Python Programming Basics

Goals

- In this module you will learn a bit about:
  - strings in Python3
  - Displaying output with the **print** function

  - Obtaining input from the keyboard with the **input** function

Creating a string

- To create a string in Python put either a single or double quote around a value (i.e. letter, digit or symbol)

Examples:

```
>>> 'Ann San Mateo'
'Ann San Mateo‘
>>> "Chuck Canada"
'Chuck Canada‘
>>> '5'
'5'
>>> "******"
'******'
```

String Quote Pairs

- Opening and Closing string quotes must match

  Example:

```
>>> 'Ann San Mateo"
  File "<stdin>", line 1
    'Ann San Mateo"
              ^
SyntaxError: EOL while scanning string literal
```

Special Characters in Strings

- To enclose a single quote inside a string use double quotes around the string

  Example:

```
>>> "Chuck's"
"Chuck's"
```

- To enclose a double quote inside a string use single quotes around the string

  Example:

```
>>> '"nary a day" goes by'
'"nary a day" goes by'
```

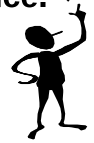
Nesting Quotes

- To enclose both single and double quotes in one string, you can use the escape character \

Example:

```
>>> 'The director said, "That\'s a wrap folks."'
'The director said, "That\'s a wrap folks."'
```

The combination of the backslash and the single \’ quote is called an escape sequence.



Maximum Line Length

- Keep all line lengths < 80 characters
  - This is to avoid text-wrap around in some editors
  - One option is to use the backslash character

  Example:

```
$ cat longline.py
print("No line of source code should have more than 79 characters; " \
      "lines longer than this can use the continuation character.")
```

Multiline Strings

- Creating a single string using single or double quotes requires that the entire string fit onto a single line

Example:

```
>>> 'line1
  File "<stdin>", line 1
    'line1
         ^
SyntaxError: EOL while scanning string literal
```

Multiline Strings

- To successfully span multiple lines with a single string, you can use either 3 single quotes or 3 double quotes around the string
- Examples:

```
>>> '''line1
... line2'''
'line1\nline2'
>>> """line1
... line2"""
'line1\nline2'
```

print function

- Python provides a number of built-in functions for us
  - You can use the **print** function to display output

  Examples:

```
>>> print("Hello")
Hello
>>> print(1 + 2)
3
>>> print("One times two equals", 1 * 2)
One times two equals 2
```

help documentation

- You can use the help documentation to learn more

Example:

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    …..
```

input function

- Another useful Python built-in function is **input**
  - You can use the **input** function to read a single line of text from the keyboard
  - **Note:** Whatever the user enters in is returned as a string

```
>>> codename = input()
Ann San Mateo
>>> codename
'Ann San Mateo'
>>> age = input()
25
>>> age
'25'
```

Numerical Input

- The **input** function can only obtain a string of text from the user
- To read an integer value, use the **input** function to obtain the data and then convert the string to an integer using the **int** function

Example:

```
>>> age = input()
25
>>> age
'25'
>>> age = int(age)
>>> age
25
```

Summary

- Type **str** represents a string
- Strings can be created by using a matching pair of single or double quotes
- Special characters can be included using an escape sequence in the string
- Values can be printed using the built-in **print** function
- Strings can be obtained from the keyboard using the built-in **input** function