Welcome to The Carpentries Python iSchool Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

All content is publicly available under the Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/

Zoom Meeting for this workshop:
https://ucla.zoom.us/j/95416512948?pwd=VXZrTDdIR2llRXAvN3ZqTUVJVThwUT09

Meeting ID: 954 1651 2948
Passcode: 266335

Workshop website: https://ucla-data-archive.github.io/2021-01-08-UCLA-iSchool/

Install help needed?
Please join breakout room 1

https://www.anaconda.com/products/individual
Scroll to bottom, download 64 bit graphical install for your platform (Windows, MacOS, Linxu)

Sign in:

Name / email / Affiliation / Pronoun / Social media / Operating System (OS) / FAVORITE TV SHOW
Kristian Allen / kallen2@library.ucla.edu / UCLA Library / he-him / MacOS / Rewatching Mad Men
Ryan Gan / rgan@elcamino.edu / El Camino College & Illinois iSchool / he-his-him / Windows / Right now: Letterkenny and What We Do in the Shadows
Lisa McAulay / emcaulay@library.ucla.edu / UCLA Library / she / MacOS / Call the Midwife
Scott Peterson / speterso@berkeley.edu / UC Berkeley Library /he: him: his /Windows /Justified
Tim Dennis / timdennis@ucla.edu / UCLA Lib. Data Sci Center / he:him / mac & linux / the queen right now
Geno Sanchez / genosanchez@library.ucla.edu / UCLA Library / he:him / MacOS / Halt and Catch Fire
Jamie Jamison / jamison@library.ucla.edu / UCLA Data Science Center / she:her /
Ki Kim / khk002@ucsd.edu / UCSD Library IT / he-him / MacOS/Windows / GOT
peggy cabrera / peggy.cabrera@sjsu.edu / loved "Pose"
Leigh Phan / leighphan@ucla.edu / UCLA Library Data Science Center/ she-her / Mac
Lisa Kahn / lisakahn36@gmail.com / UCLA MLIS Student / she|her / Mac / Buffy (except Whedon) currently on Hulu
Richard Leuchter/rleuchter@mednet.ucla.edu/UCLA Health/he-his/Windows/John Wick
Bailey Berry/ baileyber17@gmail.com / UCLA MLIS / she / her/ Windows/ re-watching 30 Rock

Syann/ syannlunsford@gmail.com/ @syanobacteria/ UCLA MLIS/ they / Windows/ Pushing Daisies on HBO!!
Ashley Peterson / ashleypeterson@library.ucla.edu / she series / mac / always evangelizing for Lodge 49
Josh Belmont / joshua.belmont.2023@anderson.ucla.edu / UCLA Anderson / he-him / Windows / Queen's Gambit
Kirk Wang / k2wang@ucsd.edu / k2wang@ucsd.edu / he / Windows 10 / Dissenchantment (Netflix)
Yen Tran / yen.tran@sjsu.edu / Ssan Jose State University / she-her / mac /
Anna Isorena / aisorena@ucsd.edu / UC San Diego Library / she-her / Windows / Schitts Creek (Netflix)
Win Shih / winyuans@usc.edu / USC Libraries / he him / Windows 10 / Modern Love
Chelsea Kern / ckern@ucla.edu/ UCLA English / she/her/hers / Mac / Recently: The Crown
Patricia L. Villon / pvillon@g.ucla.edu / UCLA MLIS / she/her / macOS / hm, I'm rewatching Twin Peaks right now
Hannah Lien / hannahlien@g.ucla.edu / UCLA MLIS / she/her / Windows / Xena: Warrior Princess (sadly I don't think it's on anything)
Elizabeth Wood / emwood@ucla.edu / UCLA MLIS / she/her / macOS / Great British Bake Off!
Jordan Wood / jwood@bto.ucla.edu / UCLA BTO / she/her / Windows / Parks & Rec
Noelle Leong / nkleong@g.ucla.edu / UCLA MLIS / she-her / macOS / Insecure
Julia Tanenbaum julia.tanenbaum@gmail.com MLIS she/her bake off
Jessica Craig / jessicaecraig@g.ucla.edu / MLIS / she,her,hers / Windows / currently watching Search Party
Monica Vaughn/mvaughn@jsei.ucla.edu/Stein/She/GreatBritishBakeOff
Samantha Stevens / sas109@ucsd.edu / / UCSD / she/her / macOS
Susan Oh / soh@saonet.ucla.edu / UCLA / she/her / macOS / Finding Your Roots (PBS)
Jade Levandofsky/ levandoj19@gmail.com / UCLA MLIS/ They-Them / Windows / Ugly Betty
Julianne Wagner / jwagner15@ucla.edu / UCLA MLIS / she/her / macOS / Idk if it's my favorite ever, but I am watching Itaewon Class rn
Erica Zhang/ ezhang20@library.ucla.edu / UCLA Library / she-her / Windows / Brooklyn 99
Kiumars Edalati/ kedalati0@ucla.edu/UCLA/he him/ windows
Leia Ku Cheng Yee / kcychengyee@gmail.com/ she,her/ MacOs/Shitts Creek
Kelly Aldana/ kellyaldana11@ucla.edu/ she-her/ MacOs/ Grey's Anatomy
Meng Li / mengli2@usc.edu / USC Libraries / she-her / MacOS
Victoria Maglonzo / maglonzo@usc.edu / USC MMLIS / she - her / MacOS / Modern Love
Heather Kirkpatrick/ UCLA EPSS/ hkirkpatrick@ucla.edu/ she-her/ Windows/ The Good Place, Schitt's Creek
Jinhong Qi/jqi@ucsd.edu/UCSD Library/she-her/Windows/
Evelyn Arias/earias@mednet.ucla.edu / she | her / Windows / TV: Peaky Blinders
Kelly Aldana/ kellyaldana11@ucla.edu/ she-her/ MacOs/ Grey's Anatomy

(so exciting reading all these TV shows! i want to run away and start binging)

Finding where to stream stuff - https://www.justwatch.com/
UCLA access to film on kanopy access: https://guides.library.ucla.edu/videocollections/kanopy

Lessons sourced from:
Lesson: https://librarycarpentry.org/lc-python-intro/

Schedule: Library Carpentry - Python

- **9:00 - Introductions**
- **9:15 - Jupyter notebooks**
  - http://swcarpentry.github.io/python-novice-gapminder/01-run-quit/index.html
- **9:30 - Variables and types**
  - https://librarycarpentry.org/lc-python-intro/02-variables/index.html
- **10:00 - Break / Challenge**
- **10:20 - Libraries and built in functions (Tim up to here)**
  - https://librarycarpentry.org/lc-python-intro/04-built-in/index.html
  - https://librarycarpentry.org/lc-python-intro/06-libraries/index.html
- **10:45 - Break / Challenge**
- **11:00 - Lists and loops**
  - https://librarycarpentry.org/lc-python-intro/11-lists/index.html
  - https://librarycarpentry.org/lc-python-intro/12-for-loops/index.html
- **11:30 - Break / Challenge**
- **11:45 - Conditionals**
  - https://librarycarpentry.org/lc-python-intro/17-conditionals/index.html
- **12:15 - Break / Challenge**
- **12:30 - Review / Questions**
- 12:45 - End / Feedback

Class Notes

Help us take notes!

Use variables to store values.

- Variables are names for values.
- In Python the = symbol assigns the value on the right to the name on the left.
- The variable is created when a value is assigned to it.
- Here, Python assigns an age to a variable age and a name in quotation marks to a variable first_name.

General questions & notes (about Jupyter Notebook, Python, etc.)

- is this like openrefine where it doesn't rely on an internet connection, it just uses the browser as an interface?
  - Yes, Jupyter Notebook runs locally on your computer. indicated by the the "localhost:8888/tree/Desktop" address in the browser.

- API = application programming interface. This is a way to connect to another system with a program
- How did you get the gray numbers in front of the commands?
    - View> Toggle Line Numbers
- In python you can use single or double quotes without an error, but the community convention is to use single quotes
- What is the windows keyboard shortcut for running the code?
    - Shift-Enter
- what do you hit to run the command?
    - Press the Play button at top of Jupyter Notebook window, or Shift-Enter
- Python counts letters as A-Z, then a-z.

```
age = 42
first_name = 'Ahmed'

print(first_name, 'is', age, 'years old')
Ahmed is 42 years old
```

Variables must be created before they are used.

```
print(eye_color)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-c1fbb4e96102> in <module>()
----> 1 print(eye_color)

NameError: name 'eye_color' is not defined
```

Variables can be used in calculations.

```
age = age + 3
print('Age in three years:', age)
Age in three years: 45
```

Use an index to get a single character from a string.

```
element = 'helium'
print(element[0])
h
```

- The characters (individual letters, numbers, and so on) in a string are ordered. For example, the string ‘AB’ is not the same as ‘BA’. Because of this ordering, we can treat the string as a list of

characters.
- Each position in the string (first, second, etc.) is given a number. This number is called an index or sometimes a subscript.
- Indices are numbered from 0 rather than 1.
- Use the position's index in square brackets to get the character at that position.

Use a slice to get a substring.

- A part of a string is called a substring. A substring can be as short as a single character.

```
element = 'sodium'
print(element[0:3])
```

```
sod
```

Use the built-in function len to find the length of a string.

```
print(len('helium'))
```

```
6
```

Python is case-sensitive.

Use meaningful variable names!

- Python thinks that upper- and lower-case letters are different, so **Name** and **name** are different variables.

ex:
Name = 'Ahmed'
name = 'Stacy'

Breakout Session #1 - Exercises 1.1 - 1.3 (10 min)

Types of Data in Python

- string ** ''
- int (integer) - whole numbers
- float (short for 'floating point') - decimal (e.g. 1.0, 3.8, 3.5) numbers

```
type(52)
```

```
int
```

```
title = 'biochem'
type(title)
str
```

```
print(5 - 3)
2
```

```
print('hello' - 'h')
---------------------------------------------------------------------------TypeError
Traceback (most recent call last)<ipython-input-5-35e8597b28d6> in <module>----
> 1 print('hello' - 'h')
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

```
full_name = 'Ahmed' + ' ' + 'Walsh'
print(full_name)
Ahmed Walsh
```

```
separator = '=' * 10
print(separator)
==========
```

Must convert numbers to strings or vice versa when operating on them.

- Cannot add numbers and strings.

```
print(1 + 'A')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-fe4f54a023c6> in <module>()
----> 1 print(1 + '2')

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

- Not allowed because it’s ambiguous: should 1 + '2' be 3 or '12'?

```
print(1 + int('2'))
3
```

```
print(str(1) + '2')
12
```

- Some types can be converted to other types by using the type name as a function.

```
type('12345')
str

int('12345')
12345
```

Comments

You can also comments in your code by using the '#' symbol. Lines that are preceded by # will not run as code. This is handy to leave a clarifying note for yourself to refer to later.

Python comes with 'Standard Libraries' which are functions that are 'built-in' or included in Python.

Some examples:

- len( )
- type( )
- print( )

Commonly-used built-in functions include max, min, and round.

```
print(max(1, 2, 3))
print(min('a', 'b', 'c'))
print(min('a', 'A'))
3
a
A
```

- 'a' is smaller than 'b'
- 'A' is smaller than 'a'
- '0' is smaller than 'a'

```
round(3.712)
4
```

```
round(3.712, 1)
3.7
```

- We can specify the number of decimal places we want.

Use the built-in function help to get help for a function.

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.  Otherwise
    the return value has the same type as the number.  ndigits may be negative.
```

Libraries

PIP: tool for installing Python packages.
e.g.

```
pip install pandas
```

A program must import a library module before using it.

- Use import to load a library module into a program’s memory.
- Then refer to things from the module as module_name.thing_name.
    - Python uses . to mean “part of”.
- Using string, one of the modules in the standard library:

```
import string

print('The lower ascii letters are', string.ascii_lowercase)
print(string.capwords('capitalise this sentence please.'))
The lower ascii letters are abcdefghijklmnopqrstuvwxyz
Capitalise This Sentence Please.
```

Create an alias for a library module when importing it to shorten programs

- Use import ... as ... to give a library a short *alias* while importing it.
- Then refer to items in the library using that shortened name

```
import string as s

print(s.capwords('capitalise this sentence again please.'))
Capitalise This Sentence Again Please.
```

For more information run:

```
help(string)
```

Import specific items from a library module to shorten programs.

- Use from ... import ... to load only specific items from a library module.
- Then refer to them directly without library name as prefix.

```
from string import ascii_letters

print('The ASCII letters are', ascii_letters)
The ASCII letters are
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Create an alias for a library module when importing it to shorten programs.

- Use import ... as ... to give a library a short *alias* while importing it.
- Then refer to items in the library using that shortened name.

```
import string as s

print(s.capwords('capitalize these words'))
Capitalize These Words
```

Breakout Session #2 - Exercises 2.1 - 2.6

Lists

```
car_model = 'dodge'

car_model
dodge
```

```
car_model2 = 'nissan'
car_models = ['dodge', 'nissan']
```

- above is a variable called 'car_models', which is a list with the values 'dodge' and 'nissan'

```
car_models
['dodge', 'nissan']


car_models.append('bmw')

car_models
['dodge', 'nissan', 'bmw']
```

Let's create a list of temperatures:

```
temps = 16.5, 17.5, 18, 17.5]

print('the temperatures we collected are:', temps)
```

- the temperatures we collected are: [temps = 16.5, 17.5, 18, 17.5]

```
print('the first temp of the day is:', temps[0])
```

- the first temp of the day is: 16.5

```
temps.append(2)

temps
[16.5, 17.5, 18, 17.5, 20]
```

Deleting items from a list:

```
del temps[4]
```

- since we want to delete '20' we tell Python to delete the 4th item on the list (since Python starts counting at zero)

```
print(temps)
```

- [16.5, 17.5, 18, 17.5]

Add an item to a list:

```
temps.append(19.5)
```

Below, we tell Python to print the item in the list up to, but not including, the 2nd item:

```
print(temp[:2])
[16.5, 17.5]

print(temps[::])
```

```
temps.append([50, 60, 70])
temps
[16.5, 17.5, 18, 17.5, 19.5, [50,06,70]]

temps.insert(4, 100)

temps = [101, 16.5, 17.5, 18, 17.5, 7, 6, 57]
```

sorted() will sort the list

```
my_sorted_temps = sorted(temps)

print('the temperatures we collected are:', temps)
the temperatures we collected are: [101, 16.5, 17.5, 18, 17.5, 7, 6, 57]
```

## For Loops

- "A loop is something that repeats over and over again until there is nothing left to do
- I think of a "for" loop as a loop that means "**for each** blah **do** blech" -- Lisa McAulay

Format of a For loop:
for eachofthing in collection:
  do something

  Everything indented under the first line and after the ' : ' is included in the loop

```
for number in [2, 3, 5]:
    print(number)
```

number above is called a loop variable, only used within for loop

```
for temp in my_sorted_temps:
```

- `print('the temp is', temp)`

```
the temp is 6
the temp is 7
the temp is 16.5
the temp is 17.5
the temp is 17.5
the temp is 18
the temp is 57
the temp is 101

primes = [2,3,5]
for prime in  primes:
```

- squared

```
range(startnum, stopnum)
```

range(0, 3)

```
for num in range(0, 3):
  print(num)
```

will print 0,1,2

print(range(0,3))

for number in range(0,3):

- print(number)

for number in range(2,5):

- print(temps[number])
- 17.5
- 18
- 17.5

Review: to create a list:

- create a variable
- set it equal to some items, enclosed by []
- the list can even be empty

example:

```
my_new_list = []
my_new_list.append('kristian')
my_new_list.append('tim')
my_new_list[0]
```

- 'kristian'

my_new_list[-1]

- 'tim'

Exercises:

General Instructions:

1. Introduce yourself to your break room mates!
2. Choose one person to share their screen and work through the problems together.
3. You can always run the code in your notebook, but try and predict results before you do so.

Challenges/Break 1 (10 min):

1.1. Swapping Values

Draw a table showing the values of the variables in this program after each statement is executed. In simple terms, what do the last three lines of this program do?

- x = 1.0
- y = 3.0
- swap = x
- x = y
- y = swap
- swap = x # x->1.0 y->3.0 swap->1.0
- x = y   # x->3.0 y->3.0 swap->1.0
- y = swap # x->3.0 y->1.0 swap->1.0

1.2. Predicting Values

What is the final value of position in the program below? (Try to predict the value without running the program, then check your prediction.)

```
initial = "left"
position = initial
initial = "right"

initial = "left"  # Initial is assigned the string "left"
position = initial  # Position is assigned the variable initial, currently "left"
initial = "right"  # Initial is assigned the string "right"
print(position)
```

```
left
```

1.3. Slicing

What does the following program print? (Run the code!)

```
library_name = 'social sciences'
print('library_name[1:3] is:', library_name[1:3])
```

1. What does thing [0:4] do?
    1. slice b/t 0 and 4
    2. It will slice the string, starting at the low index and ending an element before the high index
2. What does thing[low:] (without a value after the colon) do?
3. What does thing[:high] (without a value before the colon) do?
4. What does thing[:] (just a colon) do?
5. What does thing[number:negative-number] do?

```
library_name[1:3] is: oc
```

1. It will slice the string, starting at the low index and ending an element before the high index
2. It will slice the string, starting at the low index and stopping at the end of the string
3. It will slice the string, starting at the beginning on the string, and ending an element before the high index
4. It will print the entire string
5. It will slice the string, starting the number index, and ending a distance of the absolute value of negative-number elements from the end of the string

Stop here:

Challenges/Break 2 (10 min):

2.1 Fractions

What type of value is 3.4? How can you find out?

Answer:
It is a floating-point number (often abbreviated “float”).

```
print(type(3.4))
<class 'float'>
```

2.2 Automatic Type Conversion

What type of value is 3.25 + 4?

Answer:
It is a float: integers are automatically converted to floats as necessary.

```
result = 3.25 + 4
print(result, 'is', type(result))
7.25 is <class 'float'>
```

2.3 Spot the Difference

1. Predict what each of the print statements in the program below will print.
2. Does max(len(rich), poor) run or produce an error message? If it runs, does its result make any sense?

```
rich = "gold"
```

```
poor = "tin"
print(max(rich, poor))
print(max(len(rich), len(poor)))
```
  the length of the value 'rich' is 4 and the length of the value 'poor' is 3. The maximum of the two values is 4.

Answer:

```
tin
4
TypeError: '>' not supported between instances of 'str' and 'int'
```

2.4 Exploring the os Library

The os library provides a way of accessing operating system functionality.

1. What function from the os library can you use to determine the current working directory?

Answer:
2.4
Using help(os) we see that we've got os.getcwd() which returns a string representing the current working directory.

2.5 Locating the Right Module

Given the variables year, month and day, how would you generate a date in the standard iso format:

```
year = 2016
month = 10
day = 22
```

1. Which standard library https://docs.python.org/3/library/ module could help you?
2. Which function would you select from that module?
3. Try to write a program that uses the function.

The datetime module seems like it could help you. https://docs.python.org/3/library/datetime.html

You could use date(year, month, date).isoformat() to convert your date:

```
import datetime

iso_date = datetime.date(year, month, day).isoformat()
print(iso_date)
```
or more compactly:

```
import datetime

print(datetime.date(year, month, day).isoformat())
```
Answer:

2.6 When Is Help Available?

When a colleague of yours types help(os), Python reports an error:

```
NameError: name 'os' is not defined
```

What has your colleague forgotten to do?

Answer:
2.6
Importing the os module (import os)

We are working on For Loops
Here are some notes

A loop is a piece of programming that moves through the same set of actions over and over again (in a "loop", hence the name)

Loops exist in most programming languages and even in the shell; see the Library Carpentry Lesson on Bash for a different example

Challenges/Breakout Session 3:

# Create a list with the names of 5 people you know
# Create a new list with the contents of your people list, but organized alphabetically
# Print out a greeting to the first and last person in your sorted list

Example - Let's created a list called 'friends':

```
friends = ['Kiumars', 'Elizabeth', 'Leigh', 'Lisa','Cory']
```

```
sorted(friends)
```

- ['Cory', 'Elizabeth', 'Kiumars', 'Leigh', 'Lisa']

We told Python to sort the list called 'friends' and now we're going to rename the list 'sorted friends'
```
sorted_friends = sorted(friends)
```

Let's say we want to print 'Hi, Cory!'
```
print('Hi, ' + sorted_friends[0] + '!')
```

- Hi, Cory!

If we want to print 'What is for lunch, Lisa?'
```
print('What is for lunch, ' + sorted_friends[4] + '?')
What is for lunch, Lisa?
```

Another example:

```
names = ['jamie', 'kristian', 'lisa', 'tim'

for name in range(0,4):
```

- print(names[name])

```
print('hello ' + names[0] + ' and ' + names[-1])
```

- hello jamie and tim

```
masses = [3.54, 2.07, 9.22, 1.86, 1.71]
masses
[3.54, 2.07, 9.22, 1.86, 1.71]

for mass in masses:
```

- print(mass)

```
3.54
2.07
9.22
1.86
1.7
```

Creating if...else statements

```
for mass in masses:
    if mass > 3.0:
        print(mass, 'is pretty large')
3.54 is pretty large
9.22 is pretty large
```

let's add the 'else' portion

```
for mass in masses:
    if mass > 3.0:
        print(mass, 'is pretty large')
```

- else:
    - print(mass, 'is not greater than 3.0')

```
for mass in masses:
    if mass > 3.0:
        print(mass, 'is pretty large')
    elif mass = 1.86:
        print(mass, 'this is the greatest mass ever!!!')
    else:
        print(mass, 'no criteria met')
```

elif stands for 'else if'

If (then) else are also a type of logic that exists in most programming languages
The syntax varies, but you see the same concept in bash, Java, python, etc

Challenges/Break 4:

# Run the following code, an error should occur, identify and correct the error
# Why does this error occur?

```
seasons = ['Spring', 'Summer', 'Fall', 'Winter']
print('My favorite season is ', seasons[4])
```

# Fill in the blanks below, to create a reversed string (tin becomes nit)
\#original = "tin"

```
#result = ____
#for char in original:
#    result = ____
#print(result)
```

Please leave us Feedback for today!

https://forms.gle/axsUDgYi74Lpqh88A