Introduction to Strings

- Strings are used in Python to record text information, such as names. It could either be a word, a phrase, a sentence, a paragraph or an entire encyclopedia. Strings in Python are actually a *sequence*, which basically means Python keeps track of every element in the string as a sequence. For example, Python understands the string "joker' to be a sequence of letters in a specific order. This means we will be able to use indexing to grab particular letters (like the first letter, or the last letter).
- This idea of a sequence is an important one in Python and we will touch upon it later on in the future.

Creating a String

- To create a string in Python you need to use either single quotes or double quotes. For example:

```
# Single word
my_first_string= 'algebra'
my_first_string
```

```
'algebra'
```

```
# Entire phrase
phrase = 'Statistics sits at the heart of machine learning'
print(phrase)
```

```
Statistics sits at the heart of machine learning
```

```
# Statement to get the type of the variable
type(phrase)
```

```
str
```

```
# We can also use double quote
my_string = "String built with double quotes"
print(my_string)   # Use the print command
```

```
String built with double quotes
```

```
# Be careful with quotes!
sentence= 'I'm using single quotes, but this will create an error'
print(sentence)
```

```
I'm using single quotes, but this will create an error
```

- The reason for the error above is because the single quote in `I'm` stopped the string. You can use combinations of double and single quotes to get the complete statement.

```
sentence= "I'm using single quotes, but this will create an error"
print(sentence)
```

```
I'm using single quotes, but this will create an error
```

```
hashtag = "#"
print(hashtag)
```

```
#
```

```
type(hashtag)
```

```
str
```

How to print strings

- Using Jupyter notebook with just a string in a cell will automatically output strings, but the correct way to display strings in your output is by using a print function.

```
# We can simply declare a string
'Deep Learning'
```

```
# Note that we can't output multiple strings this way
'Linear Algebra'
'Calculus'
```

We can use the `print()` statement to print a string.

```
# print('Linear Algebra')
# print('Calculus')
print('Use \n to print a new line')
print('\n')
print('See what \n I mean?')
```

```
Use
 to print a new line


See what
 I mean?
```

Playing with strings

- We can also use a function called `len()` to check the length of a string!

```
algo = 'regres sion '
len(algo)
```

```
12
```

Python's built-in len() function counts all of the characters in the string, including spaces and punctuation.

String Indexing

- We know strings are a sequence, which means Python can use indexes to call parts of the sequence.
- A string index refers to the location of an element present in a string.
- The indexing begins from 0 in Python.
- The first element is assigned an index 0, the second element is assigned an index of 1 and so on and so forth.
- In Python, we use brackets `[]` after an object to call its index.

[ ] ↳ 8 cells hidden

String Slicing

[ ] ↳ 25 cells hidden

String Properties

- It's important to note that strings have an important property known as *immutability.*
- This means that once a string is created, the elements within it can not be changed or replaced via item assignment. We will see how we can do such operation using string methods

```
# Can we change our string 'Hello' to 'Cello'? Lets try replacing the first letter H with C
string='Hello'
```

```
string[0] = 'C'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-76-f55372fc219f> in <module>()
      1 # Can we change our string 'Hello' to 'Cello'? Lets try replacing the first letter H with C
      2 string='Hello'
----> 3 string[0] = 'C'

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

- Notice how the error tells us directly what we can't do, that is we can't change the item assignment!
- Something we *can* do is concatenate strings!

```
# Concatenate strings!

string1='abc'
string2='def'
print(string1 + ' ' + string2 )
```

```
abc def
```

```
print(string1 + 4 + string2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-79-34b0f8d1c2ec> in <module>()
----> 1 print(string1 + 4 + string2 )

TypeError: can only concatenate str (not "int") to str
```

- To convert an integer into a string, you can use the `str()` function or you can simply write the number in quotes

```
# Concatenate strings!

string1='abc'
string2='def'
num = 4
print(string1 + str(4) + string2)
```

```
abc4def
```

```
str(num)
```

```
'4'
```

```
# Concatenate strings!
string1='abc'
string2='def'
string1 + '4'+ string2
```

```
'abc4def'
```

```
print(string)
```

```
Hello
```

```
# We can reassign string completely though!
string = string + ' concatenate me!'
print(string)
```

```
Hello concatenate me! concatenate me!
```

```
letters = 'wubba'
letters*3
```

```
'wubbawubbawubba'
```

String functions and methods

```
algorithm = 'Neural Networks'
```

```
print(algorithm)
```

```
Neural Networks
```

`len()`

- `len()` function returns the length of the string

[ ] ↳ 1 cell hidden

`lower()`

- `lower()` method converts the string to lowercase

[ ] ↳ 4 cells hidden

`upper()`

- `upper()` method converts the string to uppercase

[ ] ↳ 2 cells hidden

`count()`

- `count()` method returns the count of a string in the given string. Unlike `lower()` and `upper()` method, the `count()` method takes a string as an argument

[ ] ↳ 6 cells hidden

`find()`

- `find()` method returns the index of the first occurrence of a string present in a given string. Similar to the `count()` method, the `find()` method takes a string as an argument

[ ] ↳ 5 cells hidden

`replace()`

- `replace()` method takes two arguments - (i) the string to replace and (ii) the string to replace with, and returns a modified string after the operation

[ ] ↳ 6 cells hidden

Printing strings a bit differently

[ ] ↳ 5 cells hidden

Check if a string contains a particular word or character

```
my_string = 'Albert Einstein'
```

```
'Albert' in my_string
```

```
True
```

```
'Alberta' in my_string
```

```
False
```

Start coding or generate with AI.