
>>> stop_balrog.startswith("you")
```

```
False
```

```
>>> chant.endswith("ver")
```

```
True
```

```
>>> chant.title()
```

```
"Wakanda Forever"
```

```
>>> chant.islower()
```

```
True
```

```
>>> spaces.isspace()
```

```
True
```

```
>>> number.isdigit()
```

```
True
```



# String Functions

`s.upper()`

`s.lower()`

`s.replace()`

`s.strip()`

`s.title()`

Because strings are immutable, these functions **don't change** the string and return a new string instead.



# String Functions

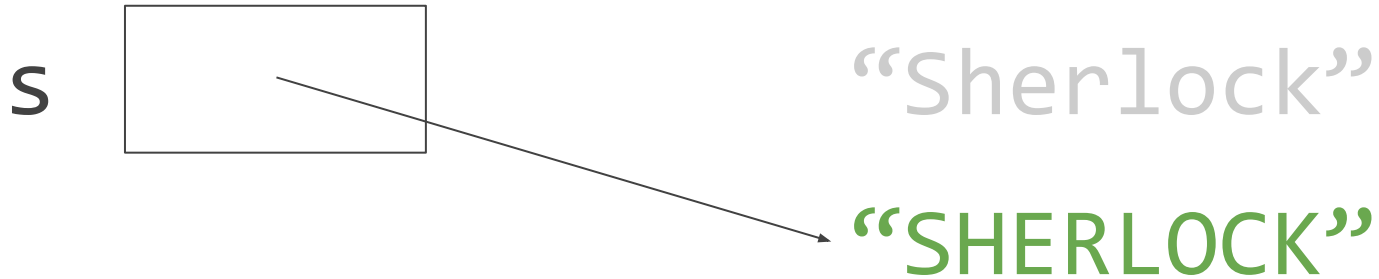
`s = "Sherlock"`





# String Functions

```
s = s.upper()
```





# How to Process A String

Processing a string involves transforming  
or inspecting **the contents of the string**



```
for i in range(len(s)):
    char = s[i]
    # process char
```

```
for i in range(len(s)):
    char = s[i]
    # process char
```

---

```
for char in s:
    # process char
```

```
for i in range(len(s)):
    char = s[i]
    # process char
```

If you need both the  
index (i) and the  
character (char)

```
for char in s:
    # process char
```

If you need just the  
character (char)

# Reversing a string



# Reversing a String

J u l i e

0 1 2 3 4



# Reversing a String

J u l i e

0 1 2 3 4

J

0





# Reversing a String

J u l i e

0 1 2 3 4

u J

0 1



# Reversing a String

J u l i e

0 1 2 3 4

l u J

0 1 2



# Reversing a String

J u l i e

0 1 2 3 4

i l u J

0 1 2 3



# Reversing a String

J u l i e

0 1 2 3 4

e i l u J

0 1 2 3 4



# Reversing a String

J u l i e

e i l u J

We constructed a reversed string by **going over the characters** in the original string, and **inserting them at the start** of the reversed string



```
def reverse_string(s):  
    pass
```

Let's write a function to do that!



```
def reverse_string(s):  
    for i in range(len(s)):  
        ch = s[i]
```

First, go over the characters in the original string



```
def reverse_string(s):
```

```
    for i in range(len(s)):
```

```
        ch = s[i]
```

```
        # ch needs to be inserted at the start of the
```

```
        # reversed string
```

Now we have a character, what do we do it?





```
def reverse_string(s):  
    reverse = ""  
    for i in range(len(s)):  
        ch = s[i]  
        # ch needs to be inserted at the start of the  
        # reversed string
```

First, make a variable that stores the reversed string...



```
def reverse_string(s):  
    reverse = ""  
    for i in range(len(s)):  
        ch = s[i]  
        reverse = ch + reverse
```

...and then insert ch at the beginning of reverse.



```
def reverse_string(s):  
    reverse = ""  
    for i in range(len(s)):  
        ch = s[i]  
        reverse = ch + reverse  
    return reverse
```

When we've gone through all of the characters, **return** the reversed string!



```
def reverse_string(s):  
    reverse = ""  
    for i in range(len(s)):  
        ch = s[i]  
        reverse = ch + reverse  
    return reverse
```

Now, notice that the only place we use the `i` variable is to get a character from the string...



```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse
```

...so we can condense our **for** loop into the simpler for-each style.



```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



main

name  → "Julie"



```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



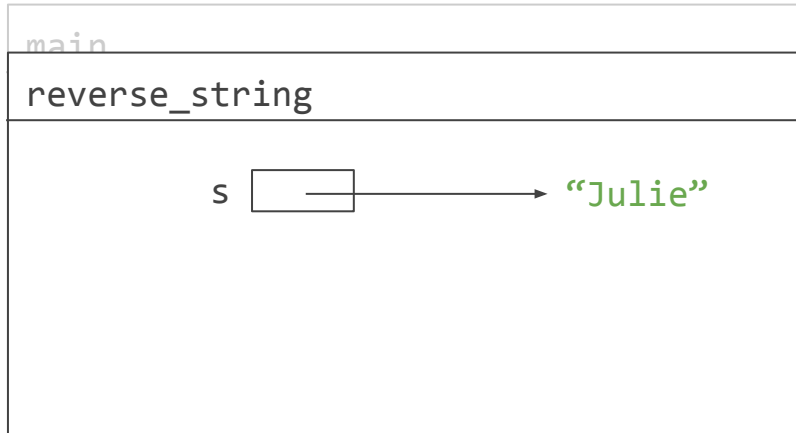
main

reverse\_string



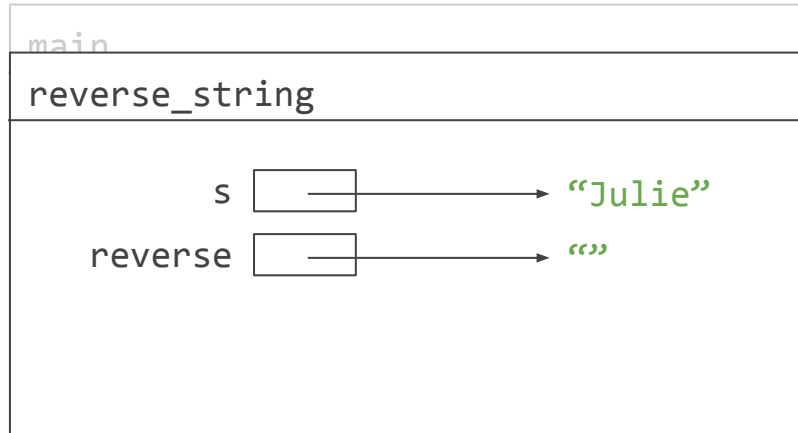


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



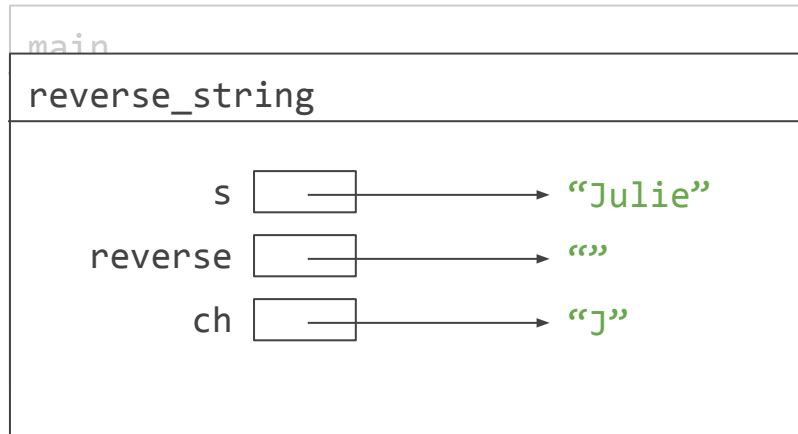


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



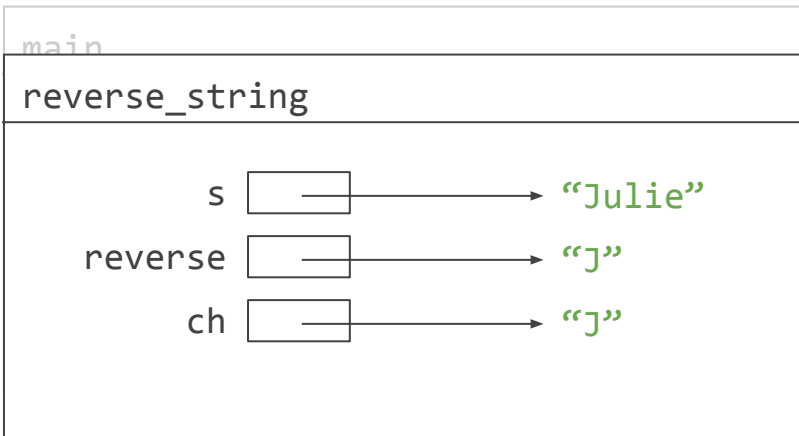


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



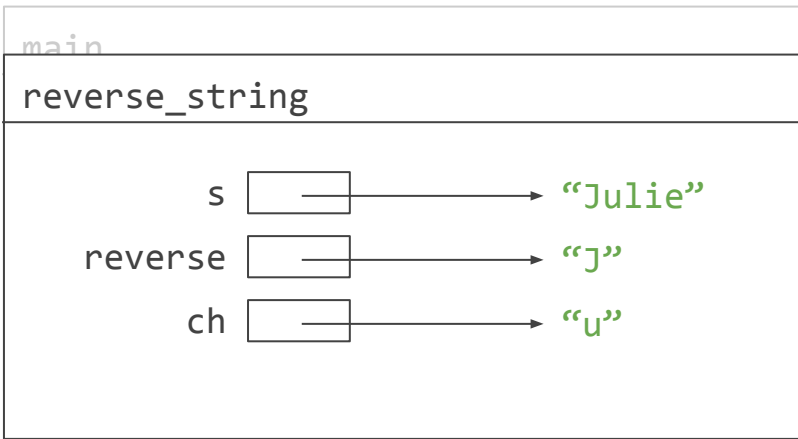


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



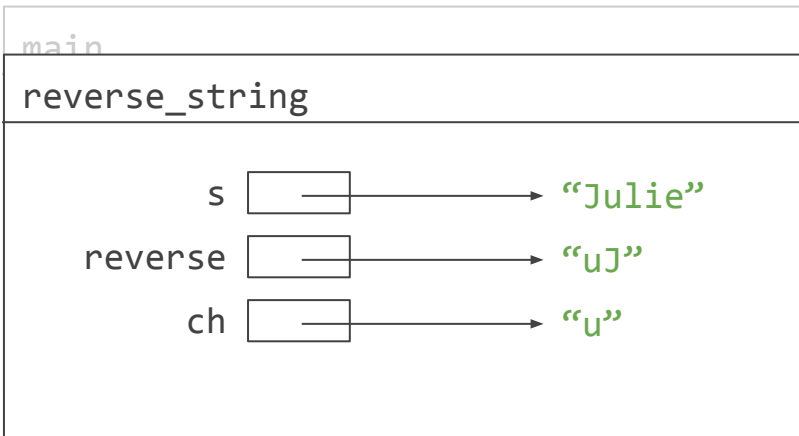


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```





```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```

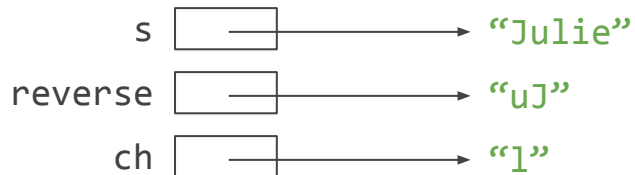




```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```

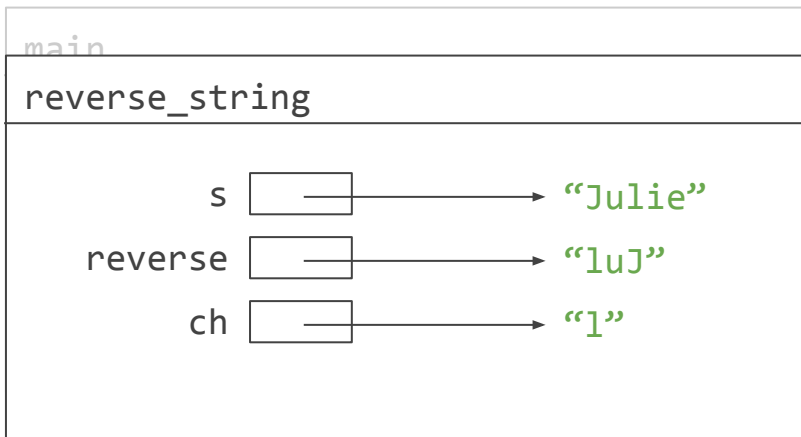
main

reverse\_string





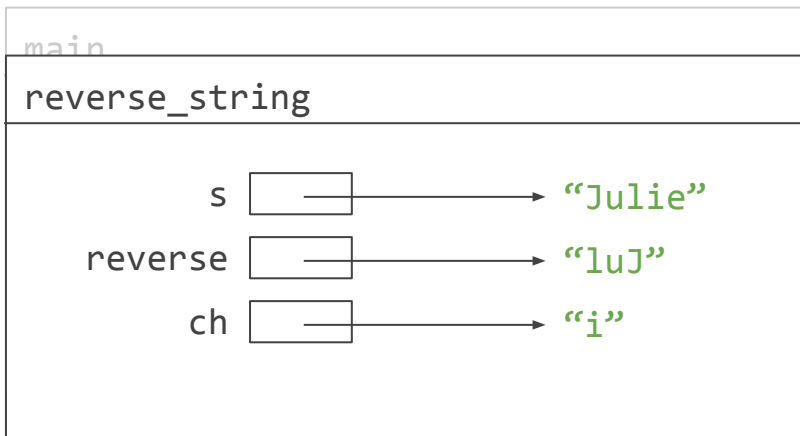
```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```





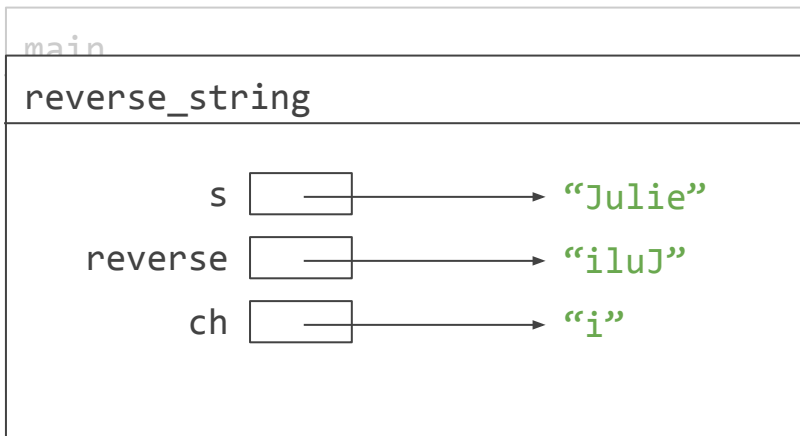


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



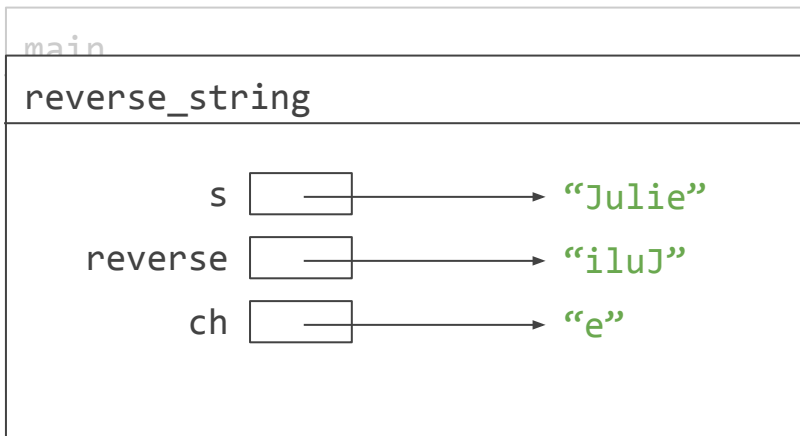


```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```





```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```

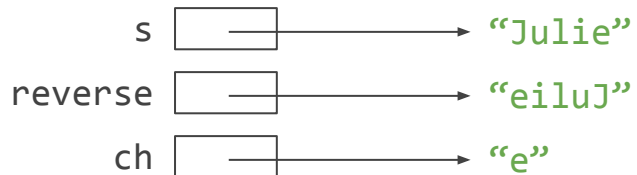




```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```

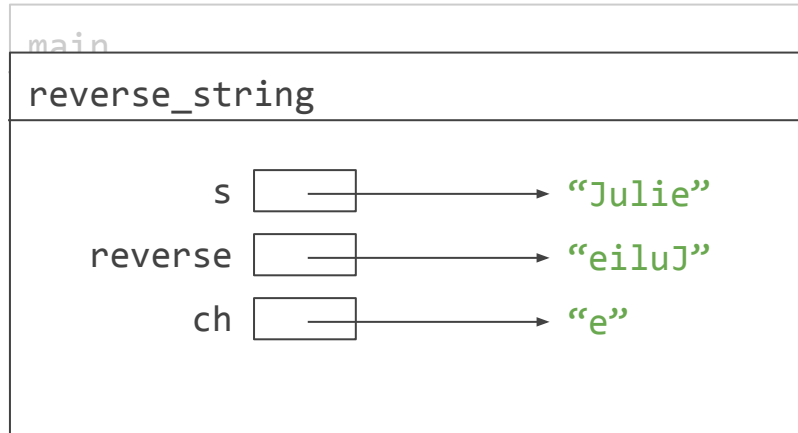
main

reverse\_string





```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```





```
def reverse_string(s):  
    reverse = ""  
    for ch in s:  
        reverse = ch + reverse  
    return reverse  
  
def main():  
    name = "Julie"  
    reverse = reverse_string(name)
```



main

name  → "Julie"

reverse  → "eiluJ"



# Kayak



# A man, a plan, a canal - Panama!





कडक

# Solving the Puzzle



How do we  
generate these  
numbers?

Valid numbers

971620000

542980001

.

.

.

433430128

.

.

.

284675959

155895960



971620000

542980001

.

.

.

433430128

.

.

.

284675959

155895960



97162 0000

54298 0001

.

.

.

43343 0128

.

.

.

28467 5959

15589 5960



Random      Increasing

97162      0000

54298      0001

•

•

•

43343      0128

•

•

•

28467      5959

15589      5960



Only the medicine  
manufacturer knows  
the random parts

Random	Increasing
97162	0000
54298	0001
	.
	.
	.
43343	0128
	.
	.
	.
28467	5959
15589	5960



```
N_LABELS = 5000
```

```
def main():  
    for i in range(N_LABELS):  
        rand_part = <5 digit string>  
        unique_part = <4 digit string>  
        id = rand_part + unique_part  
        print(id)
```

Random      Increasing

97162      0000

54298      0001

.

.

.

43343      0128

.

.

.

28467      5959

15589      5960





```
import random
```

```
N_LABELS = 5000
```

```
def main():
```

```
    for i in range(N_LABELS):
```

```
        rand_part = random.randint(0, 99999)
```

```
        unique_part = i
```

```
        id = rand_part + unique_part
```

```
        print(id)
```

Random      Increasing

97162          0000

54298          0001

.

.

.

43343          0128

.

.

.

28467          5959

15589          5960



```
import random
```

```
N_LABELS = 5000
```

```
def main():
```

```
    for i in range(N_LABELS):
```

```
        rand_part = random.randint(0, 99999)
```

```
        unique_part = i
```

```
        id = rand_part + unique_part
```

```
        print(id)
```

Random      Increasing

97162      0000

54298      0001

Not necessarily correct length

43343      0120

·  
·  
·

28467      5959

15589      5960



```
import random

N_LABELS = 5000

def main():
    for i in range(N_LABELS):
        rand_part = random.randint(0, 99999)
        unique_part = i
        id = rand_part + unique_part
        print(id)
```

Random	Increasing
97162	0000
54298	0001
.	.
.	.
28467	5959
15589	5960

Integer addition rather than string concatenation





	Random	Increasing
<code>N_LABELS = 5000</code>		
<code>def main():</code>	97162	0000
<code>for i in range(N_LABELS):</code>	54298	0001
<code>rand_part = pad(random.randint(0, 99999), 5)</code>		.
<code>unique_part = pad(i, 4)</code>		.
<code>id = rand_part + unique_part</code>		.
<code>print(id)</code>	4343	0128
<code>def pad(num, length):</code>		.
<code>num_string = str(num)</code>		.
<code>while len(num_string) &lt; length:</code>		.
<code>num_string = "0" + num_string</code>	28467	5959
<code>return num_string</code>	15589	5960