Discussion 1

DSC 20, Winter 2024

Welcome to DSC 20! 

Agenda

- How discussion will work this quarter
- Skill Tests
- **Content**
    - basic operators
    - basic data types
    - type casting
    - booleans
    - return vs print keyword
    - short-circuits
    - if conditions
- Discussion Questions
- File Structure
- Style
- Doctests
- Skill Test Demo
- Class Advice (time permitting)

About Me

Benjamin (call me Ben)

- Originally from the Bay Area (San Jose/Fremont)
- Fourth year Data Science major
- 7th(?) quarter teaching DSC 20
- Outside the classroom 👨‍🏫: track & field, badminton, marching band, reading, video games, anime, cooking, eating, travelling, bracelet making, video/photo editing, etc.

How discussion will work this quarter

Discussions (3%)

The purpose of discussion is to supplement course content and offer practice for taking physical exams.

Process:

- Along with regular content, you will be given a few problems emulating exam questions and a limited time.
- After the time is up, staff will go over the solutions.
- In order to record your participation, you will respond to questions on webclicker similarly to lecture participation, so please bring your phones.

Notes:

- 1 discussion attendance will be dropped.
- If your exam score (midterm or/and Final) is above 90%, then 4 more discussions will be dropped. For example, if you do well on both the midterm and the final, you will only be held liable for 1 discussion (1 default skip + 4 midterm skip + 4 final skip).

- after reinforcing content, discussions will feature 4 practice questions on physical worksheets.
- attendance is taken using webclicker; they will be feedback regarding the practice questions.
- webclicker is geo-locked; you have to be in/near the room to get points

Skill Tests

Skill Test (10%)

Rather than a 2 midterm class model, we will be replacing 1 midterm with a set of short skill tests administered by tutors. Starting week 2, there will be 5 skill tests administered on a weekly basis.

- Each skill test will be worth 2% of the category and test a specific topic, ensuring basic understanding.
- Each test will be accompanied with a set of instructions to carry out and will be time constrained.

Refer to EdStem for more detail.

Week	Topic
Week 2	Terminal
Week 3	Loops
Week 4	Dictionaries
Week 5	Files
Week 6	Recursion

- skill tests will be held remotely (more detail coming soon)

Content

Basic Operations

+ - Numerical addition / concatenation operator

- - Numerical Subtraction operator

/ - Classic Division operator

// - Floor Division operator

***** - Numerical Multiplication / repetition operator

****** - Numerical Exponential operator

% - Numeric remainder operator

Basic Data Types

String - Data type for text

Int - Data type for whole numbers

Float - Data type for non-integers

Bool - True or False

note: **None** is technically its own type

In [8]:
```
print(f'this is a string: {"abc"}')
print(f'this is an int: {1}')
print(f'this is a float: {1.5}')
print(f'this is a bool: {True}')
```

```
this is a string: abc
this is an int: 1
this is a float: 1.5
this is a bool: True
```

Type Casting in Python

- refers to converting a variable from one type to another
- **Casting can fail** if the conversion is not logically possible
- Some automatic conversions happen ex.converting integers to floats in math operations

In [18]:
```
a=1
a
```

Out[18]:
```
1
```

In [19]:
```
str(a)
```

Out[19]:
```
'1'
```

In [20]:
```
float(a)
```

Out[20]:
```
1.0
```

In [21]:
```
bool(a)
```

Out[21]:
```
True
```

Checkpoint

What is the type of the result from the resulting expression?

In [7]:
```
def fractional_addition(x,y):
    return float(x) + y

fractional_addition(4, 5)
```

Checkpoint Solution

In [8]:
```
fractional_addition(4, 5)
```

Out[8]:
```
9.0
```

Boolean Operators

Logical Operators (in order priority):

- **not** - reverses the outcome of the following expression
- **and** - all expressions compared with “and” must be True to evaluate True
- **or** - at least one expression compared with “or” must be True to evaluate True

Comparison Operators (generates booleans):

- **==** - equality check
- **!=** - inequality check
- **>, >=, <, <=** directional check

note: order of evaluation is overriden by parentheses (just like PEMDAS)

Checkpoint

What is the result of the following expression?

In [11]:
```
(5 > 2) and not (10 == 0) and not (4 != 4)
```

Checkpoint Solution

In [13]:
```
(5 > 2) and not (10 == 0) and not (4 != 4)
```

Out[13]:
```
True
```

return vs print

'return' is a very important and special keyword in python

- It's python's functional way to pass a value out of a function
- not every function needs a return (but many have one)

'print' displays some output onto a console

- commonly used for debugging, sometimes integral to the purpose of a function.
- print's result is None (it doesn't "return" a meaningful value)

In [6]:
```
def returning(x):
    return x
def printing(x):
    print(x)

a = returning('marina')
a
```

Out[6]:
```
'marina'
```

In [7]:
```
b = printing('marina')
```

```
marina
```

In [21]:
```
print(f'the output of a is: {a}')
print(f'the output of b is: {b}')
```

```
the output of a is: marina
the output of b is: None
```

Short-circuits

- In python, expressions are evaluated from left to right
- With certain operations, a "short-ciruit" can happen if conditions are met
- Certain functions within python also act as short-circuits -> return is a short circuit

In [23]:
```
# expression short circuited before an error
True or 1/0
```

Out[23]:
```
True
```

In [14]:
```
# and isn't a short circuit
True and 1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent
call last)
Cell In[14], line 2
      1 # and isn't a short circuit
----> 2 True and 1/0

ZeroDivisionError: division by zero
```

In [26]:
```
def shortcircuit(x):
    return x
    x/0
def broken(x):
    x/0
    return x
```

In [27]:
```
shortcircuit(5)
```

Out[27]:
```
5
```

In [28]:
```
broken(5)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent
call last)
Cell In[28], line 1
----> 1 broken(5)

Cell In[26], line 5, in broken(x)
      4 def broken(x):
----> 5     x/0
      6     return x

ZeroDivisionError: division by zero
```

Conditional Statements

if (boolean expression):

    //Do stuff

elif (other boolean expression):

note: elif is optional, can have as many elifs as necessary

    //Do other stuff

else:

note: else is optional, will execute only if the conditions in the “if” and “elif” statements are not true. "else" is the final part of a chained conditional statement.

    //Do other other stuff

A few practice problems

Question 1

Given the following code, what is the result of the function call?

In [15]:
```
def math(x, a):
    return x - x ** a

math(4,0.5) # <--- that one
```

Out[15]:
```
2.0
```

A) 0.0

B) 2.0

C) 4.0

D) 6.0

Question 2

Given the following code, what is the result of the boolean expression?

In [16]:
```
(True and True) or not 0 and 1/0 or True and not ''
```

Out[16]:
```
True
```

A) True

B) False

C) Error

D) Nothing

Question 3

Given the following code, what numbers are displayed?

In [20]:
```
def foo(x):
    print(x)
    print(x*2)
    return x*3
    print(x*4)
    return x*5

foo(2) # <--- that one
```

```
2
4
```

Out[20]: 6

A) 2

B) 2, 4, 6

C) 2, 4, 6, 8

D) 2, 4, 6, 8, 10

Question 4

Given the following code, what is the result?

In [18]:
```
bool(-10)
```

Out[18]:
```
True
```

A) True

B) False

C) Error

D) Nothing

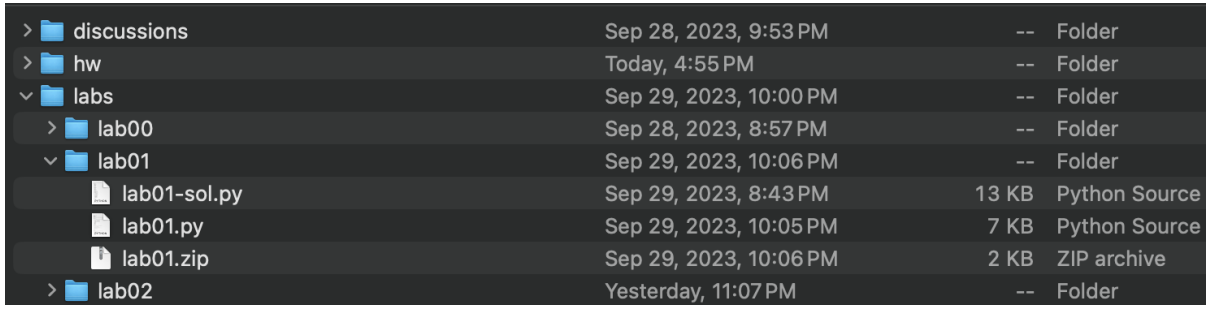
Skill Test Demo

PythonTutor - Debugging (demo)

Python Tutor

File Structure

- Staying organized is crucial to your success in both this class and later on
- Use a file structure that makes sense (we subtly enforce this)
- Enables tutors to help you faster, makes it easier for you to track things
- sample file structure:



note: make a new subfolder in your DSC20 one for skill tests :)

Style

Why enforce style? - having poor style while coding is a death sentence if you run into a bug. Having clean code makes identifying break points easy and smooths out your workflow if you don't finish in one sitting.

1. limit lines to 79 characters or less (use the ruler feature in your IDE)
2. avoid magic numbers (using a number randomly instead of identifying it) except for -1,0,1
3. avoid meaningless variable names (ex. one=1)
4. add docstrings to every function (inner, helper) you write!
5. use snake_case for names

There are more rules, this is just a (very) brief overview. The entire ruleset can be found on the course website.

Doctests

- Tests to check that your function works as intended
- denoted by the '>>> ' symbol (space included)!
- the line right after the '>>> ' represents the intended output
- well written doctests make sure your code is logically sound

to run doctests: terminal -> file location -> python -m doctest [filename].py

note: if you want to see the doctests you passed when you run them, add a '-v' at the end.

Why bother learning how to code?

- Coding is a transferrable skill (mindset, language)
- Everything cool you see is a tool, not a solution

Things I would do if I were a student taking DSC20 now

1. Utilize Resources
2. Use Online Content, don't depend on them
3. Useful Tools
4. Practice
5. Take a break!

1. Utilize Resources

- Office hours are great, not just for getting help on assignments or studying for the class, but as interpersonal resources. All the tutors are amazing and they're open to just talking as peers - they're people you can ask about future classes, career advice, etc. Some tutors will have more input than others, but don't be shy! We're still students and we were/are in your shoes.

- Discussions are intended to be particuarly helpful. Rather than just reinforcing class concepts, they will also occassionally teach side topics that will help you become a strong programmer and help you adjust to exams (paper coding).

- If you're ever struggling, don't be afraid to reach out to the staff for help. We do our best to support students.

2. Use Online Content, don’t depend on them

- Tools like ChatGPT are amazing, but abusing them will only hurt your own learning
- Copying code from medium articles or w3school can get you a quick grade, but teaches you nothing and can get you AI'd
- Asking friends for a question can get not only you AI'd, but also your friend
- This class is best traversed with class content and staff support. You can consult your peers on programming ideas, but not the implementation itself

3. Useful Tools

1. Python Tutor - Debugging

    - invaluable for debugging, amazing for understanding why some code works, and why others don't

2. w3school, geeksforgeeks, other documentation and example websites - Reference

    - Stores documentation for what functions/methods do with examples
    - Always includes examples so that you can understand what's actually happening

3. Fig - File Navigation

    - terminal plugin for mac that autocompletes file names
    - can help you navigate your files much more easily

4. Iterm2

    - mac terminal alternative; has a lot of useful features

5. oh my zsh / power level 10k

    - framework to modify your terminal (enables plugins)
    - powerlevel10k is a ohmyzsh plugin to beautify terminal
    - makes things a lot more readable!(why my terminal looks nice^)

4. Practice

- like all new skills, practice makes perfect
- staring at code and not writing any will ultimately do nothing to make you a better coder
- Websites such as coding bat has practice problems for you to familiarize yourself with python
- codecademy is something that I personally used with DSC20 to help learn coding. It walks you through the fundamental functions of python and always frames it in sample code
- more challenging questions can be found on leetcode, though I don't recommend this for people new to coding.

more can be found on course website

5. Take a break!

- parking at the desk for 3+ hours and staring at a screen never helped anyone
- remember to take breaks while coding, sometimes your brain just comes up with the solution
- I personally have a lot of solutions come to me after a shower

Thanks for coming!