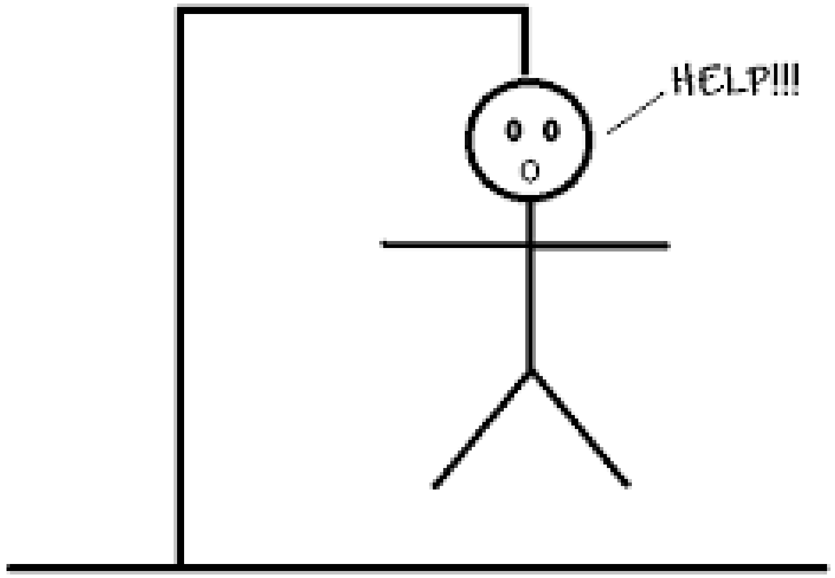
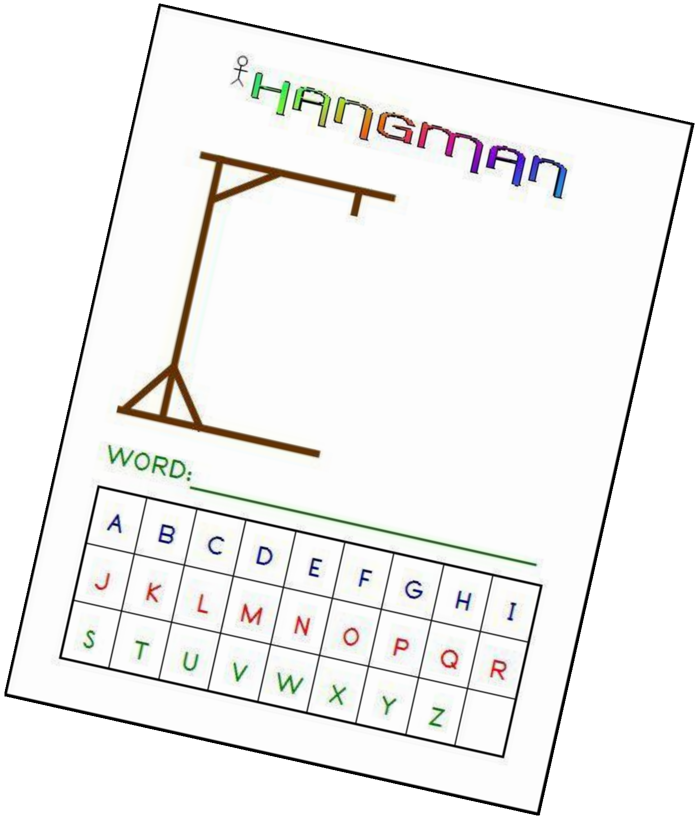
Hangman



in python

Play the game



List all the steps you took !

What steps did you find ?

Set lives to 9

Check if the user has run out of lives

Choose a word

Check if users guess has already been used

Ask user to guess letter

Check if users guess is in word

Make guess --” same length as word

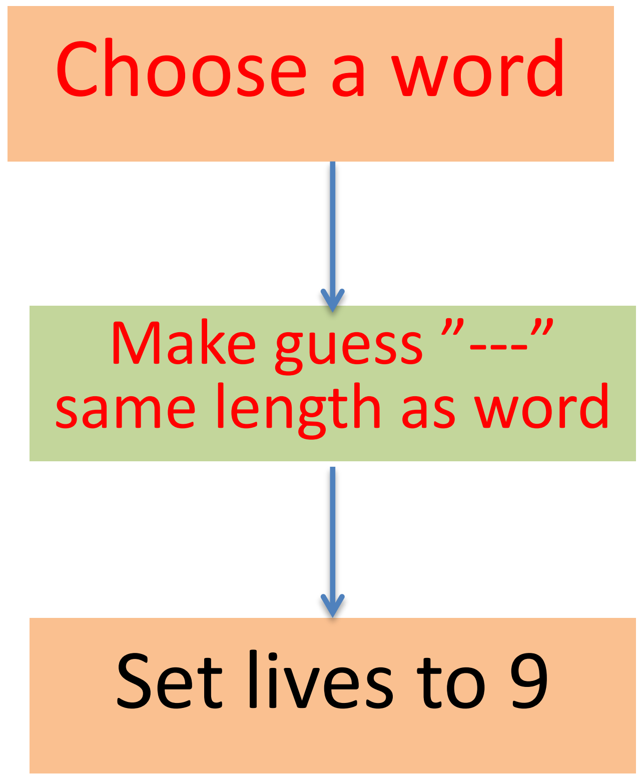
Draw hangman

Place guessed letter in guess if found

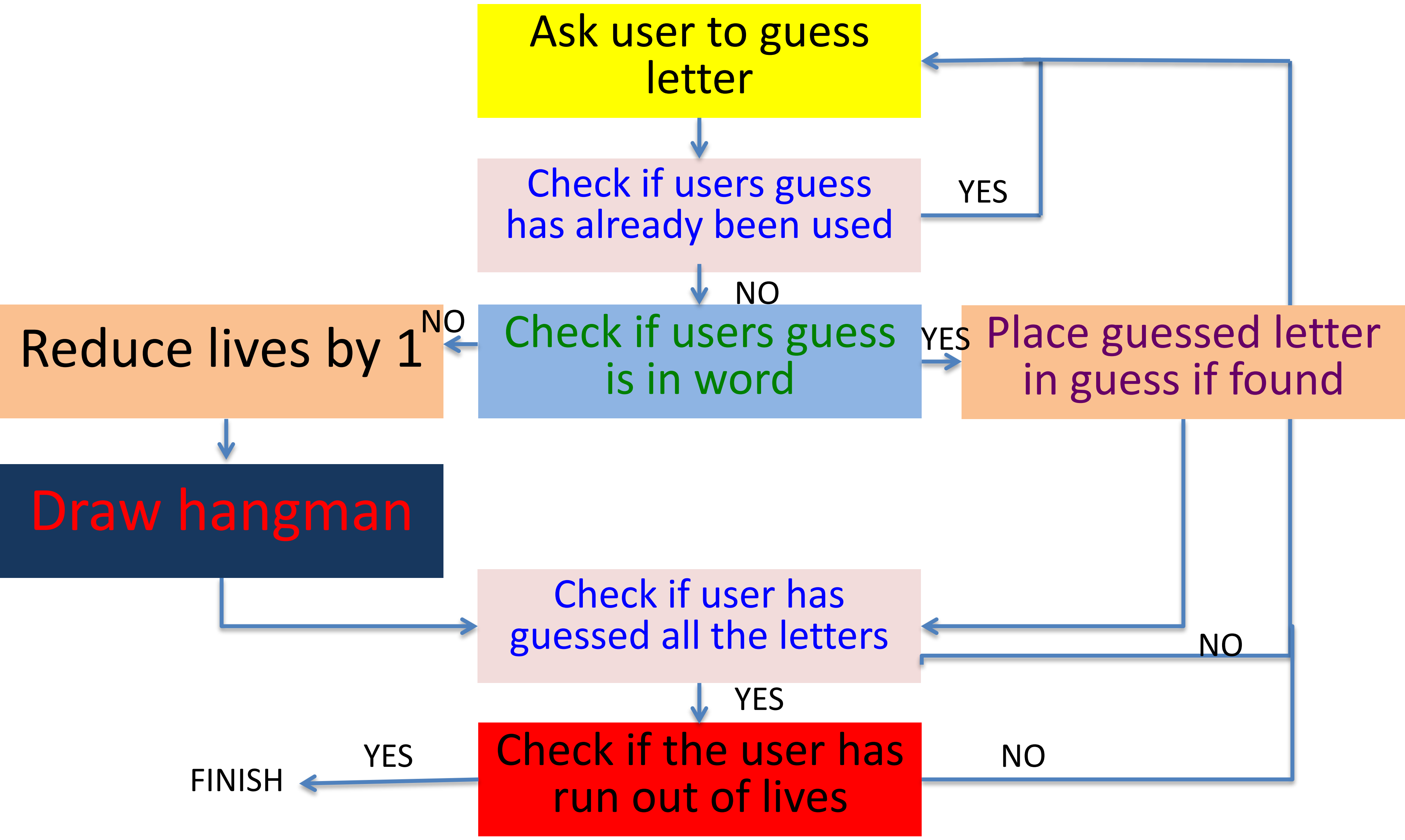
Check if user has guessed all the letters

Reduce lives by 1

What steps did you find ?



What steps did you find ?


Ask user to guess letter
Check if users guess has already been used
YES
NO
Check if users guess is in word
NO
Reduce lives by 1
YES
Place guessed letter in guess if found
Draw hangman
Check if user has guessed all the letters
NO
YES
Check if the user has run out of lives
YES
FINISH
NO

The Data for hangman

- What data are we going to need to process
- How are we going to represent it in Python ?

Lives

Word

Current guess

Strings

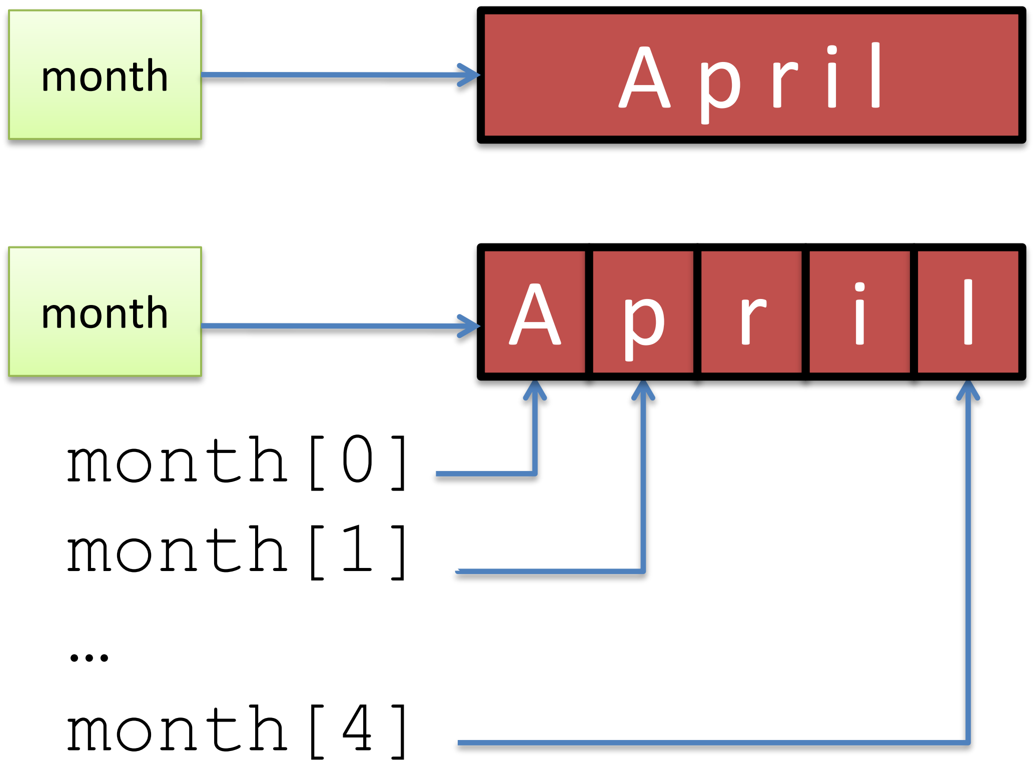
How to slice and dice them !

In

python™

Creating and indexing a string

```
month = "April"
```



Each letter is stored in memory as its ASCII value

We can access each letter through its index (position)

Give it a go !

Can you ? Print each letter from a word one at a time using its position ?

```
word = "Superb"
word_length = len(word)
for index in range (0,word_length):
    print("Index",index,"Letter",word[index])
```

- We say we are iterating through the word.
- This method is useful when we need to know the position of each letter

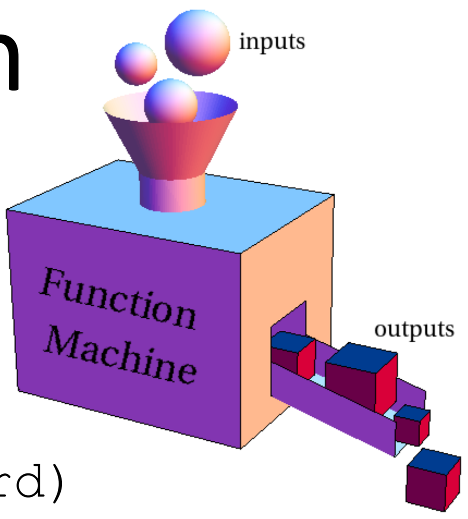
Hangman Part 1

```
r e a l i t y   ? a  => 2
```

Following the design phase for a hangman game, we need a function to :-

- *Look for a letter in a word and return the position of the letter.*

The first Function



```
def main():
    letter = "s"
    word = "television"
    location= find_letter_in(letter,word)
    print ("Found",letter,"at position",location)

def find_letter_in(letter,word):

    print("I am looking for ",letter,"in",word)
##      put your code in here
    return ??? What do we need to return ????

main()
```

The first Function - solution

```
def main():
    …..

def find_letter_in(letter,word):
    print("I am looking for ",letter,"in",word)
    word_length = len(word)
    for index in range (0,word_length):
        if letter == word[index]:
            print("Found at pos:",index)
            location = index
    return location

main()
```

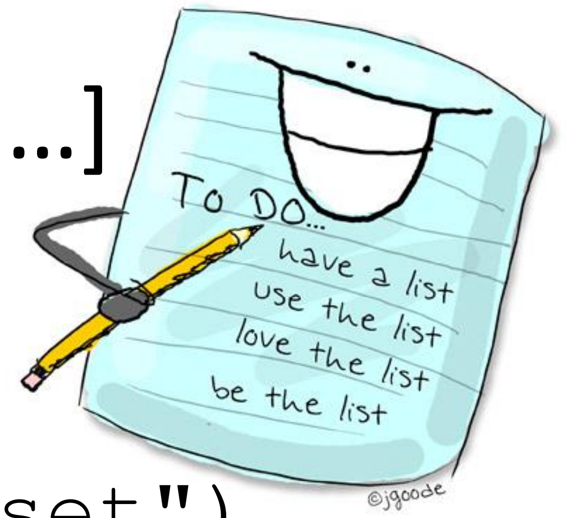
Test your function !!!

Does it work ?

- Try different letters and words
  - Try words and a letter which is not in the word
  - Try words where the letter occurs more than once

What other possibilities does our function need to be able to return ?

We need lists [item1,item2,…]



`find_letter_in("p", "sunset")`
- `should return []`

`find_letter_in("n", "newspaper")`
- `should return[0]`

`find_letter_in("e", "television")`
- `should return [1,3]`

but we will leave this for later

Hangman – what next ?

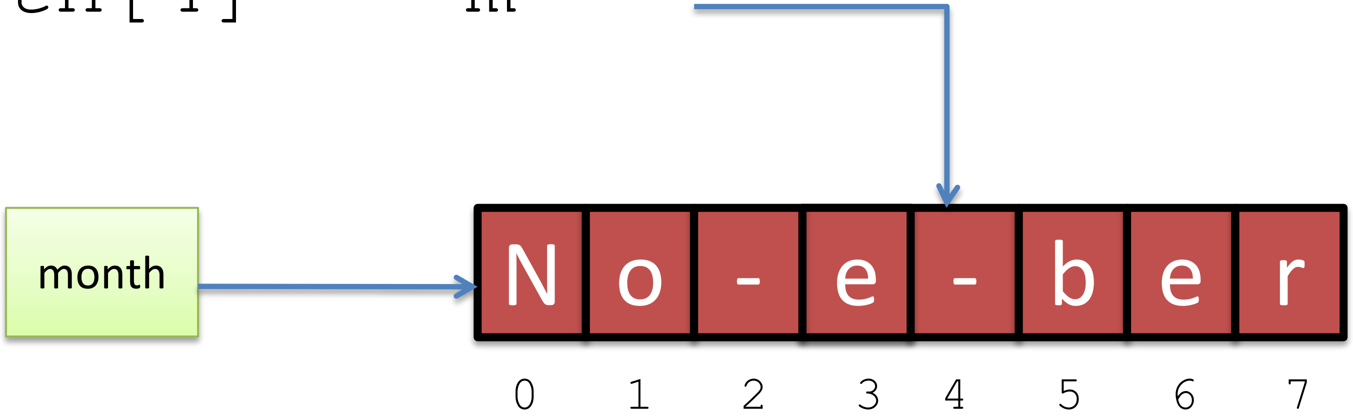
- Having guessed a letter correctly we need to we need to put it in the guessed word at the correct location.

```
 Found 'e' at position 3.


current_guess =  "- - - - - "
current_guess = "- - - e – "
```

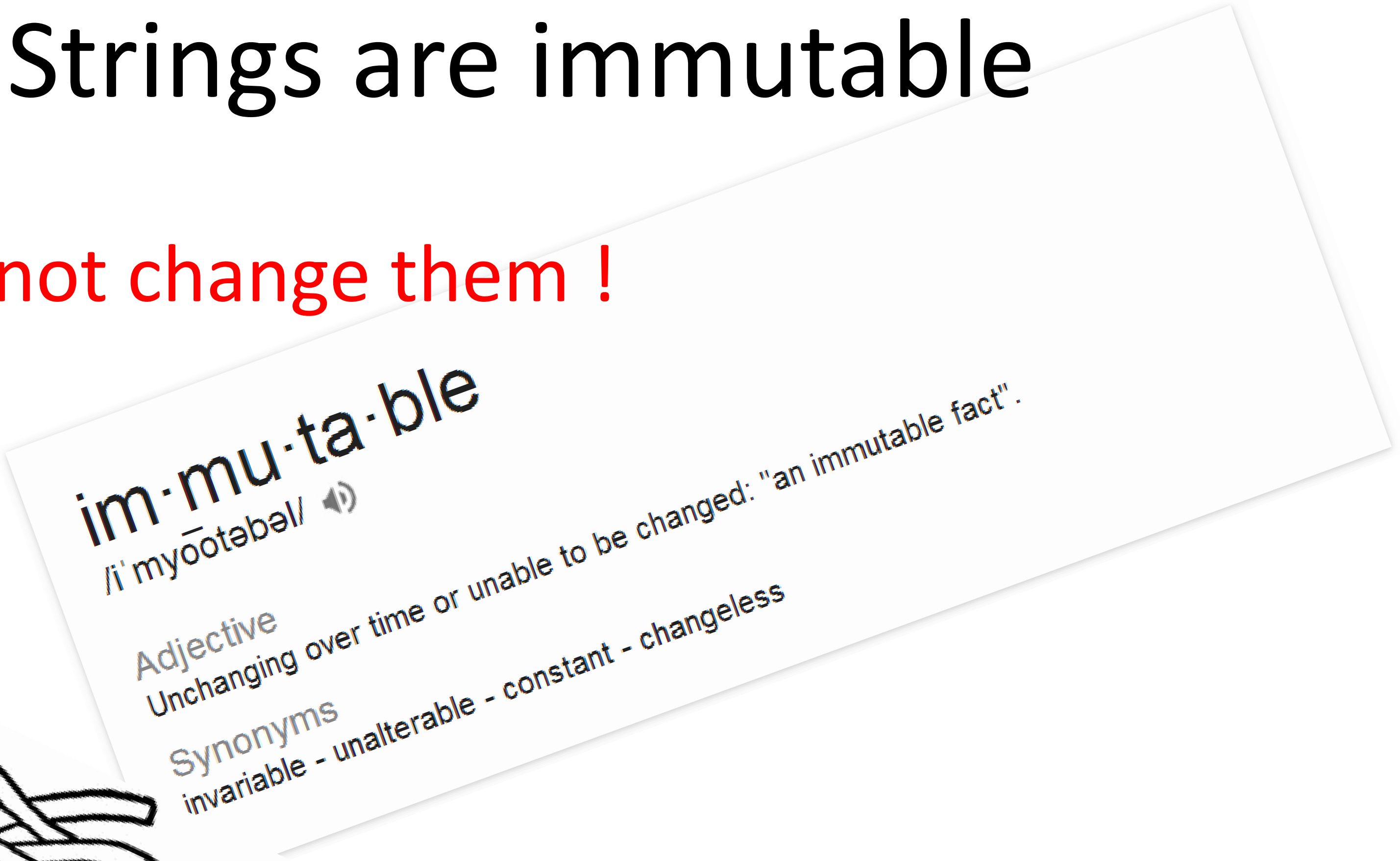
Can you changing a letter in a string ?

```
month= "No-e-ber"
month[4] = "m"
```



Strings are immutable

– we cannot change them !



im·mu·ta·ble

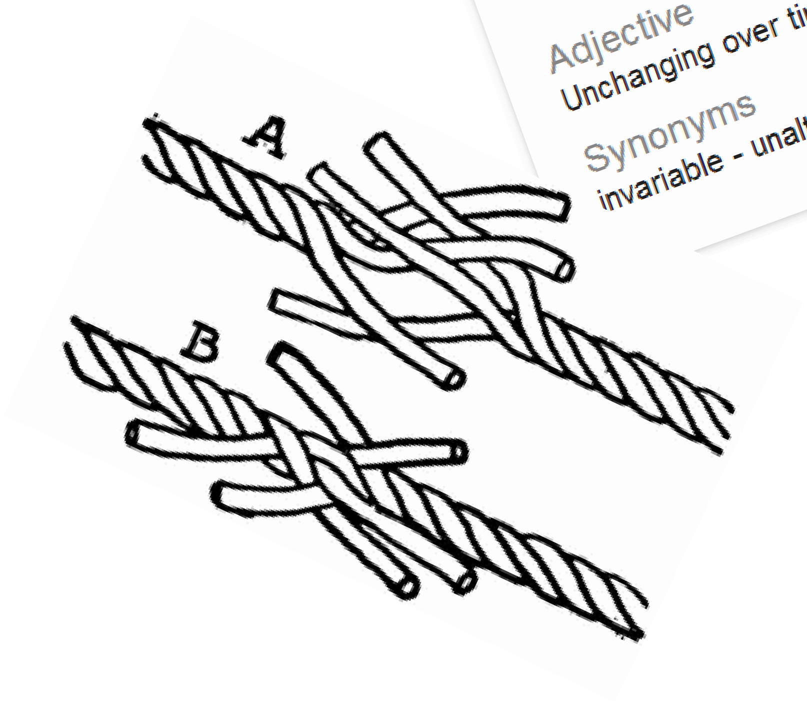
/iˈmyo͞otəbəl/

Adjective

Unchanging over time or unable to be changed: "an immutable fact".

Synonyms

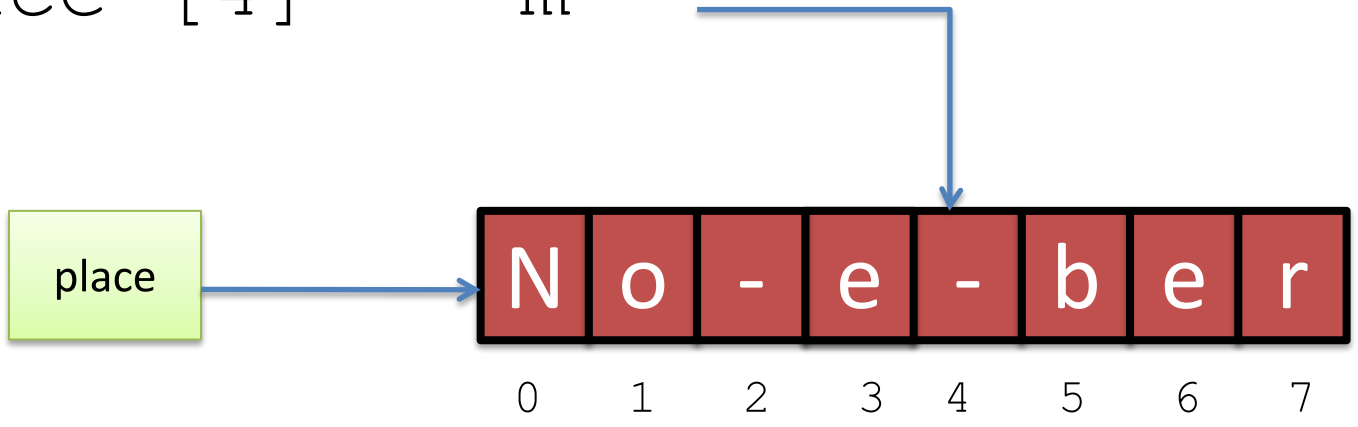
invariable - unalterable - constant - changeless



But we can slice and splice them !

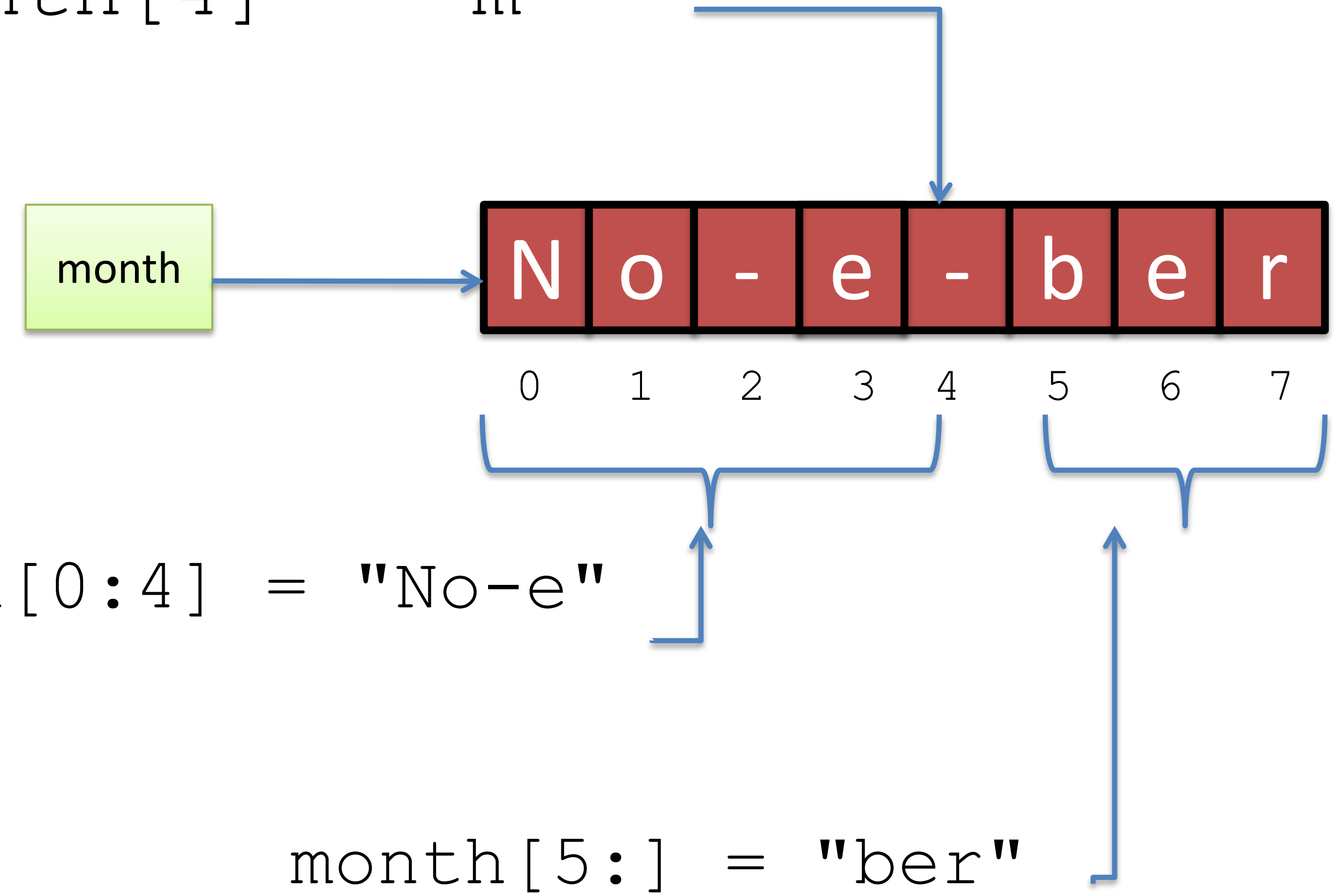
Can you changing a letter in a string ?

```
place = "No-e-ber"
place [4] = "m"
```



Can you changing a letter in a string ?

```
month= "No-e-ber"
month[4] = "m"
```

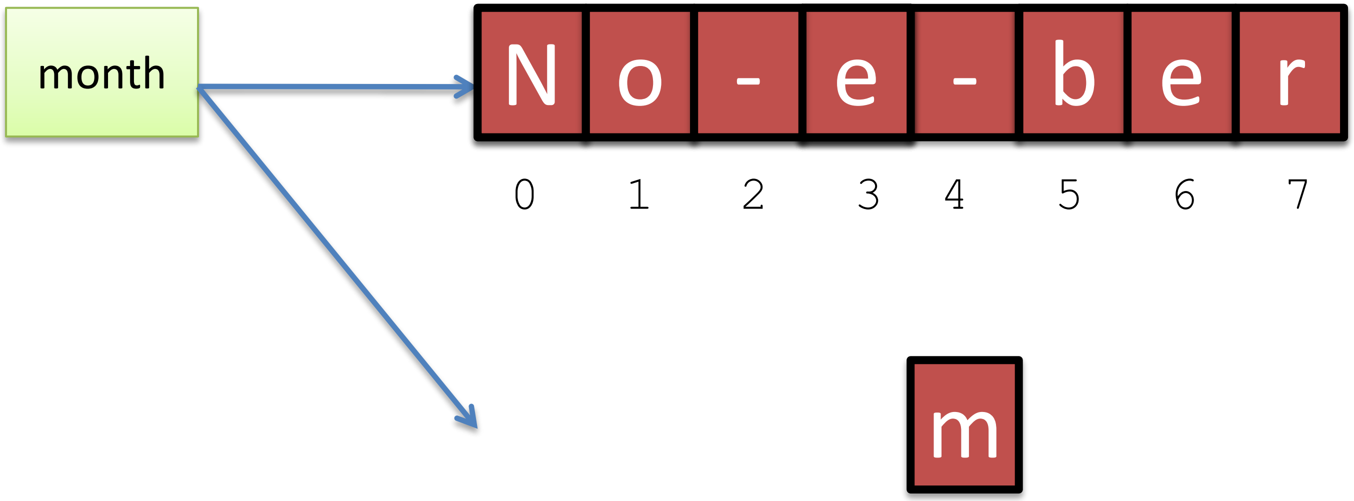


```
month[0:4] = "No-e"
```

```
month[5:] = "ber"
```

Changing a letter in a string

Change the letter at position 4 to a “m”



```
month=    month[0:4] + "m"  + month[5:]
```

Hangman Part 2

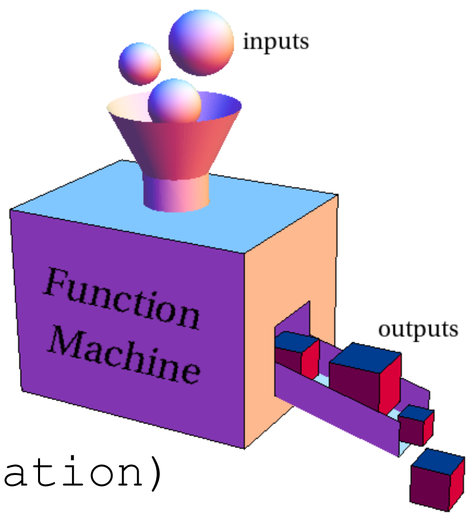
a , 2 , "_ _ _ _ _ _ _ _ "

=> "_ _ a _ _ _ _"

Following the design phase for a hangman game, we need a function to :-

- *Given a letter, a position and a word; replace the character in the word at the given position with the letter.*

Second function

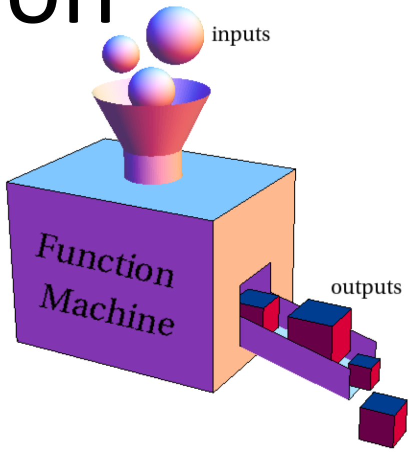


```
def main():
     letter = "s"
     word = "television"
     current_guess = "----------"
     location= find_letter_in(letter,word)
     print ("Found",letter,"at position",location)
     current_guess = add_found_to_guess …
                        /… (current_guess,location,letter)
    print("Current_guess is",current_guess)

def add_found_to_guess(current_guess,location,letter):
    ###  Your code in here !!!
    return ???? What should you return >
```

!!! … / … indicated the code is all on the same line !!!!!!!!!

Second function solution



```
def main():
    …..

def add_found_to_guess …
          /…(current_guess,location,letter):
   current_guess = current_guess[0:location]…
          /…+ letter+ current_guess[location+1:]
    return current_guess
```

!!! … / … indicated the code is all on the same line !!!!!!!!!

Test your function !!!

Does it work ?

- Try different letters at different positions.

What if we had found the letter at more than one position ?

We need lists !

Lists and how to uses them ! In python™

Lists

- Python, like most other languages has a data type for storing collections of things
- Often called Arrays
- Imagine a list for a lunch_menu

Creating a list

- We can create a list of any types of data
- List are enclosed by []
- Separate items in a list are separated by commas

```
lunch_menu = ["Burger", "Salad",
"Jacket Potato", "Pizza"]
```

Printing the whole list !

```
lunch_menu = ["Burger", "Salad",
"Jacket Potato", "Pizza"]

print(lunch_menu)
```

Printing an item from the list!

```
lunch_menu = ["Burger","Salad","Pizza"]
                 0          1        2

print(lunch_menu[2])
```

Iterating through a list with an index

- In a similar way we iterate through the letters in a string, we can iterate through the items in a list.

```
lunch_menu = ["Burger”,"Salad","Pizza"]
for item in lunch_menu:
    print("Item:",item)
```

NOTE
This way has not generated an index value, but has just pulled out the items from the list one at a time.
We Could have done it the same ways that we did with letters in a string, but here we were not interested in the position in the list

Appending an item to a list 1 of 2.

- To add an item to a list we use the append method.

```
letters = ["s","t"]

print(letters)
new_char = input("Enter a letter ")
letters.append(new_char)
print(letters)
```

Appending an item to a list 2 of 2.

```
letters.append(new_char)
```

- NOTE – We have changed the letters list, not created a new one.
- LISTS are mutable !!! We don’t code :-

```
letters = letters.append(new_char)
```

Hangman Part 3

Looking at our code so far …

What happens if we search for an ‘e’ in television ?

We need to rethink our code design !

Revisit our two functions.

Hangman Part 1

```
t e l e v i s i o n      ? e
                     => [1,3]
```

Following the design phase for a hangman game, we need a function to :-

- *Look for a letter in a word and return the position (or positions) of the letter as a list.*
- *A null list [] will indicate the letter was not found.*

Hangman Part 2

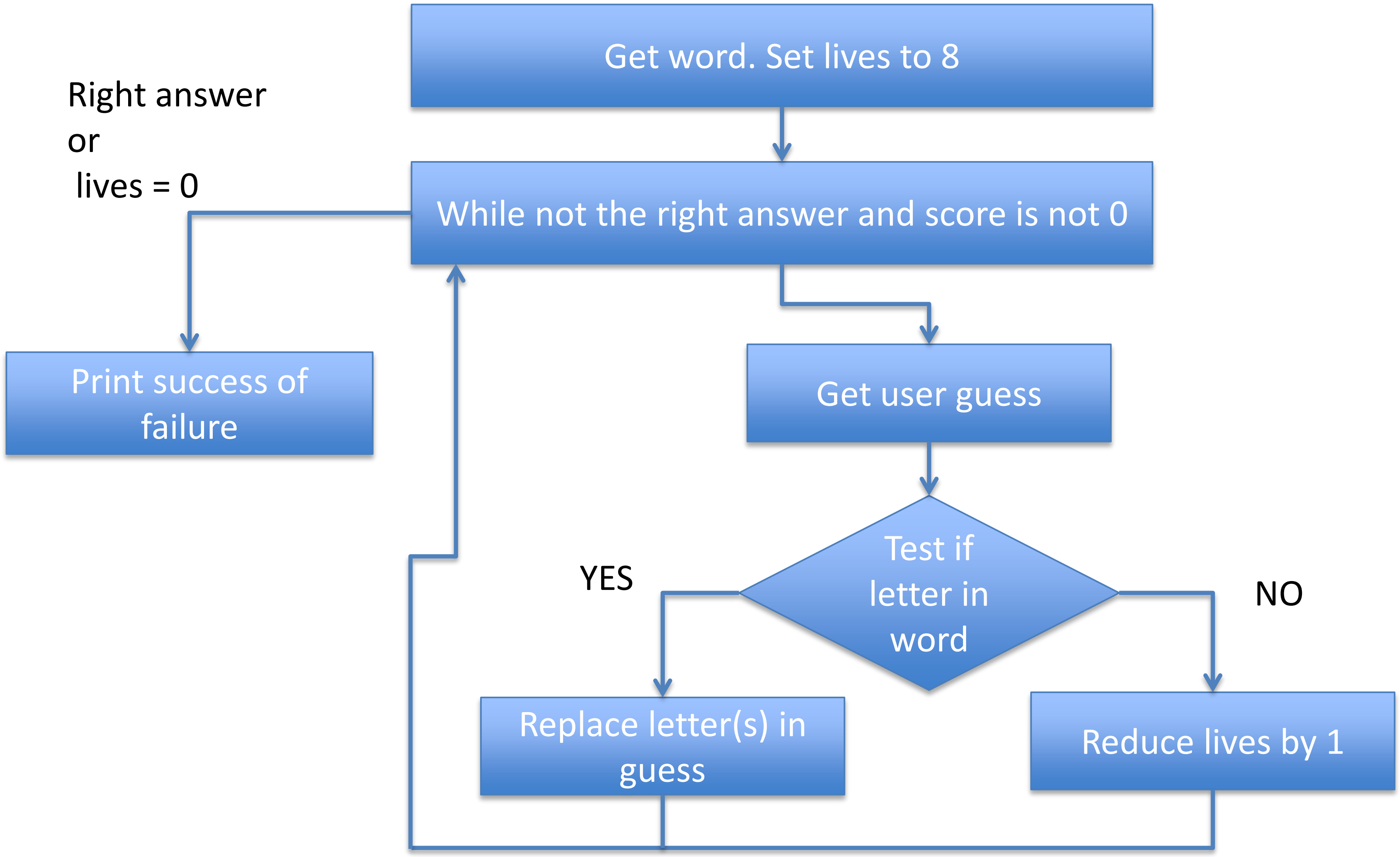
```
a, [2,4],  "_ _ _ _ _ _ _"

                 => "_ _ a _ a _ _"
```

Following the design phase for a hangman game, we need a function to :-

- *Given a letter, a position (or positions) as a list and a word; replace the character in the word at the given position(s) with the letter.*

Hangman Part 4



Hangman 4

- Code the remaining parts to tie the functions together.