Week 1, Lesson 2

More Command Line, and the basics of Python

Review

Homework & Last Lesson

Homework:

- Finish Command Line Exercises

Last Lesson:

- Tools
- Console

A few extra things

- Bash Profile
  - `.bash_profile`
  - `.bashrc`
- PS1
  - `http://ezprompt.net/`
- Colors
  - `export CLICOLOR=1`
- Other Shells
  - SH
  - CSH
  - ZSH
  - FSH

Visual Studio Code

- Also called VSCode
- Install Shell Command

Basic Programming Concepts

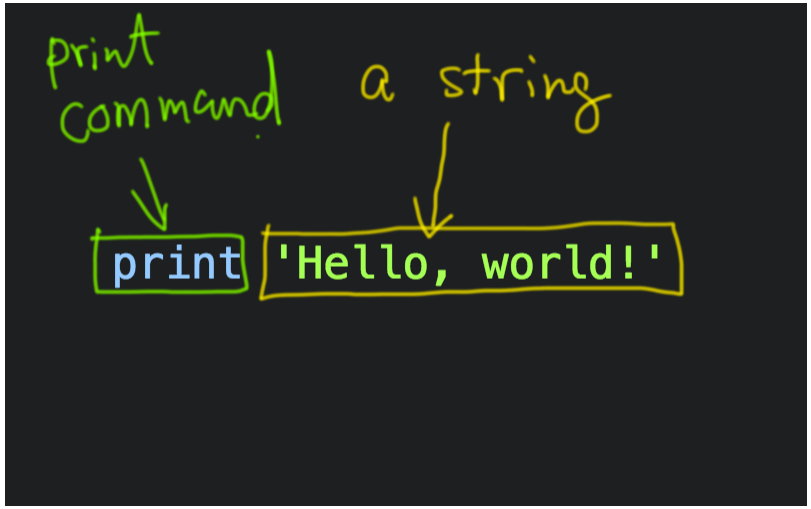
- Variables
- Strings
- Numbers
- Printing
- User input
- String formatting
- If statements
- Booleans
- While loops

Printing

Print Statement

```
print 'Hello, world!'
```

Print Statement



Print Statement

```
print 'Hello'
print 'World'
```

Outputs:

```
Hello
World
```

Print Multiple Values

```
print 'One', 'Two', 'Three'
```

Outputs:

```
One Two Three
```

Print Statement without Newline

```
print 'One',
print 'Two',
print 'Three',
```

Outputs:

```
One Two Three
```

Print Statement

```
import sys
sys.stdout.write('Hello')
sys.stdout.write('Hello')
sys.stdout.write('Hello')
```

Outputs:

```
HelloHelloHello
```

Strings

String Literals

```
"I am a string."
```

String Literals

```
'I am a string too.'
```

String Literals

```
"I'm a string and I have a single quote."
```

String Literals

```
'I\'m a string and I have to escape my single quote.'
```

Multi-line String Literals

```
"""
I am a string
and I can
span multiple lines!
"""
```

Concatenating Strings

```
'abc' + 'def'
```

Gives you:

```
'abcdef'
```

Newline Characters

```
'I am one line.\nI am another line.'
```

Variables

Variables

LHS

variable name (identifier)

```
name = 'Amjad'
```

RHS value (happens to be a string)

Variables

```
age = 5
```

RHS
value is
a number

Variables

```
numbers = [7, 83, 23, 48, 2, 6]
```

RHS is a list

Variables

- LHS is a name a.k.a an identifier
- RHS is an expression

Identifiers

- used to identify a variable, function, class, or module
- starts with a letter between `a-z` and `A-Z` or an underscore: `_`
- followed by a letter between `a-z` and `A-Z` or an underscore, or any digit between `0-9`
- but an identifier cannot be a reserved word

Python Reserved Words

```
and
assert
break
class
continue
def
del
elif
else
except
exec
finally
for
from
global
if
import
in
is
lambda
not
or
pass
print
raise
return
try
while
with
yield
```

Valid Identifiers	Invalid Identifiers
name	isTrue?
first_name	class
_name	$name
name135	135name
lastName	first-name

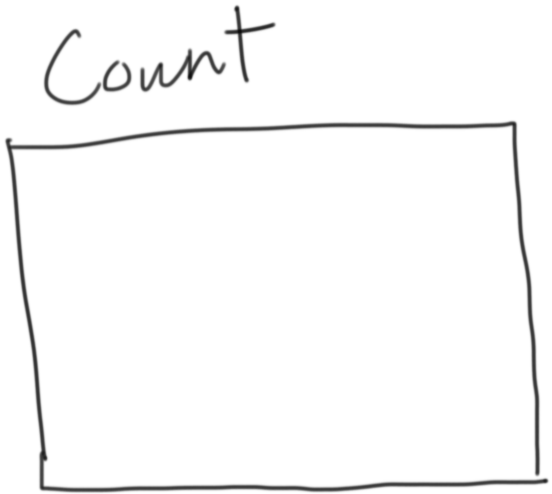
Expressions

- Expressions are syntax that return a value
- Examples:
  - A string literal is an expression: `‘Kelly’`
  - A number literal is an expression: `9`
  - A list literal is an expression: `[1, 4, 5, 8]`
  - A function call is an expression: `square(2)`

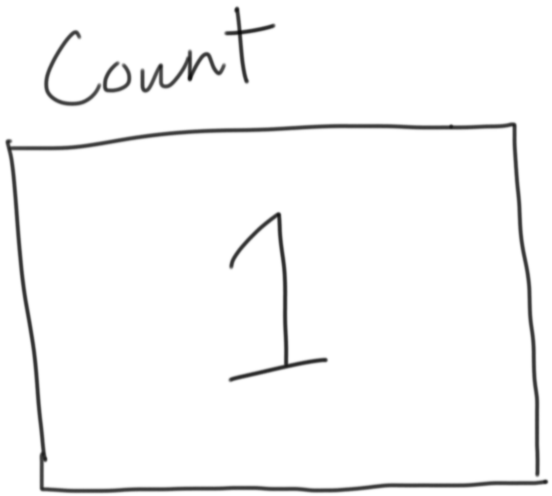
Variables: 3 Ways

- A container
- A placeholder
- An alias

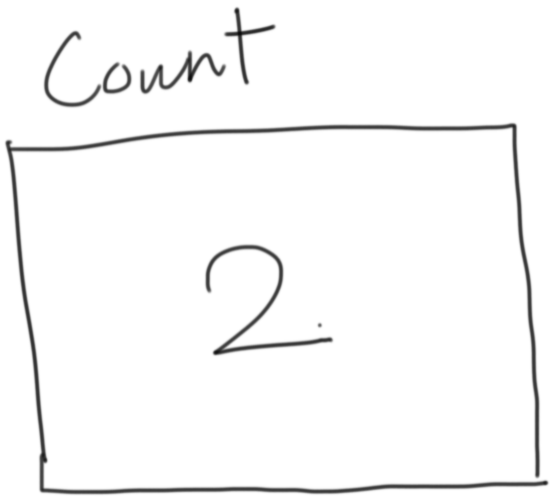
Variables as Containers



Variables as Containers

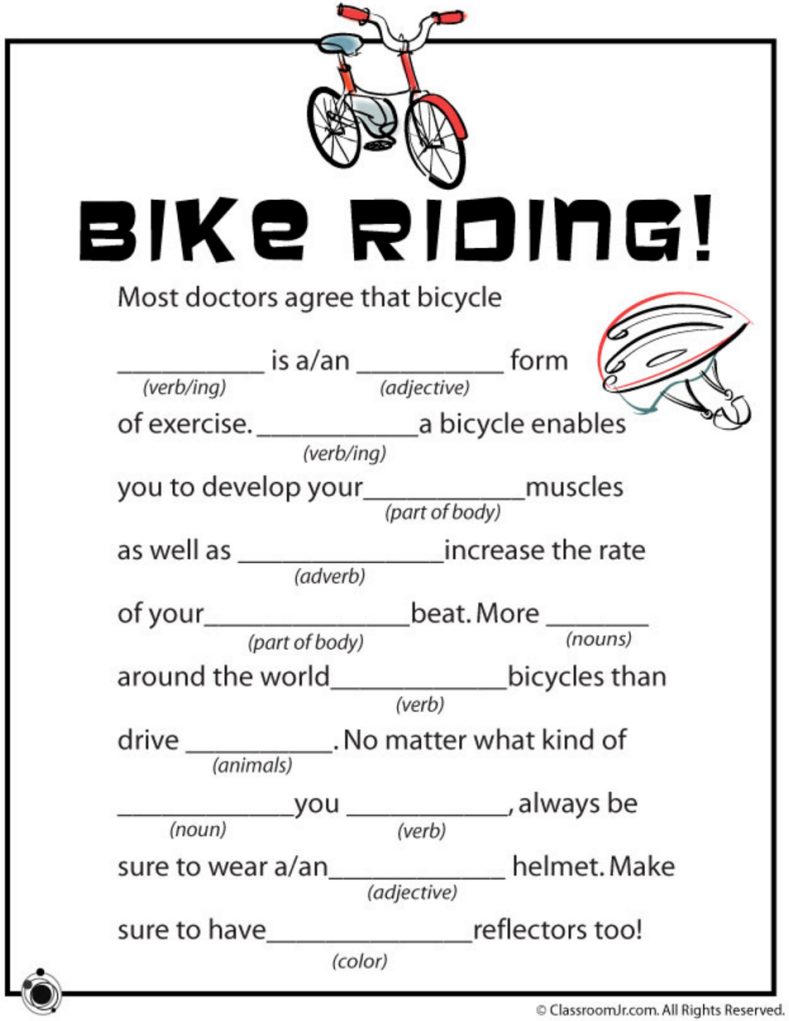


Variables as Containers



See that in Python Tutor!

Variables as placeholders



BIKE RIDING!

Most doctors agree that bicycle

_____ is a/an _____ form
(verb/ing) (adjective)

of exercise. _____a bicycle enables
(verb/ing)

you to develop your_____muscles
(part of body)

as well as _____increase the rate
(adverb)

of your_____beat. More _____
(part of body) (nouns)

around the world_____bicycles than
(verb)

drive _____. No matter what kind of
(animals)

_____you _____, always be
(noun) (verb)

sure to wear a/an_____ helmet. Make
(adjective)

sure to have_____reflectors too!
(color)

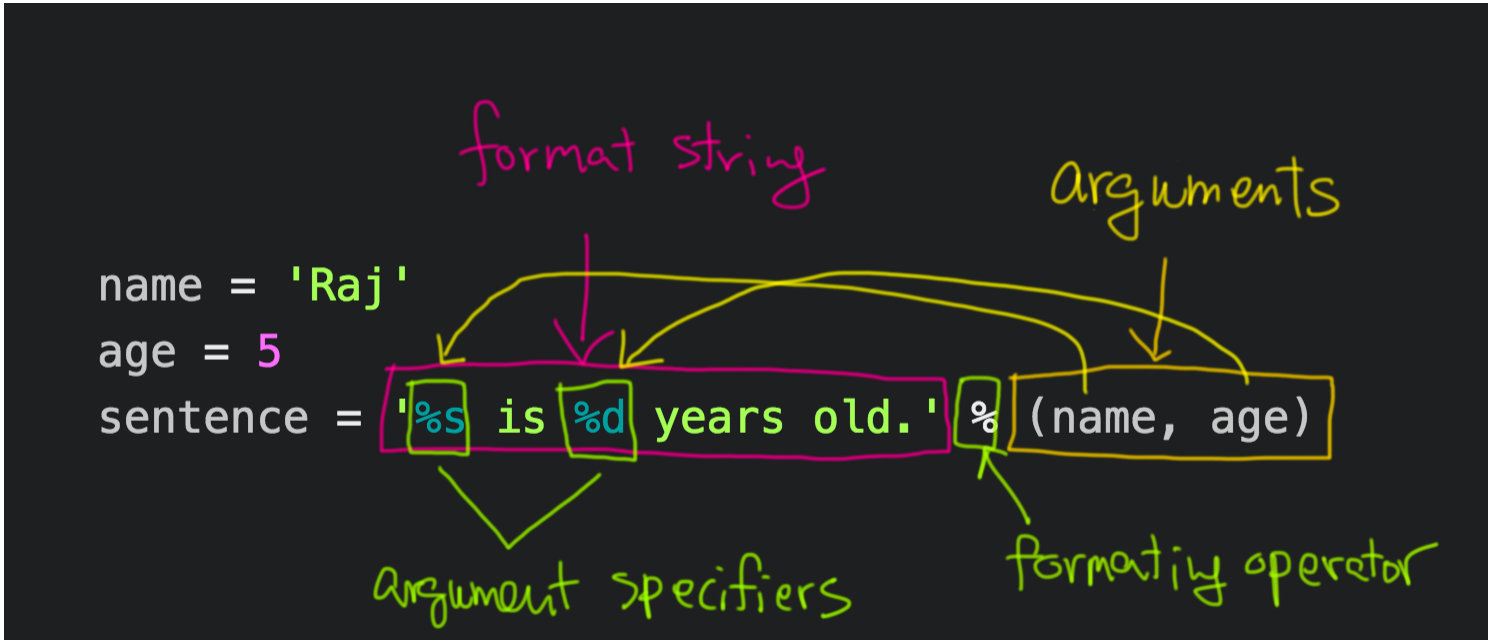
Variables as placeholders

```
verb = 'riding'
adjective = 'great'
sentence = 'Most doctors agree that bicycle %s is
a/an %s form of exercise.' % (verb, adjective)
```

Variables as placeholders

```
name = 'Raj'
age = 5
sentence = '%s is %d years old.' % (name, age)
```

Python String Formatting



Only one argument

```
name = 'Raj'
sentence = 'I am %s' % name
```

Variables as Aliases

```
name = 'Raj'
age = 5
print '%s is %d years old.' % (name, age)
```

Variables as Aliases

```
name = 'Raj'
age = 5
print '%s is %d years old.' % (name, age)
```

alias this expression

Variables as Aliases

```
name = 'Raj'
age = 5
sentence = '%s is %d years old.' % (name, age)
print sentence
```

Variables as Aliases

```
name = 'Raj'
age = 5
sentence = '%s is %d years old.' % (name, age)
print sentence
```

the alias

Can alias *any* expression

Numbers

Number literals

```
8    # integers (int)
5.5  # floating point numbers (float)
```

Integer Division Returns Integer

```
8 / 3 # you get 2
```

Use Floats

```
8.0 / 3 # you get 2.6666666666666665
```

Comparisons

- `5 > 4`
- `9 < 5`
- `7 >= 9`
- `4 <= 8`

Convert a String to a Number

```
string = '45'
num = int(string)
```

Convert a String to a Number

```
int('blah')
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'blah'
```

Arithmetic

- `2 * 3`
- `1 + 1`
- `3 - 2`
- `8 / 4`
- `abs(-5)`
- `pow(2, 2)`
- `round(5.5)`

Math Module

- `import math`
- `floor, ceil`
- `sin, cos, tan`
- `sqrt`
- `log`
- `pi, e`

User Input

User Input

```
name = raw_input("What's your name? ")
```

User Input

```
What's your name?
```

User Input

```
What's your name? Toby
```

User Input

```
age = int(raw_input("What's your age? "))
```

If Statements

if Statement

```
if age >= 21:
    print 'You are an adult.'
```

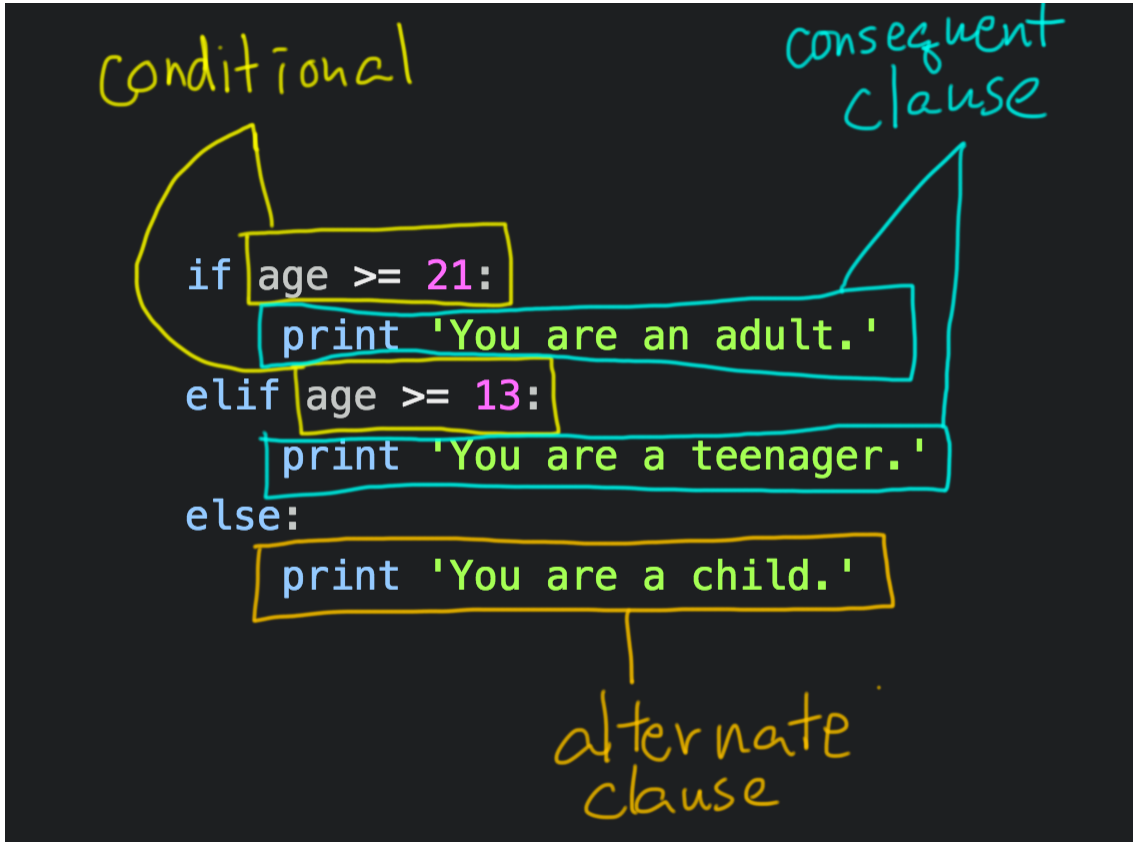
if-elif Statement

```
if age >= 21:
    print 'You are an adult.'
elif age >= 13:
    print 'You are a teenager.'
```

if-elif-else Statement

```
if age >= 21:
    print 'You are an adult.'
elif age >= 13:
    print 'You are a teenager.'
else:
    print 'You are a child.'
```

if-elif-else Statement



Indentation is Important

```
age = 5
if age >= 21:
    print 'You are an adult.'
elif age >= 13:
    print 'You are a teenager.'
else:
    print 'You are a child.'
```

colons start an indentation block

Indentation is Important

```
age = 5
if age >= 21:
print 'You are an adult.'
elif age >= 13:
print 'You are a teenager.'
else:
print 'You are a child.'
```

```
  File "if.py", line 7
    print 'You are an adult.'
        ^
IndentationError: expected an indented block
```

Indentation is Important

```
print 'Hello'
    print 'World'
```

```
  File "if.py", line 6
    print 'World'
    ^
IndentationError: unexpected indent
```

While Loop

While Loop

```
count = 0
while count < 10:
    count = count + 1
    print "count is now %d" % count
print "Done."
```

While Loop

conditional

body.

```
count = 0
while count < 10:
    count = count + 1
    print "count is now %d" % count
print "Done."
```

Run

While Loop

```
say = raw_input('Say when: ')
while say != 'when':
    print "Cheese"
    say = raw_input('Say when: ')
```

While Loop: `break`

```
while True:
    say = raw_input('Say when: ')
    if say == 'when':
        break
    print "Cheese"
```

Booleans

Booleans

- A boolean is a data type that can either be `True` or `False`

Booleans

```
>>> True
True
>>> False
False
```

Booleans

```
if True:
    print "I <3 Python!"


if False:
    print "I hate Python!"
```

Booleans

These resolve to True or False

```
if age >= 21:
    print 'You are an adult.'
elif age >= 13:
    print 'You are a teenager.'
else:
    print 'You are a child.'
```

Booleans

```
>>> age = 18
>>> age >= 21
False
>>> age >= 13
True
```

Booleans

```
>>> age = 18
>>> age >= 21
False
>>> age >= 13
True
```

expressions

Booleans

```
>>> is_adult = age >= 21
>>> is_teenager = age >= 13
>>> is_adult
False
>>> is_teenager
True
```

Booleans

```
if is_adult:
    print "You are an adult."
elif is_teenager:
    print "You are a teenager."
else:
    print "You are a child."
```

Truthiness

```
age = 5
if age:
    print "Python is awesome!"
```

What Happens Here?

```
age = 5
if age:
    print "Python is awesome!"
```

5 is truthy

```
age = 5
if age:
    print "Python is awesome!"
```

Truthiness Rules

- `False`, `0`, `None`, `“”`, and empty collections like `[]` are falsey
- Everything else is truthy

3 Ways to Run Python

- Python Tutor
- Python to run a .py file
- Python Interactive Shell