

String: Text as a Value

- String are quoted characters
 - 'abc d' (Python prefers)
 - "abc d" (most languages)

Type: str

- How to write quotes in quotes?
 - Delineate with "other quote"
 - Example:** "' ' or ' '"
 - What if need both " and ' ?

Char	Meaning
\'	single quote
\"	double quote
\n	new line
\t	tab
\\	backslash

- Solution:** escape characters
 - Format: \ + letter
 - Special or invisible chars

String are Indexed

s = 'abc d'

0	1	2	3	4
a	b	c		d

s = 'Hello all'

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with []
 - s[0] is 'a'
 - s[4] is 'd'
 - s[5] **causes an error**
 - s[0:2] is 'ab' (excludes e)
 - s[2:] is 'cd'

What is s[3:6]?

A:	'o all'
B:	'ello'
C:	'ello'
D:	'o '
E:	I do not know

- Called "string slicing"

String are Indexed

s = 'abc d'

0	1	2	3	4
a	b	c		d

s = 'Hello all'

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with []
 - s[0] is 'a'
 - s[4] is 'd'
 - s[5] **causes an error**
 - s[0:2] is 'ab' (excludes e)
 - s[2:] is 'cd'
- Called "string slicing"

What is s[4:]?

A:	'o all'
B:	'Hello'
C:	'Hell'
D:	Error!
E:	I do not know

Other Things We Can Do With Strings

- Operation** in: s₁ in s₂
 - Tests if s₁ "a part of" s₂
 - Say s₁ a *substring* of s₂
 - Evaluates to a bool

- Function** len: len(s)
 - Value is # of chars in s
 - Evaluates to an int

Examples:

- s = 'abracadabra'
- 'a' in s == True
- 'ead' in s == True
- 'foo' in s == False

Examples:

- s = 'abracadabra'
- len(s) == 11
- len(s[1:5]) == 4
- s[1:len(s)-1] == 'bracadabr'

Function Calls

- Python supports expressions with math-like functions
 - A function in an expression is a **function call**
 - Will explain the meaning of this later
- Function expressions have the form **fun(x,y,...)**

function name	argument
---------------	----------

- Examples** (math functions that work in Python):
 - round(2.34)
 - max(a+3, 24)

Arguments can be any **expression**

Built-In Functions

- You have seen many functions already
 - Type casting functions: `int()`, `float()`, `bool()`
 - Dynamically type an expression: `type()`
 - Help function: `help()`
- Getting user input: `raw_input()`
- `print <string>` is **not** a function call
 - It is simply a statement (like assignment)
 - But it is in Python 3.x: `print(<string>)`

Arguments go in (), but name() refers to function in general

Method: A Special Type of Function

- Methods are unique (right now) to strings
- Like a function call with a “string in front”
 - Usage: *string.method(x.y...)*
 - The string is an *implicit argument*
- Example: upper()
 - `s = 'Hello World'`
 - `s.upper() == 'HELLO WORLD'`
 - `s[1:5].upper() == 'ELLO'`
 - `'abc'.upper() == 'ABC'`

Will see why we do it this way later in course

Examples of String Methods

- `s1.index(s2)`
 - Position of the first instance of `s2` in `s1`
- `s1.count(s2)`
 - Number of times `s2` appears inside of `s1`
- `s.strip()`
 - A copy of `s` with white-space removed at ends

- `s = 'abracadabra'`
- `s.index('a') == 0`
- `s.index('rac') == 2`
- `s.count('a') == 5`

- `' a b '.strip() == 'a b'`

See Python Docs for more

Built-in Functions vs Modules

- The number of built-in functions is small
 - <http://docs.python.org/2/library/functions.html>
- Missing a lot of functions you would expect
 - **Example:** `cos()`, `sqrt()`
- **Module:** file that contains Python code
 - A way for Python to provide optional functions
 - To access a module, the `import` command
 - Access the functions using module as a *prefix*

Example: Module math

```
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

To access math functions

Functions require math prefix!

Module has variables too!

Other Modules

- `io`
 - Read/write from files
- `random`
 - Generate random numbers
 - Can pick any distribution
- `string`
 - Useful string functions
- `sys`
 - Information about your OS

Reading the Python Documentation

The screenshot shows the Python documentation for `math.ceil(x)`. Callouts identify:

- Function name:** `math.ceil(x)`
- Possible arguments:** `x`
- Module:** `math`
- What the function evaluates to:** Return the ceiling of `x` as a float, the smallest integer value greater than or equal to `x`.

 A URL <http://docs.python.org/library> is also visible at the bottom.

Using the from Keyword

```
>>> import math
>>> math.pi
3.141592653589793
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
-1.0
```

Must prefix with module name

No prefix needed for variable pi

No prefix needed for anything in math

- Be careful using `from!`
- Using `import` is *safer*
 - Modules might conflict (functions w/ same name)
 - What if `import` both?
- **Example:** Turtles
 - Older version of A4
 - 2 modules: `turtle`, `tkturtle`
 - Both have `func. Turtle()`