Strings

Strings

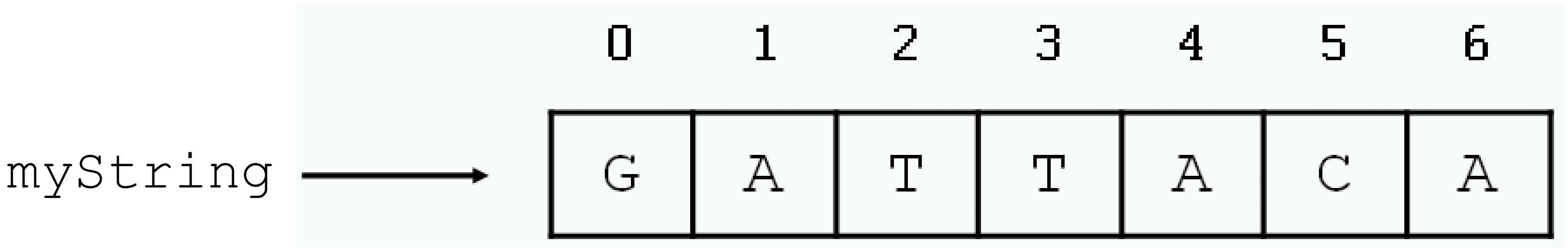
- A string is a sequence of letters (called characters).
- In Python, strings start and end with single or double quotes.

```
>>> “foo”
‘foo’
>>> ‘foo’
‘foo’
```

Defining strings

- Each string is stored in the computer’s memory as a list of characters.

```
>>> myString = “GATTACA”
```



Accessing single characters

- You can access individual characters by using indices in square brackets.
- The index[ ] operator

```
>>> myString = “GATTACA”
>>> myString[0]
‘G’
>>> myString[1]
‘A’
>>> myString[-1]
‘A’
>>> myString[-2]
‘C’
>>> myString[7]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
IndexError: string index out of range
```

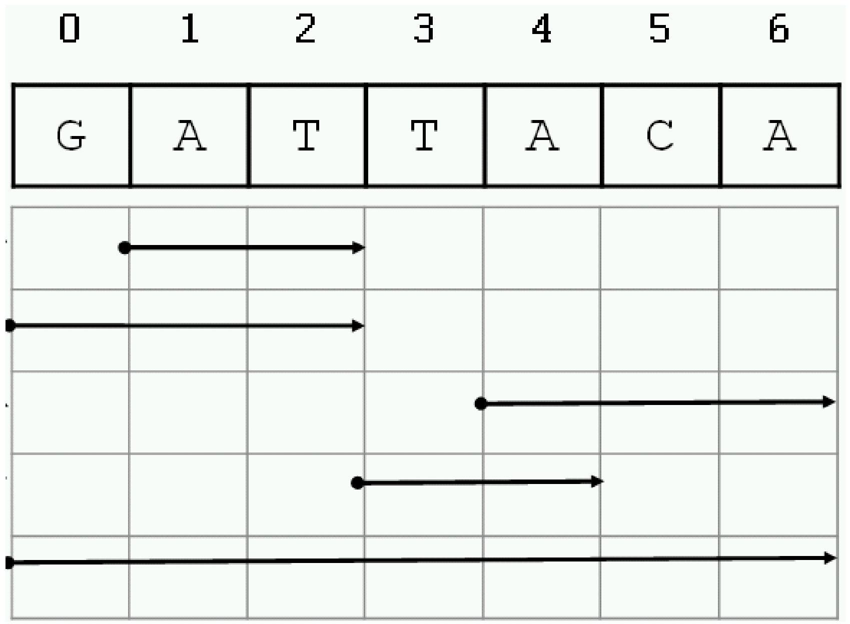
Negative indices start at the end of the string and move left.

STRING OPERATORS

- The string slicing operator [start:end]
  - To select a portion of a string
- String slicing with step size [start_index:end_index:step_size]
  - To select characters from a string with step size
- String +,* and in Operators
  - Concatenation, repetition operator(multiplication operator) and substring checking

Accessing substrings

```
The string slicing operator [start:end]
>>> myString = “GATTACA”
>>> myString[1:3]
‘AT’
>>> myString[:3]
‘GAT’
>>> myString[4:]
‘ACA’
>>> myString[3:5]
‘TA’
>>> myString[:]
‘GATTACA’
```



String slicing with step size
[start_index:end_index:step_size]

```
>>> myString = “GATTACA”
>>> myString[0:len(myString):2]
‘GTAA’
>>>myString[0:len(myString):3]
‘GTA’
myString[1:5:2]
‘AT’
```

More string functionality

```
>>> len(“GATTACA”)
7
>>> “GAT” + “TACA”
‘GATTACA’
>>> “A” * 10
‘AAAAAAAAAA
>>> “GAT” in “GATTACA”
True
>>> “AGT” in “GATTACA”
False
```

← Length

← Concatenation

← Repeat

← Substring test

STRING OPERATIONS

- String comparison (==,>,<,>=,<= and !=)
- Format() method
- Split() method

String Comparison

- String comparison (==,>,<,>=,<= and !=)
- Python compares the string by comparing numerical values of individual characters.
- Returns True or False

```
>>>s1="abc"
>>>s2="ABC"
>>>s1>s2        //a=97 ,A=65
True
```

format() method

- Programmers can include %s inside a string and follow it with a list of values for each %.

```
        >>>”My name is %s and I am from 
%s”%(“Amrita”,”USA”)
```

- Using format():

```
        >>>”My name is {}and I am from 
            {}”.format(“Amrita”,”USA”)
```

- Using index as arguments in{}

  >>>”My name is {0}and I am from {1}”.format(“Amrita”,”USA”)

- Using Keyword arguments

  >>>”My name is {0}and I am from {country}”.format(“Amrita”,country=”USA”)

split() method

- Used to breakup a string into smaller strings.
- You can specify the separator, default separator is any whitespace.
- *Syntax: string*.split(*separator, maxsplit*)
  - separator: optional. Specifies the separator to use when splitting the string. Default is whitespace.
  - Maxsplit: optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

```
>>>txt = "apple#banana#cherry#orange“
>>>x = txt.split()
>>>print(x) //
[‘apple’,’banana’,’cherry’,’orange’]
>>>y = txt.split("#", 1) // # setting the maxsplit parameter to 1, will
                              return a list with 2 elements!
>>>print(y)
[‘apple’,’#banana#cherry#orange’]
```

String methods

- In Python, a method is a function that is defined with respect to a particular object.
- The syntax is
  `<object>.<method>(<parameters>)`

```
>>> dna = “ACGT”
>>> dna.find(“T”)
3
```

String methods

```
>>> "GATTACA".find("ATT")
1
>>> "GATTACA".count("T")
2
>>> "GATTACA".lower()
'gattaca'
>>> "gattaca".upper()
'GATTACA'
>>> "GATTACA".replace("G", "U")
'UATTACA`
>>> "GATTACA".replace("C", "U")
'GATTAUA'
>>> "GATTACA".replace("AT", "**")
'G**TACA'
>>> "GATTACA".startswith("G")
True
>>> "GATTACA".startswith("g")
False
```

Strings are immutable

- Strings cannot be modified; instead, create a new one.

```
>>> s = "GATTACA"
>>> s[3] = "C"
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
>>> s = s[:3] + "C" + s[4:]
>>> s
'GATCACA'
>>> s = s.replace("G","U")
>>> s
'UATCACA'
```

Strings are immutable

- String methods do not modify the string; they return a new string.

```
>>> sequence = “ACGT”
>>> sequence.replace(“A”, “G”)
‘GCGT’
>>> print sequence
ACGT

>>> sequence = “ACGT”
>>> new_sequence = sequence.replace(“A”, “G”)
>>> print new_sequence
GCGT
```

String summary

Basic string operations:

```
S = "AATTGG"                  # assignment - or use single quotes ' '
s1 + s2                       # concatenate
s2 * 3                        # repeat string
s2[i]                         # index character at position 'i'
s2[x:y]                       # index a substring
len(S)                        # get length of string
int(S)  # or use float(S)     # turn a string into an integer or floating point decimal
```

Methods:

```
S.upper()
S.lower()
S.count(substring)
S.replace(old,new)
S.find(substring)
S.startswith(substring), S. endswith(substring)
```

Printing:

```
print var1,var2,var3          # print multiple variables
print "text",var1,"text"      # print a combination of explicit text (strings) and variables
```