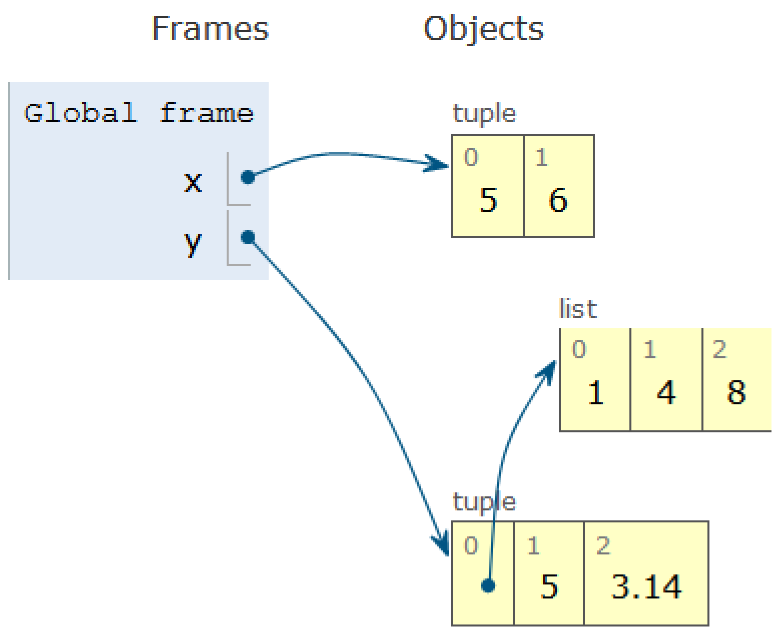
Compsci 101
DeMorgan's Law, Short circuiting, Global, Tuples



Susan Rodger

February 17, 2022

`L` is for …

- **Loops**
  - While, For, Nested – Iteration!
- **Library**
  - Where we find APIs and Implementations
- **Logic**
  - Boolean espressions in if statements, loops
- **Linux**
  - The OS that runs the world?

Keith Kirkland

- BS ME, BFA Accessories Design, MID Industrial and Product Design
- Co-founder of WearWorks
- Wayband – wearable haptic navigation device for blind
- Device guided blind marathon runner in NYC marathon

“We design products that shift people’s lives in a meaningful way”

“We take large challenges and turn them into opportunities that will one day help people and awaken the problems that can be solved. We believe in setting new standards for what is possible. ”

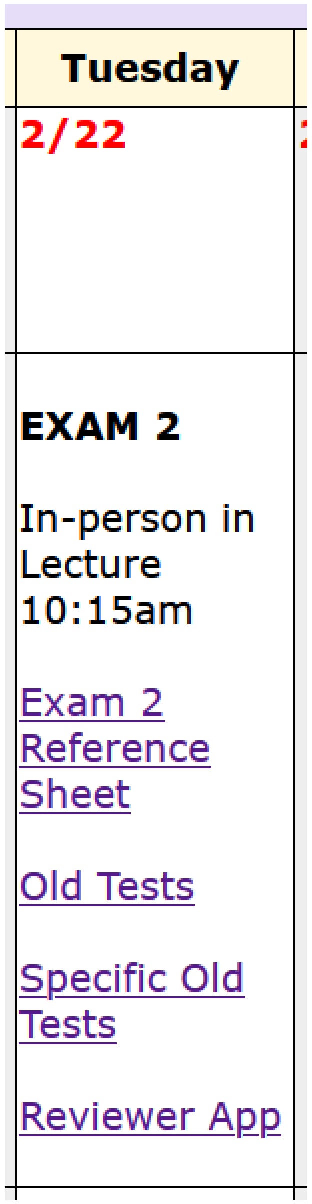
Announcements

- APT-3 due tonight
- Assign 3 due Tuesday, March 1
- Lab 6 on Friday, do prelab

- Exam 2 – Tuesday, Feb 22
  - in person during lecture time
- APT Quiz 1 – Feb 24-27

Exam 2 – in person – Tues, Feb 22

- **Exam is in class on paper – 10:15am**
  - Need pen or pencil
- **See study materials under 2/22 date**
  - Exam 2 Reference sheet - part of exam
- **Covers**
  - topics /reading through today
  - APTs through APT3
  - Labs through Lab 5, Lab 6 (Part 1 – list comprehensions)
  - Assignments through Assignment 2
  - Concepts from Assign2, No turtles



Tuesday

2/22

EXAM 2

In-person in Lecture 10:15am

Exam 2 Reference Sheet

Old Tests

Specific Old Tests

Reviewer App

Exam 2 – How to Study

- Practice writing code on paper!
- Rewrite an APT
- Try to write code from lecture from scratch
- Try to write code from lab from scratch
- Practice from old exams
- Put up old Sakai quizzes, but better to practice writing code
- Look at Exam 2 reference sheet when writing code!

Exam 2

- Exam 2 is your own work!
- No looking at other people's exam
- You cannot use any notes, books, computing devices, calculators, or any extra paper
- Bring only a pen or pencil
- The exam has extra white space and has the Exam 2 reference sheet as part of the exam.

- Do not discuss any problems on the exam with others until it is handed back

APT Quiz 1 Feb 24-27

- Opens 2/24 11:30am
- Closes at 11pm 2/27 – must finish all by this time
- There are two parts based on APTs 1-3
  - Each part has two APT problems
  - Each part is 1.5 hours – more if you get accommodations
  - Each part starts in Sakai under tests and quizzes
  - Sakai is a starting point with countdown timer that sends you to a new apt page just for each part
  - Could do each part on different day or same days
- Will put up problems today from an old APT Quiz so you can practice (not for credit) – on APT Page

APT Quiz 1

- Is your own work!
  - No collaboration with others!
  - Use your notes, lecture notes, your code, textbook
  - DO NOT search for answers!
  - Do not talk to others about the quiz until grades are posted
- Post private questions on Ed Discussion
  - We are not on between 10pm and 8am!
  - We are not on all the time
  - Will try to answer questions between 8am – 10pm
- See 101 APT page for tips on debugging APTs

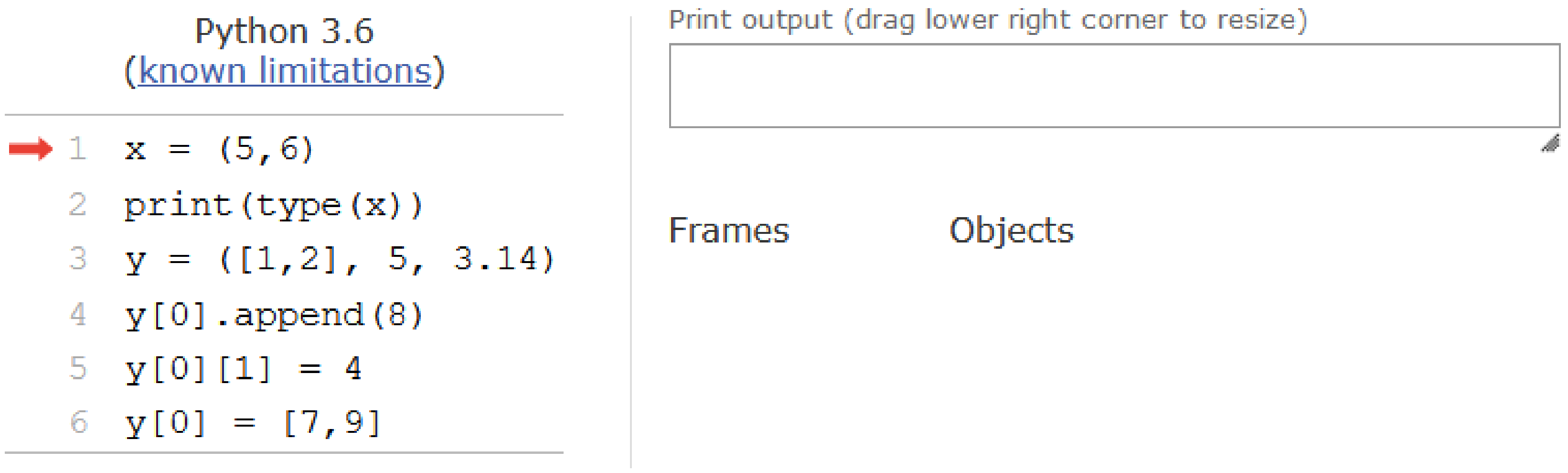
PFTD

- Tuples
- Global
- DeMorgan’s Law
- Short Circuiting
- APT

Tuple: What and Why?

- Similar to a list in indexing starting at 0
  - Can store any type of element
  - Can iterate over
- Immutable - Cannot mutate/change its value(s)
  - Efficient because it can't be altered
- Examples:
  - **`x = (5,6)`**
  - **`y = ([1,2],3.14)`**

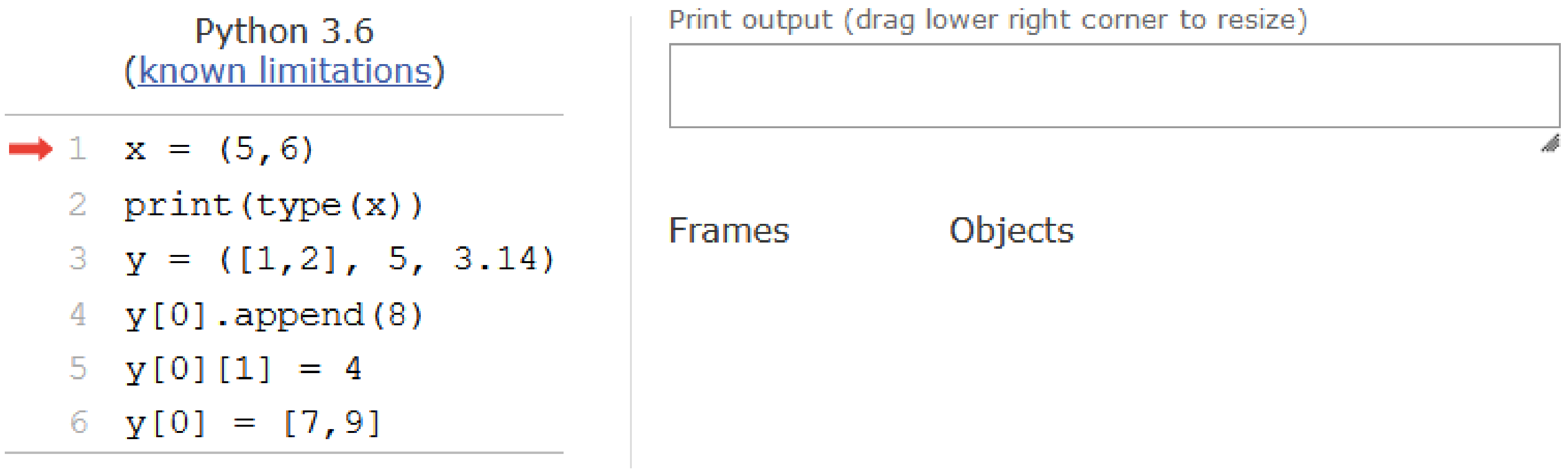
Tuple Trace in Python Tutor



Python 3.6
(known limitations)

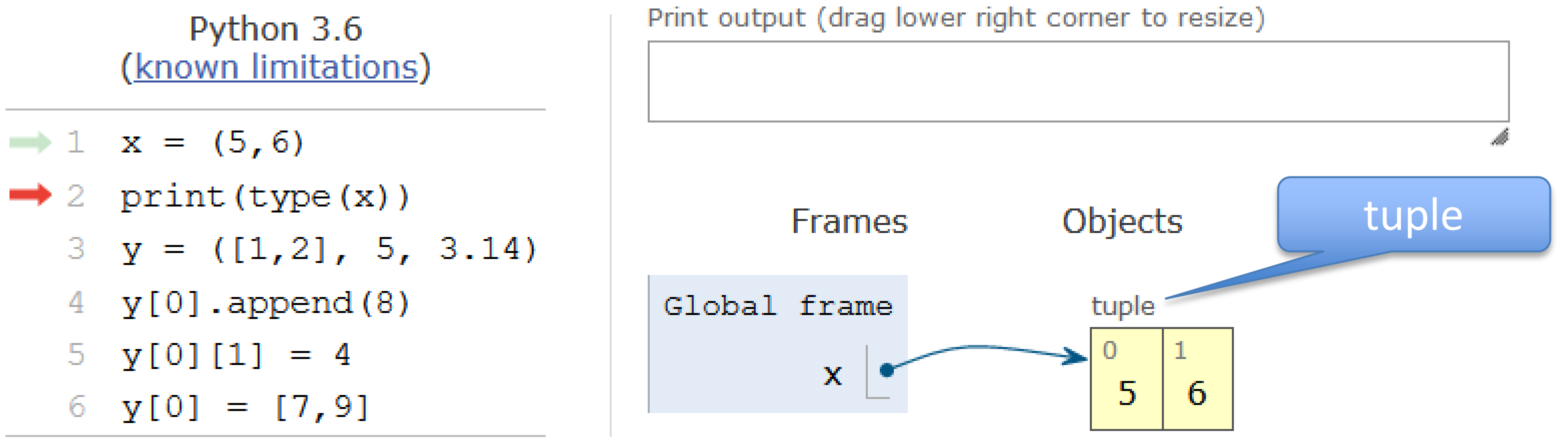
```
x = (5,6)
print(type(x))
y = ([1,2], 5, 3.14)
y[0].append(8)
y[0][1] = 4
y[0] = [7,9]
```

Tuple Trace in Python Tutor



Python 3.6
(known limitations)

```
x = (5,6)
print(type(x))
y = ([1,2], 5, 3.14)
y[0].append(8)
y[0][1] = 4
y[0] = [7,9]
```



Python 3.6
(known limitations)

```
x = (5,6)
print(type(x))
y = ([1,2], 5, 3.14)
y[0].append(8)
y[0][1] = 4
y[0] = [7,9]
```

Variables and their Scope

- Local variable – variable in function only known in that function
- Parameter – way to pass information to a function
- Global variable - variable known throughout the whole file

When to use Global Variables

- Typically, don't use global variables
  - Harder to share a function if it refers to a global variable
  - Act differently than other variables
- Sometimes makes sense
  - Global variable is used in most functions
  - Saves passing it to every function
- Best practice = help other humans read the code
  - Global variables define at top of file
  - When global used in function, declared as global at beginning of function

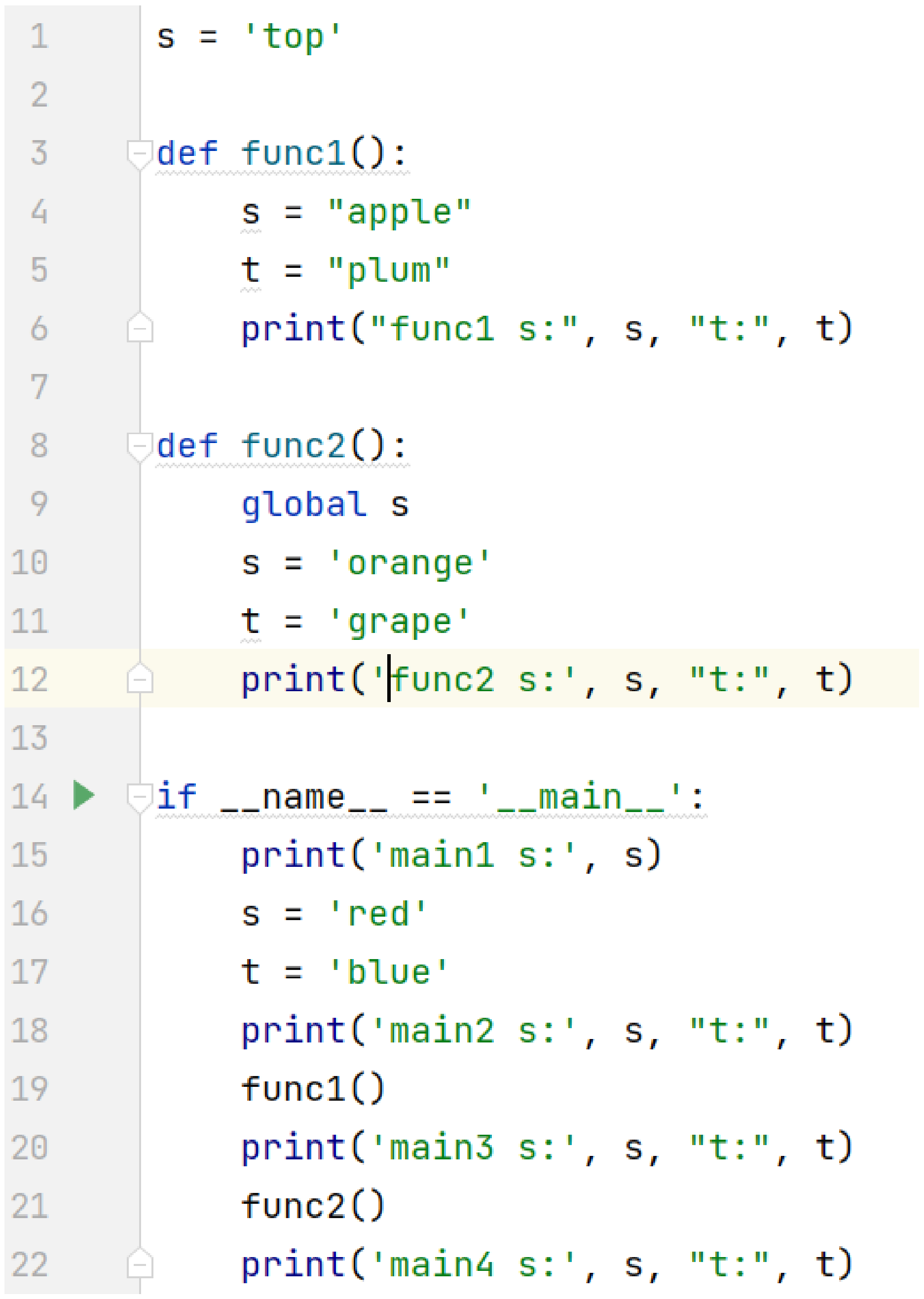
Summary - What is global?

- Accessible everywhere in the file (or "module")
- Variable is in the global frame
  - First frame in Python Tutor
- If declared global in a function:
  - The variable in the global frame can also be reassigned in that function
  - Despite Python being in a different frame!
- Eliminates the need to pass this value to all the functions that need it

When reading code with globals

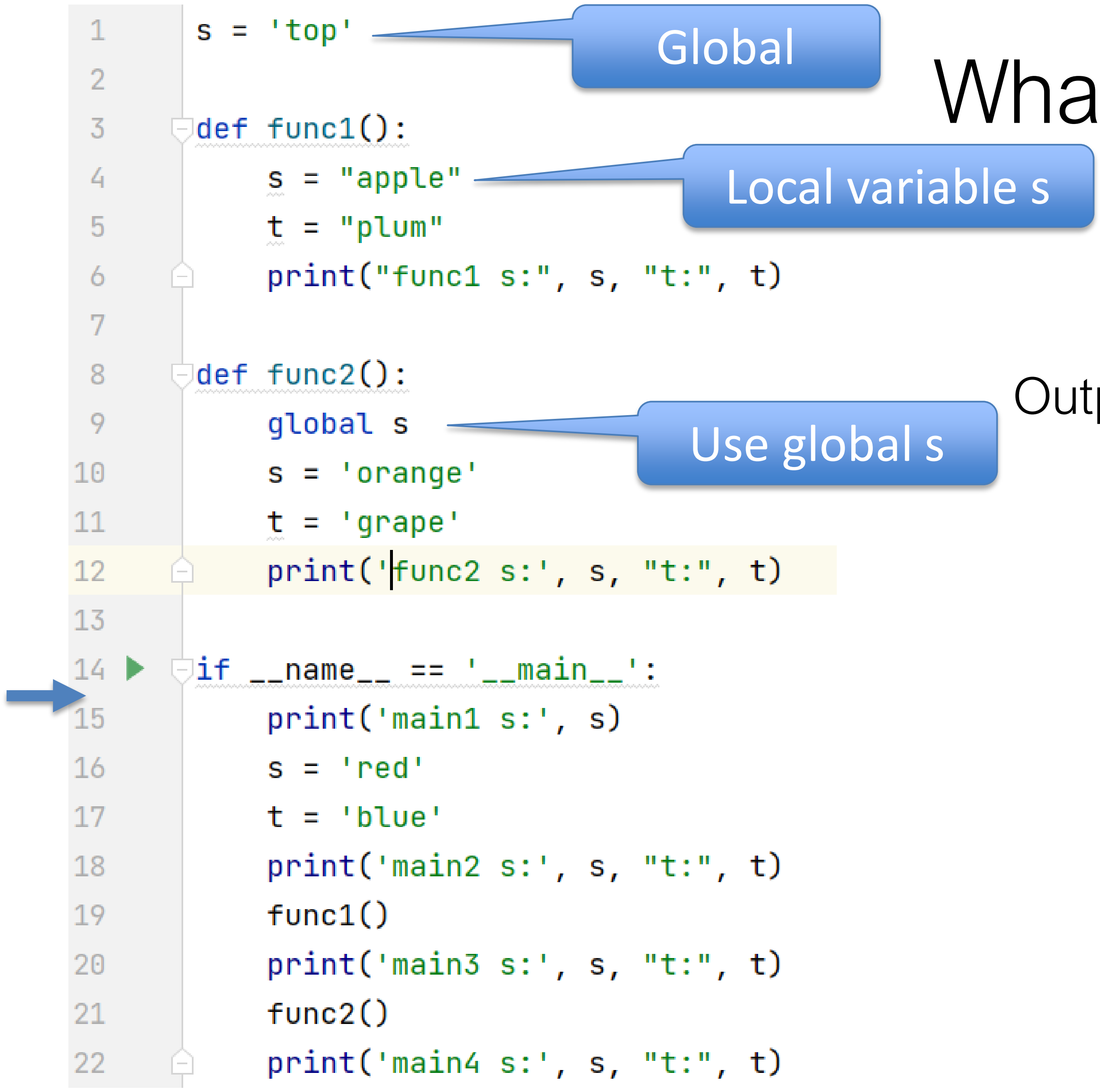
- **When checking the value of a variable, ask:**
  - Is this variable local to the function or in the global frame?
- **When in a function and assigning a value to a variable, ask:**
  - Has this variable been declared global?
    - If yes, reassign the variable in the **global frame**
    - If no, create/reassign the variable in the **function's local frame**

What will print?



```python
s = 'top'

def func1():
    s = "apple"
    t = "plum"
    print("func1 s:", s, "t:", t)

def func2():
    global s
    s = 'orange'
    t = 'grape'
    print('func2 s:', s, "t:", t)

if __name__ == '__main__':
    print('main1 s:', s)
    s = 'red'
    t = 'blue'
    print('main2 s:', s, "t:", t)
    func1()
    print('main3 s:', s, "t:", t)
    func2()
    print('main4 s:', s, "t:", t)
```

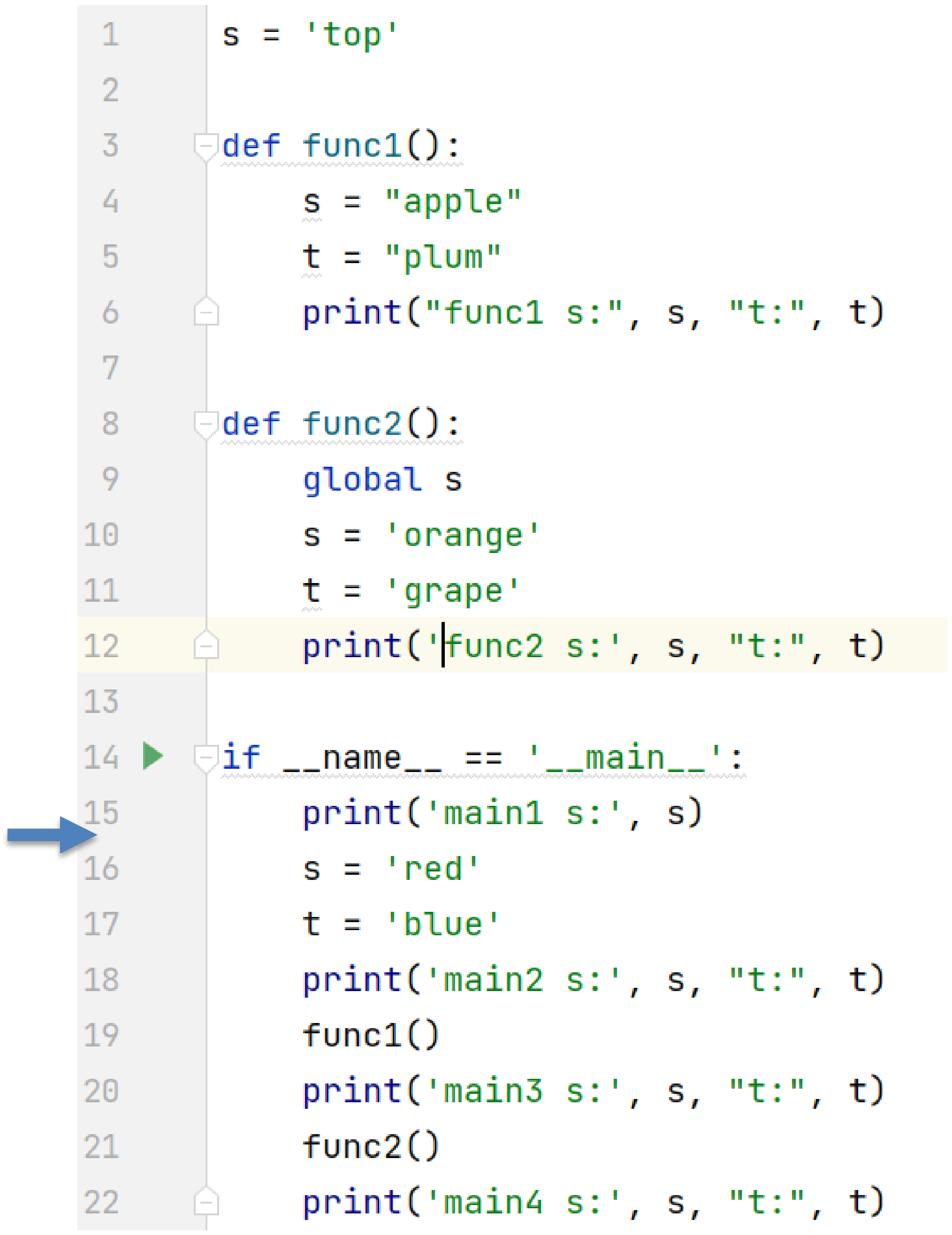
What will print?



```python
s = 'top'

def func1():
    s = "apple"
    t = "plum"
    print("func1 s:", s, "t:", t)

def func2():
    global s
    s = 'orange'
    t = 'grape'
    print('func2 s:', s, "t:", t)

if __name__ == '__main__':
    print('main1 s:', s)
    s = 'red'
    t = 'blue'
    print('main2 s:', s, "t:", t)
    func1()
    print('main3 s:', s, "t:", t)
    func2()
    print('main4 s:', s, "t:", t)
```

Output:

What will print?



```python
s = 'top'

def func1():
    s = "apple"
    t = "plum"
    print("func1 s:", s, "t:", t)

def func2():
    global s
    s = 'orange'
    t = 'grape'
    print('func2 s:', s, "t:", t)

if __name__ == '__main__':
    print('main1 s:', s)
    s = 'red'
    t = 'blue'
    print('main2 s:', s, "t:", t)
    func1()
    print('main3 s:', s, "t:", t)
    func2()
    print('main4 s:', s, "t:", t)
```

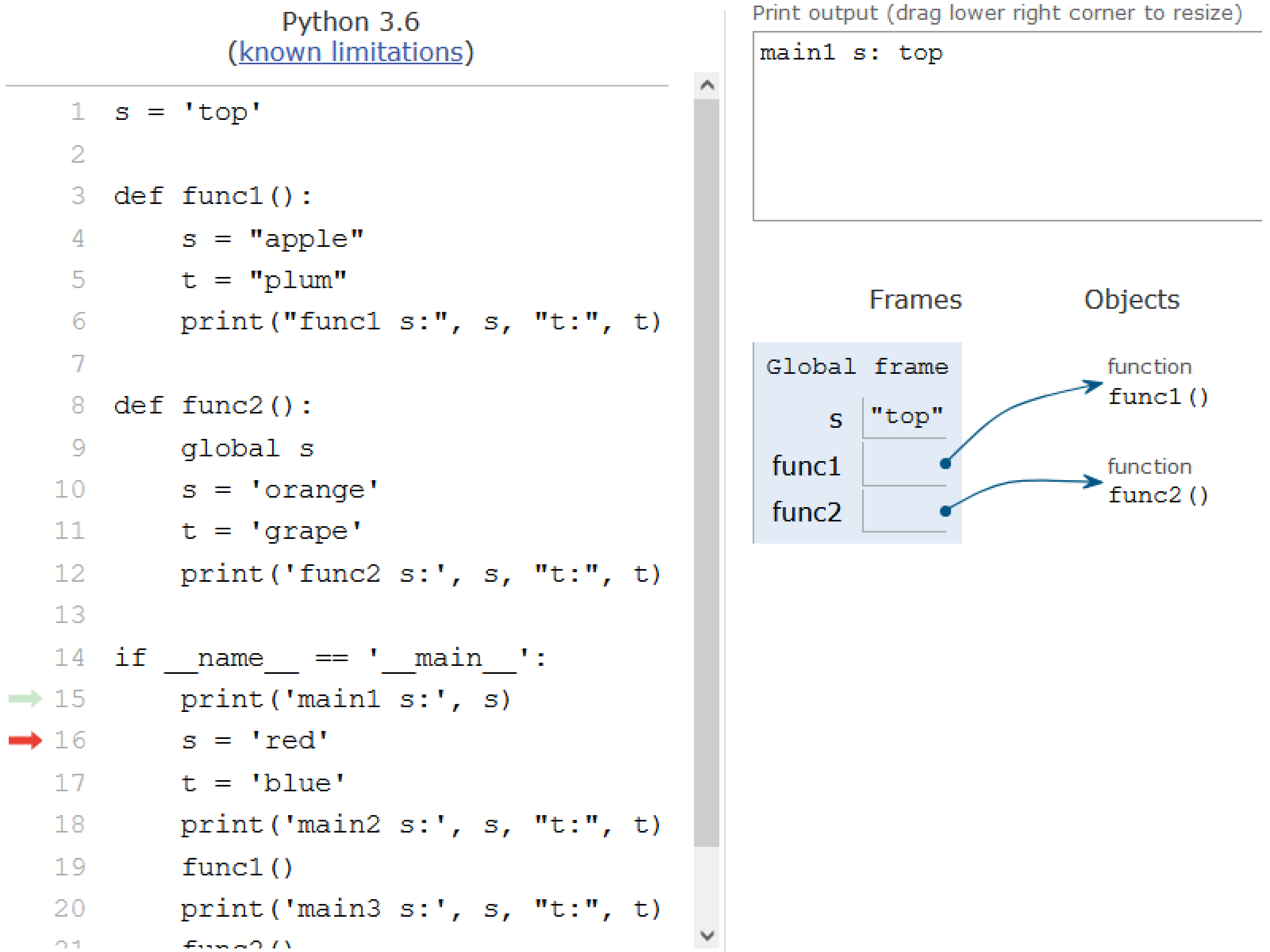
Output:

```
main1 s: top
```

Now let’s see the same thing in Python Tutor

- Global variables are in the global frame

Python Tutor – Step 6



Python 3.6
(known limitations)

```
s = 'top'

def func1():
    s = "apple"
    t = "plum"
    print("func1 s:", s, "t:", t)

def func2():
    global s
    s = 'orange'
    t = 'grape'
    print('func2 s:', s, "t:", t)

if __name__ == '__main__':
    print('main1 s:', s)
    s = 'red'
    t = 'blue'
    print('main2 s:', s, "t:", t)
    func1()
    print('main3 s:', s, "t:", t)
    func2()
```

Print output (drag lower right corner to resize)

```
main1 s: top
```

Variables
What, where, read, write? (in 101)

What is it?	Where first created?	Where accessible? (read)	Where reassign-able? (write)
Regular variable in main	In main	In main only (technically anywhere, but don't do that)	In main only
Regular local function variable	In function	In function only	In function only
Global variable	Top of file	If not reassigning the value, in main and all functions	In main or in any function that first declares it global

Assignment 3 Transform

- Uses several global variables.

- Only use global variables when we specify in an assignment

WOTO-1 – Globals http://bit.ly/101s22-0217-1

List .index vs String .find

```
str = "computer“                    Values:
pos = str.find("m")
pos = str.find("b")

lst = ["a", "b", "c", "a"]
indx = lst.index("b")
indx = lst.index("B")
```

Let’s Write list Index function

- Call in findIndex(lst, item)
- Write it so it works like the string find function
  - **lst** is a list
  - **elm** is an element
  - Return the position of **elm** in **lst**
  - Return **-1** if **elm** not in **lst**
  - Use while loop to implement
- What is the while loop’s Boolean condition?

```
index = 0
while BOOL_CONDITION:
   index += 1
```

While Boolean condition

```
index = 0
while BOOL_CONDITION:
    index += 1
```

- What is the while loop’s Boolean condition?

DeMorgan's Law

- While loop stopping conditions, stop with either:
  - `lst[index] == elm`
  - `index >= len(lst)`
- While loop needs negation: DeMorgan's Laws

  `not (A and B)` equivalent to **`(not A) or (not B)`**

  `not (A or B)` equivalent to **`(not A) and (not B)`**

Fill in the blanks

Think: DeMorgan's Law

A	B	not (A and B)	(not A) or (not B)
True	True		False
True	False	True	
False	True		True
False	False	True	

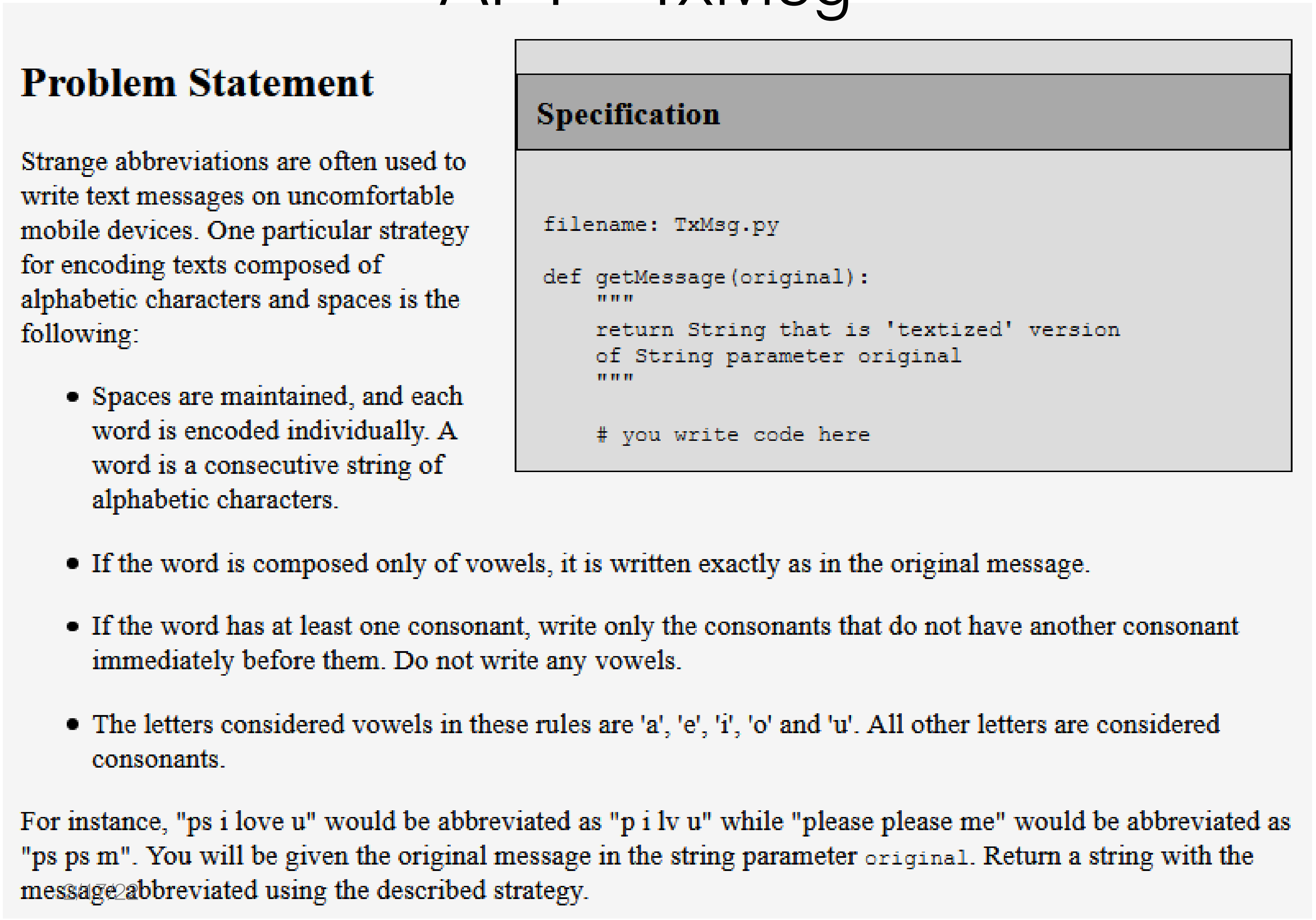
A	B	not (A or B)	(not A) and (not B)
True	True	False	
True	False		False
False	True	False	
False	False		True

WOTO-2: Will this work?
http://bit.ly/101s22-0217-2

Next look at future APT

- Understanding this APT will help you understand Assignment 3 Transform

APT - TxMsg



Problem Statement

Strange abbreviations are often used to write text messages on uncomfortable mobile devices. One particular strategy for encoding texts composed of alphabetic characters and spaces is the following:

- Spaces are maintained, and each word is encoded individually. A word is a consecutive string of alphabetic characters.
- If the word is composed only of vowels, it is written exactly as in the original message.
- If the word has at least one consonant, write only the consonants that do not have another consonant immediately before them. Do not write any vowels.
- The letters considered vowels in these rules are 'a', 'e', 'i', 'o' and 'u'. All other letters are considered consonants.

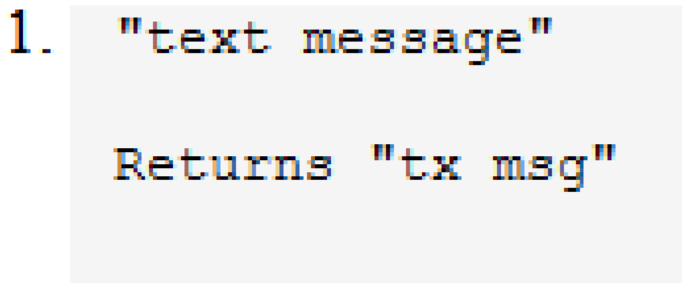
For instance, "ps i love u" would be abbreviated as "p i lv u" while "please please me" would be abbreviated as "ps ps m". You will be given the original message in the string parameter `original`. Return a string with the message abbreviated using the described strategy.

Specification

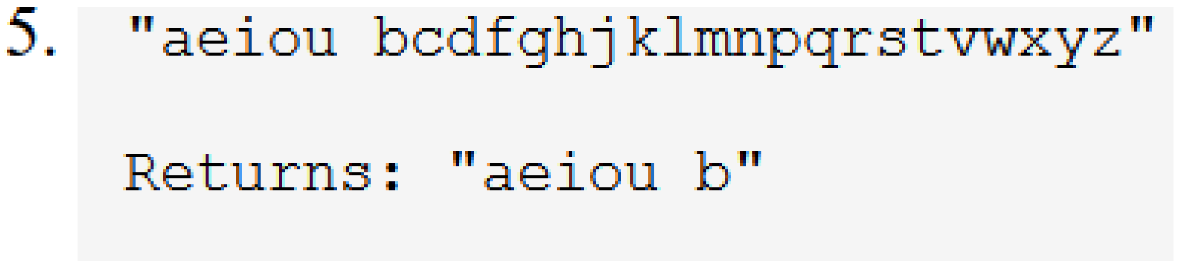
```
filename: TxMsg.py

def getMessage(original):
    """
    return String that is 'textized' version
    of String parameter original
    """

    # you write code here
```

Examples

Examples



1.
```
"text message"

Returns "tx msg"
```



5.
```
"aeiou bcdfghjklmnpqrstvwxyz"

Returns: "aeiou b"
```

WOTO-3 – TxMsg
http://bit.ly/101s22-0217-3

Write helper function *transform*

- How?
- Use seven steps
- Work an example by hand