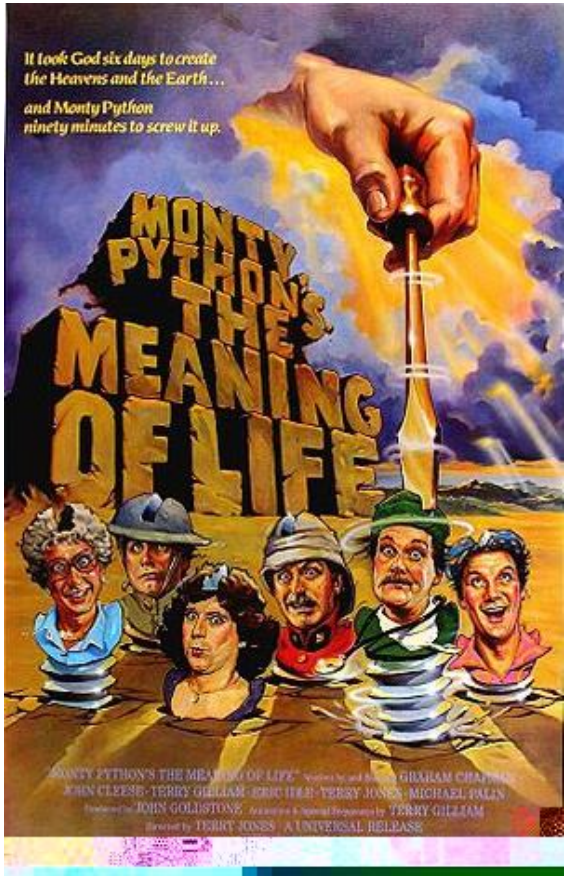
Python Strings



String

- A String is a sequence of characters
- Access characters one at a time with a bracket operator and an offset index

```
>>> fruit = 'banana'
>>>letter = fruit[1]
>>>print (letter)
a
```

  - Index must be an integer

```
>>>letter = fruit[1.5]
TypeError: string indices must be integers, not float
```

len

- Built-in function returns number of characters in a string

```
>>>fruit = 'banana'
>>> len(fruit)
6
```

- Getting the last character in a string the wrong way

```
>>> length = len(fruit)
>>> last = fruit[length]
IndexError: string index of range
```

- Getting the last character in a string the right way

```
>>> last= len(fruit-1)
>>> print (last)
a
```

Traversal

```
index = 0
while index < len(fruit):
    letter = fruit[index]
    print (letter)
    index = index + 1

for char in fruit
    print (char)
```

String slices

- A slice is a segment of a string
  - Selecting a slice is similar to selecting a char

```
>>>s = 'Monty Python'
>>> print (s[0:5])   #returns the 0 to the 4th char
Monty
```

- Can omit the first index

```
>>>s = 'Monty Python'
>>> print (s[:5])    #returns the 0 to the 4th char
Monty
```

- Can omit the second index

```
>>>s = 'Monty Python'
>>> print (s[6:])    #returns the 6th to the last char
Python
```

- If the first index is >= second, returns the empty string

Strings are immutable

- Don’t use [] on left side of assignment operator to attempt to change a char in the string

```
>>> greeting = ‘Helli’
>>> greeting[4] = ‘o’
TypeError: ‘str’ object does not support item
assignment
```

- You cannot change an existing immutable object.
  - Create a new one (and reassign)

```
>>> greeting = ‘Helli’
>>> greeting2 = greeting[0:4]+’o’
>>> print (greeting2)
Hello
```

String Methods

- upper(word)

```
>>>word = 'flower'
>>>upword = word.upper()
>>>print (upword)
FLOWER
```

- find(char)

```
>>>word = 'flower'
>>>index = word.find('o')
>>>print (index)
2
```

- find(string)

```
>>>word = 'flower'
>>>index = word.find('ow')
>>>print (index)
2
```

- find(string, start, end)

`String` operators

- `in (s1, s2)`
  - Takes two strings and returns True if the first appears in the second

```
>>> 'a' in 'banana'
True
>>> 'seed' in 'banana'
False
```

- Relational operators work on strings

```
>>> if word == 'banana'
print ('All right, bananas.')
```

Problem: Palindrome

```
# Program to check if a string
#  is palindrome or not

# take input from the user
my_str = input("Enter a string: ")

# make it suitable for caseless comparison
my_str = my_str.casefold()

# reverse the string
rev_str = reversed(my_str)

# check if the string is equal to its reverse
if list(my_str) == list(rev_str):
   print("It is palindrome")
else:
   print("It is not palindrome")
```