J233
Coding for Journalists

LECTURER
Soo Oh

PROMPTS

1. Download the lecture notebook on the class website.

2. Sign into https://pollev.com/soooh

If you don't open pollev.com, you won't get participation points for this lecture.

start Zoom recording + captions

Agenda

Review

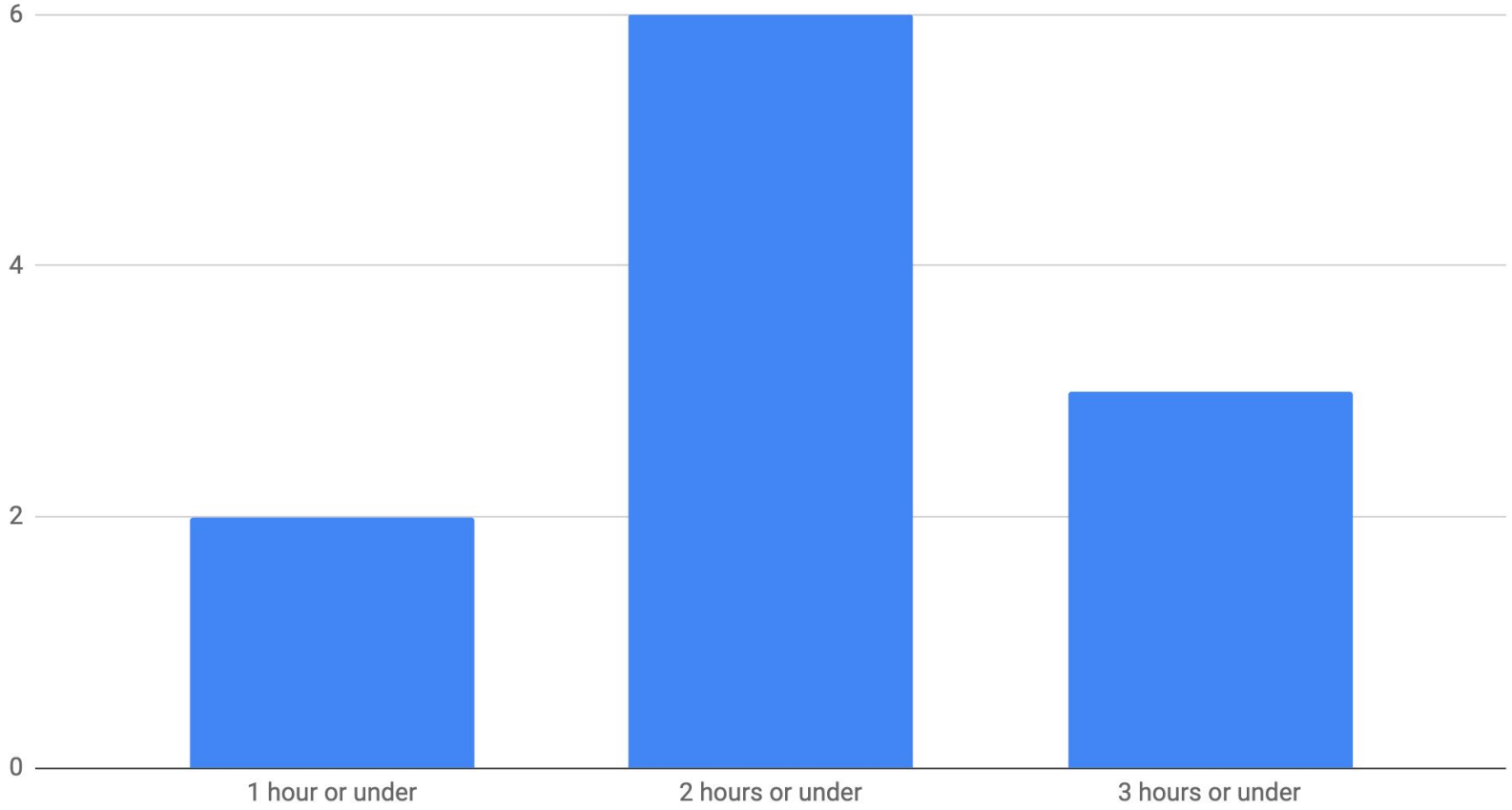
- Homework
- Final Project Qs

Lecture

- Operators
- Variable types
- Functions

BREAK at halfway point

Week of 0828: students grouped by time spent outside of lecture and office hours



Homework Review

Markdown

Jupyter cells

Markdown

- To make a list, you don't have to put spaces between lines
- Add two spaces at the end of a line to have a single line break
- Group blocks of text together in a single Markdown block — otherwise, there will be distinct line breaks.

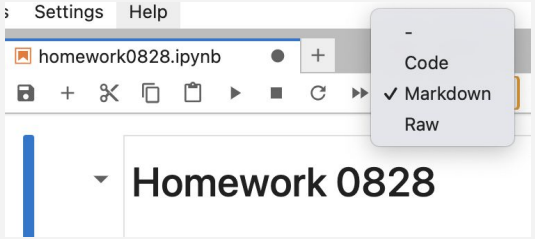
Homework Review

Markdown

Jupyter cells

Jupyter cells

Make sure to check whether a cell is **Code** or **Markdown**.



If you see `[ ]` to the left of a cell, it is definitely a code cell.

```
[ ]: This is a code cell
```

Final Project

Example

Tips on finding data

Your final notebook may look like this:

Afghanistan Humanitarian Parole Data

This particular dataset (SEQ 48) is responding to item 4 of the original records request: *Data (or other records) that show the number of Humanitarian Parole applications with all relevant fields from Afghan nationals that USCIS rejected between the period of January 1, 2021 to February 1, 2022.*

The data is disaggregated HP applications received by USCIS from 1/1/2021 to 5/1/2022. It includes both Parole I-131 and Govt. Parole. Here are some definitions according to USCIS:

- *Completion Date* : Admin steps completed and case considered complete
- *Decision Date* : Date officer made decision
- *Receipt Date* : Date application submitted.
- *Null (under Officer Case Decision Reason)* : Reason not inputted into system.

(FOIA Request 2022000785)

In [6]:
```
import pandas as pd
from loguru import logger
import datetime
from datetime import datetime, date

%load_ext lab_black
```

```
The lab_black extension is already loaded. To reload it, use:
  %reload_ext lab_black
```

Import data

In [7]:
```
hp_complete = pd.read_csv(
    "../01_inputs/csv/5 - SEQ 48 - REDACTED (Tab 1).csv", dtype=str
)

hp_complete["Source File"] = "SEQ48-CLOSED"

hp_pending = pd.read_csv(
    "../01_inputs/csv/5 - SEQ 48 - REDACTED (Tab 2).csv", dtype=str
)
```

Final Project

Example

Tips on finding data

How to find data

- Try adding “github” to Google searches
- Backup datasets: Look through city and state data portals
- Have at least one backup dataset by mid-October
- Come to office hours to brainstorm

Organizing your projects

Creating and organizing files

Python naming conventions

Before you create a new file...

Think about how you're going to organize your files and then name your notebook something useful.

Organizing your projects

Creating and organizing files

Python naming conventions

Example 1

Root level

- J233
  - week01
    - class exercise
    - sandbox
  - week02
    - assignment
    - class exercise
    - sandbox

Organizing your projects

Creating and organizing files

Python naming conventions

Example 2

Root level

- 2021-08-28
  - sandbox
  - in class
  - homework
- 2021-09-11
  - sandbox
  - in class
  - homework
  - quiz

Organizing your projects

Creating and organizing files

Python naming conventions

Example 3

Root level

- 01 sandbox
- 01 in-class assignment
- 01 homework
- 02 in-class assignment
- 02 quiz

Organizing your projects

Creating and organizing files

Python naming conventions

Bad example

Root level

- 🗄️ 01 sandbox
- 🗄️ 01 homework
- 🗄️ 01 homework copy
- 🗄️ 01 homework copy copy
- 🗄️ 01 homework copy copy FINAL
- 🗄️ 01 homework copy copy FINAL copy REAL FINAL
- 🗄️ 02 class
- 🗄️ 02 quiz
- 🗄️ assignment
- 🗄️ quiz 02

Organizing your projects

Creating and organizing files

Python naming conventions

Naming and organizing conventions

Python files *can* start with a digit, but this is a hassle when you're building software (not writing notebooks).

One bug: don't use quotation marks in your notebook titles.

Let’s spend a little time organizing your J233 files

When you’re done, **upload a screenshot of your file system and naming** to Slack **in the thread** I’m going to create right now.

Screenshot on Mac:
SHIFT-COMMAND-4

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

But first, new things!

Preliminary slides will occasionally have something called “Student version of slide” — pay closer attention to the lecture screen at those times.

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

Student version of slide

```
# Python
x = 7
x = 100
print(x)
Out[]:

x = 68
x = 60
x = 27
x = 94
x = 5
print(x)
Out[]:
```

But first, new things!

Slides will have notes to go to the Jupyter notebook you downloaded for lecture.

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 8
b = 12

a - b
# What is a - b?


a = a * 3
# What is a?


a / b
# What is a / b?
```

Play in Jupyter notebook

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

Before we do anything, it might be helpful if you created a new notebook in Jupyter as a “sandbox.”

https://pollev.com/soooh

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

What is a variable?

It’s a name that stores a value.

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```python
# Python
x = 7
```

```r
# R
x <- 7
```

```javascript
// JavaScript
var x = 7

// JavaScript/ES6
let x = 7
const x = 7
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```python
# Python
x = 7
```

```r
# R
x <- 7
```

```javascript
// JavaScript
var x = 7

// JavaScript/ES6
let x = 7
const x = 7
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```python
# Python
x = 7
```

```r
# R
x <- 7
```

```javascript
// JavaScript
var x = 7

// JavaScript/ES6
let x = 7
const x = 7
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```python
# Python
x = 7
```

```r
# R
x <- 7
```

```javascript
// JavaScript
var x = 7

// JavaScript/ES6
let x = 7
const x = 7
```

RECAP!

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```python
# Python
x = 7
```

```r
# R
x <- 7
```

```javascript
// JavaScript
var x = 7

// JavaScript/ES6
let x = 7
const x = 7
```

RECAP!

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```python
# Python
x = 7
```

```r
# R
x <- 7
```

```javascript
// JavaScript
var x = 7

// JavaScript/ES6
let x = 7
const x = 7
```

RECAP!

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python
x = 7
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python
x = 7
x = 100
print(x)
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python
x = 7
x = 100
print(x)
Out[]: 100
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python
x = 7
x = 100
print(x)
Out[]: 100

x = 68
x = 60
x = 27
x = 94
x = 5
print(x)
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python
x = 7
x = 100
print(x)
Out[]: 100

x = 68
x = 60
x = 27
x = 94
x = 5
print(x)
Out[]: 5
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
x = 7
y = 10

x + y
# What is x + y?
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
x = 7
y = 10

x + y
# What is x + y?
Out[]: 17
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
x = 7
y = 10

x + y
# What is x + y?
Out[]: 17

x = x + 3
# What is x?
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
x = 7
y = 10

x + y
# What is x + y?
Out[]: 17

x = x + 3
# What is x?
Out[]: 10
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python
x = 7
x = 100
print(x)
Out[]: 100

x = 68
x = 60
x = 27
x = 94
x = 5
print(x)
Out[]: 5
```

Let's go back to the older slide for **Assignment Operators**

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
x = 7
y = 10

x + y
# What is x + y?
Out[]: 17

x = x + 3
# What is x?
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
x = 7
y = 10

x + y
# What is x + y?
Out[]: 17

x = x + 3
# What is x?
Out[]: 10
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 8
b = 12

a - b
# What is a - b?


a = a * 3
# What is a?


a / b
# What is a / b?
```

Play in Jupyter notebook

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 8
b = 12

a - b
# What is a - b?
Out[]: -4

a = a * 3
# What is a?
Out[]: 24

a / b
# What is a / b?
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 8
b = 12

a - b
# What is a - b?
Out[]: -4

a = a * 3
# What is a?
Out[]: 24

a / b
# What is a / b?
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 8
b = 12

a - b
# What is a - b?
Out[]: -4

a = a * 3
# What is a?
Out[]: 24

a / b
# What is a / b?
Out[]: 2.0
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
s = 5**3
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
s = 5**3 # this is an exponent (5 * 5 * 5)
Out[]: 125
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
s = 5**3 # this is an exponent (5 * 5 * 5)
Out[]: 125

m = 10 % 3
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
s = 5**3 # this is an exponent (5 * 5 * 5)
Out[]: 125

m = 10 % 3 # this is a modulo operator
Out[]: 1
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 7
a = a + 7
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 7
a = a + 7

# this is a compound operator
# it means "operate on that same variable"
a += 7 # equivalent to `a = a + 7`
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 7
a = a + 7

# this is a compound operator
# it means "operate on that same variable"
a += 7 # equivalent to `a = a + 7`
Out[]: 21
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
a = 7
a = a + 7

# this is a compound operator
# it means "operate on that same variable"
a += 7 # equivalent to `a = a + 7`
Out[]: 21

a -= 7 # subtract 7 from a
a *= 7 # multiply 7 and a
a /= 7 # divide a by 7
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

A **relational operator** tells us if 2 values are `True` or `False`.

You can also call this a **comparison operator.**

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
1 == 1
Out[]: True

1 == 2
Out[]: False

1 != 2
Out[]: True

1 < 2
Out[]: True

1 <= 2
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
1 == 1
Out[]: True

1 == 2
Out[]: False

1 != 2
Out[]: True

1 < 2
Out[]: True

1 <= 2
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
1 == 1 # 1 equals 1
Out[]: True

1 == 2 # 1 equals 2
Out[]: False

1 != 2 # 1 is not equal to 2
Out[]: True

1 < 2  # 1 is less than 2
Out[]: True

1 <= 2 # 1 is less than or equal to 2
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n 🧠 92
Out[]: True

t 🧠 31
Out[]: False

n 🧠 t
Out[]: False

n 🧠 t
Out[]: True
```

https://pollev.com/soooh

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n 🧠 92          ==
Out[]: True     <=
                >=

t 🧠 31
Out[]: False

n 🧠 t
Out[]: False

n 🧠 t
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n 🧠 92
Out[]: True

t 🧠 31
Out[]: False

n 🧠 t
Out[]: False

n 🧠 t
Out[]: True
```

!=
<
>

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n 🧠 92
Out[]: True

t 🧠 31
Out[]: False

n 🧠 t          ==
Out[]: False   <
               <=

n 🧠 t
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == 92
Out[]: True

t == 31
Out[]: False

n == t
Out[]: False

n != t
Out[]: True
```

!=
>=
>

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

Logical operators are also a kind of comparison or relational operator.

You use it to compare True or False values.

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]: False
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]: False

n >= t
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]: False

n >= t
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]: False

n >= t
Out[]: True

(n == t) and (n >= t)
Out[]:

(n == t) or  (n >= t)
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]: False

n >= t
Out[]: True

(n == t) and (n >= t)
Out[]:

(n == t) or (n >= t)
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
n = 92
t = 31

n == t
Out[]: False

n >= t
Out[]: True

(n == t) and (n >= t)
Out[]: False

(n == t) or  (n >= t)
Out[]: True
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
True and True
Out[]:

False and False
Out[]:

True and False
Out[]:

True or False
Out[]:

False or False
Out[]:
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
True and True
Out[]: True

False and False
Out[]: False

True and False
Out[]: False

True or False
Out[]: True

False or False
Out[]: False
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

Naming conventions

```
# Python
ucb_age = 155

# R
ucb.age = 155
ucb_age = 155

# JavaScript
ucbAge = 155
```

RECAP!

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

Naming conventions

```
# Python
ucb_age = 155 # snake_case

# R
ucb.age = 155 # historic, risk of confusion
ucb_age = 155

# JavaScript
ucbAge = 155 # camelCase
```

RECAP!

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

Formatting is arbitrary! **`ucbAge`** will work in Python and R, and **`ucb_age`** will work in JavaScript. (EXCEPTION: **`ucb.age`** will break in Python and JavaScript.)

It's like AP style or Chicago style — you have your preferences, but you follow whatever the person in charge wants you to use.

You should follow the code style of the organization.

RECAP!

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

```
# Python reserved keywords
# you can't use these words for variables
False None True __peg_parser__ and as assert async
await break class continue def del elif else except
finally for from global if import in is lambda
nonlocal not or pass raise return try while with
yield
```

Variables and operators

What is a variable?

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

Naming conventions

All-capped variables denote constants, or variables that don't change (it's a style across programming languages).

Variables that start with an uppercase letter denote a class — don't use them for now.

For now, start each variable with a lowercase letter.

Let’s practice naming some Python variables

https://pollev.com/soooh

Variable types

`int`, `float`

`bool`

`str`

`type()`

BREAK

The numbers you've been using are **typed** as either `int` or `float`.

```
# int
18
25
-29
67
-59
-92
33
```

```
# float
5.1
-39.784843
-92.833
-93.8432
27.54
-23.4
392.0
```

Variable types

int, float

bool

str

type()

BREAK

A `bool` only has two values: `True` or `False`

```
persian_cat_fluffy = True
sphynx_cat_fluffy = False
```

(📸 Pexels 1, 2)

Variable types

`int`, `float`

`bool`

`str`

`type()`

BREAK

A string or `str` variable is, basically, text.

A “string” of characters.

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
first_name = 'Soo'
last_name = 'Oh'
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
first_name = 'Soo'
last_name = 'Oh'
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
first_name = 'Soo'
last_name = 'Oh'

# this is the same thing
first_name = "Soo"
last_name = "Oh"
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
dog_name_single = 'Ara'
dog_name_double = "Ara"

dog_name_single == dog_name_double
Out[]:
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
dog_name_single = 'Ara'
dog_name_double = "Ara"

dog_name_single == dog_name_double
Out[]:
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
dog_name_single = 'Ara'
dog_name_double = "Ara"

dog_name_single == dog_name_double
Out[]: True
```

Variable types

`int`, `float`

`bool`

`str`

`type()`

BREAK

A string or `str` variable is, basically, text.

```
neighbor1_dog_name = 'Carnito'
neighbor2_dog_name = 'Dollar'

neighbor1_dog_name < neighbor2_dog_name
Out[]:
```

Variable types

`int`, `float`

`bool`

`str`

`type()`

BREAK

A string or `str` variable is, basically, text.

```
neighbor1_dog_name = 'Carnito'
neighbor2_dog_name = 'Dollar'

neighbor1_dog_name < neighbor2_dog_name
Out[]: True
```

Variable types

`int`, `float`

`bool`

`str`

`type()`

BREAK

A string or `str` variable is, basically, text.

```
neighbor1_dog_name = 'Carnito'
neighbor2_dog_name = 'Dollar'

neighbor1_dog_name < neighbor2_dog_name
Out[]: True

# Carnito comes before Dollar when you alphabetize
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
'carnito' == 'Carnito'
Out[]:
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
'carnito' == 'Carnito'
Out[]: False
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
'carnito' == 'Carnito'
Out[]: False

'Dollar' < 'carnito'
Out[]:
```

Variable types

int, float

bool

str

type()

BREAK

A string or `str` variable is, basically, text.

```
'carnito' == 'Carnito'
Out[]: False

'Dollar' < 'carnito'
Out[]: True
```

Variable types

int, float

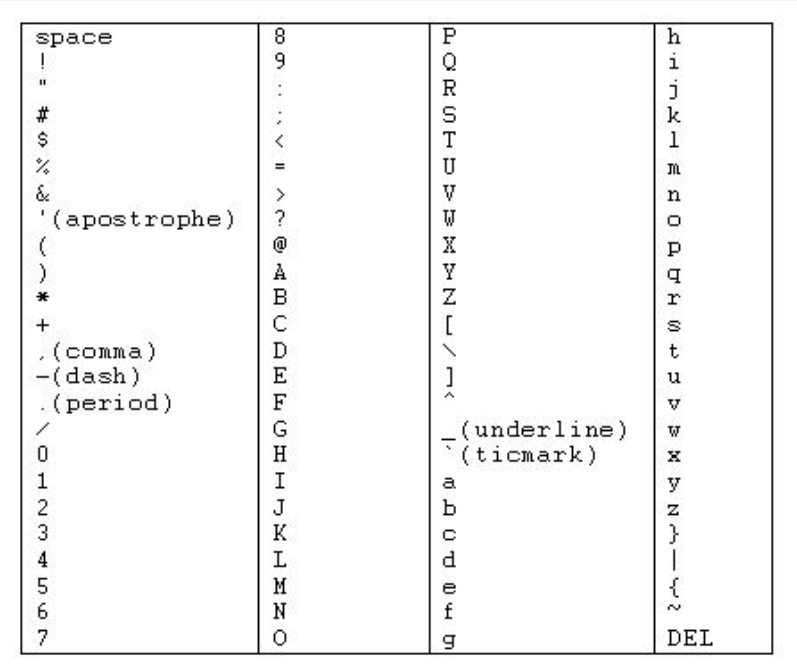
bool

str

type()

BREAK

Sort order of characters



space	8	P	h
!	9	Q	i
"	:	R	j
#	;	S	k
$	<	T	l
%	=	U	m
&	>	V	n
'(apostrophe)	?	W	o
(	@	X	p
)	A	Y	q
*	B	Z	r
+	C	[	s
,(comma)	D	\	t
-(dash)	E	]	u
.(period)	F	^	v
/	G	_(underline)	w
0	H	`(ticmark)	x
1	I	a	y
2	J	b	z
3	K	c	}
4	L	d	|
5	M	e	{
6	N	f	~
7	O	g	DEL

(ecisolutions.com)

Variable types

int, float

bool

str

type()

BREAK

You can “add” strings

```
# Using the + symbol
name = input("What is your name? ")
age = input("What is your age? ")
print(name + " is " + age + " years old.")
```

Variable types

`int`, `float`

`bool`

`str`

`type()`

BREAK

Use `type()` to get the variable's type.

That's a built-in function in Python, like `input()` and `print()` from the previous lecture.

Variable types

int, float

bool

str

type()

BREAK

```
dog_name = 'Ara'
dog_age = 4.25
dog_good = True

# What will the result be?
type(dog_name)
Out[]:

type(dog_age)
Out[]:

type(dog_good)
Out[]:
```

https://pollev.com/soooh

Variable types

int, float

bool

str

type()

BREAK

```
dog_name = 'Ara'
dog_age = 4.25
dog_good = True

# What will the result be?
type(dog_name)
Out[]: str

type(dog_age)
Out[]: float

type(dog_good)
Out[]: bool
```

Functions

Defining a function

Built-in functions

A function is a reusable block of code that performs an action. It only runs when it's called.

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average
```

Play in Jupyter notebook

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = n1 + n2 / 2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):   # parameters
    return (n1 + n2)/2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):   # parameters
    return (n1 + n2)/2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):   # parameters
    return (n1 + n2)/2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):    # parameters
    return (n1 + n2)/2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):   # parameters
    return (n1 + n2)/2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):    # parameters
    return (n1 + n2)/2
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):   # parameters
    return (n1 + n2)/2

# call the function
average(2, 6) # arguments
```

Functions

Defining a function

Built-in functions

```
# Let's create a function that
# takes two numbers and returns
# the average

# average = (n1 + n2) / 2
# What's the average of 2 and 6?
# 4

def average(n1, n2):   # parameters
    return (n1 + n2)/2

# call the function
average(2, 6) # arguments
Out[]: 4
```

Functions

Defining a function

Built-in functions

```
# type()
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]:
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]: 3
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]: 3

int(5.7)
Out[]:
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]: 3

int(5.7)
Out[]: 5

int(-2.8)
Out[]:
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]: 3

int(5.7)
Out[]: 5

int(-2.8)
Out[]: -2
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]: 3

int(5.7)
Out[]: 5

int(-2.8)
Out[]: -2

int('does this work?')
Out[]:
```

Functions

Defining a function

Built-in functions

```
# type()

# int()
int(3.0)
Out[]: 3

int(5.7)
Out[]: 5

int(-2.8)
Out[]: -2

int('does this work?')
Out[]: Error
```

Functions

Defining a function

Built-in functions

```
# float()
float(4)
Out[]:
```

Functions

Defining a function

Built-in functions

```
# float()
float(4)
Out[]: 4.0

float('how about this sentence?')
Out[]:
```

Functions

Defining a function

Built-in functions

```
# float()
float(4)
Out[]: 4.0

float('how about this sentence?')
Out[]: Error

# len()
len('chars of this sentence')
Out[]:
```

Functions

Defining a function

Built-in functions

```
# float()
float(4)
Out[]: 4.0

float('how about this sentence?')
Out[]: Error

# len()
len('chars of this sentence')
Out[]: 22

len(127)
Out[]:
```

Functions

Defining a function

Built-in functions

```
# float()
float(4)
Out[]: 4.0

float('how about this sentence?')
Out[]: Error

# len()
len('chars of this sentence')
Out[]: 22

len(127)
Out[]: Error
```

Homework

https://journ233.github.io