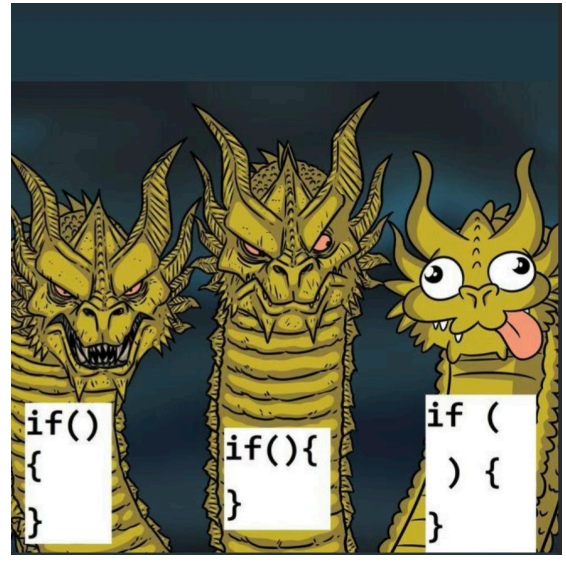
Discussion 3

DSC 20, Spring 2024

Please come up and take a worksheet (if you have an ipad, then the worksheet is already posted in the discussion tab of the calendar). Take 15-20 minutes to complete the questions to the best of your ability, we will go over answers afterwards.

Meme of the week



Agenda

- Homework Autograder
- Homework Redemption Policy
- **Content**
    - Dictionaries
    - Files
    - Text Processing
    - List Comprehension
    - Dictionary Comprehension
    - Assert Statements

General Feedback & tips: homework 1

- Make sure to save the file (control + S) after making any changes so that you're testing the most recent iteration.
- name files without spaces (use _) or else you'll regret it (trust me)
- your tab key can autocomplete file names (avoids mistypes)
- up key will pull up the last command you ran (don't need to keep typing doctests)
- Read the instructions carefully for questions and don't skip the instructions at the beginning of homework.

Homework Autograder

PSA: Make sure that your code runs against our autograder after you submit on Gradescope. The autograder has an additional function as the logic check for the code you submit; if it cannot complete running your code, you will get a 0. If your code fails our autograder, make sure to fix your code and resubmit.

Cases that may break autograder:

- fatal syntax errors
- infinite loops

Homework Redemption Policy

Homework Resubmission/Regrade Policy:

- For each homework after the grades are released, you may resubmit the homework once to correct your mistakes within a week (7 days).
- You will get 80% of your score lost back for resubmission.
- If your original submission results in autograder failure (e.g. timed out, compile error, etc.), you may additionally ask for a regrade before the resubmission (all within 7 days of grade release).
  - For the first 2 homeworks compile errors and print vs. return issues can be regraded for free.
  - After the first 2 weeks compile errors cost 10 points; print vs. return issues cannot be regraded if the write-up explicitly says return or print (use the resubmission attempt).
  - If there are issues that are out of your control, we may regrade your submission for free.
- **Important**: Please test your code locally before uploading and wait for the autograder message before you leave Gradescope.
- See Regrade Requests for other general information.

For those who want to fix some basic errors that caused a lot of errors in their homework, refer to Homework Redemption Policy on the course website for more detail

Content

Dictionaries

- Mutable storage of key, value pairs
- Can store any data type, multiple at a time
- Elements are acccessed via keys
- keys must be **hashable** and **unique**

methods

- accessing keys (as an iterable) -> dict.keys()
- accessing values (as an iterable) -> dict.values()
- accessing key,value pairs (as an iterable of tuples) -> dict.items()

note: hashablility correlates to the stability of the data - essentially, **data that can't change is hashable** (int, str, tuple, etc.) while **data that can change is not hashable (list, dictionary). Basically, it's all about mutability!**

In [2]:
```
temp = {}
temp['marina langlois'] = 1
hash('marina langlois')
```

Out[2]:
```
-2689793995949873442
```

In [3]:
```
temp[[1,2,3]] = 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent
call last)
Cell In[3], line 1
----> 1 temp[[1,2,3]] = 2

TypeError: unhashable type: 'list'
```

In [4]:
```
list(temp.keys())[0]
```

Out[4]:
```
'marina langlois'
```

In [6]:
```
temp.keys()[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent
call last)
Cell In[6], line 1
```

```
----> 1 temp.keys()[0]

TypeError: 'dict_keys' object is not subscriptable
```

Files

storage for data (think csv's from DSC10, txt's from assignments, etc.)

Access Modes

Write : 'w' -> every time the file is opened in write mode, the file is wiped. Calling file.write() will add in your data.

Append : 'a' -> file.write() will append your data to what existed in the file beforehand.

Read : 'r' -> no writing privilege, can only pull the data from the file with relevant methods.

note: If you try to open a file in write mode that doesn't exist, python will create it.

In [51]:
```
with open('files/data.txt', 'w') as f:
    f.write('Marina Langlois is the DSC20 professor\nSuraj Rampure is t
```

Text Processing

reading data:

- file.read() -> reads in all the data as a single string
- file.readline() -> reads in data line by line (has to be recalled)
- file.readlines() -> reads in all the data as a list where each line is another element of the list

After reading in the data, you can transform it however you'd like, and then rewrite it back into the file using .write() (if this is relevant).

In [52]:
```
with open('files/data.txt', 'r') as f:
    # notice to read the data, I have to open in 'r' mode first
    data = f.read()
print("the data in the file currently is: ")
print(data)
```

```
the data in the file currently is:
Marina Langlois is the DSC20 professor
Suraj Rampure is the DSC180A professor
```

In [53]:
```
with open('files/data.txt', 'w') as f:
    data = data.replace('DSC180A', 'DSC10')
    f.write(data)
print("the data in the file is now: ")
with open('files/data.txt', 'r') as f: print(f.read())
```

```
the data in the file is now:
Marina Langlois is the DSC20 professor
Suraj Rampure is the DSC10 professor
```

In [7]:
```
# how can I ingest the data without the column names?
with open('files/names.csv', 'r') as f:
    print(f.read())
```

```
First Name,Last Name
Emily,Wilson
Jennifer,Sanchez
Denise,Ramos
Manuel,Barnett
Ashley,Hampton
```

In [8]:
```
with open('files/names.csv', 'r') as f:
    f.readline() # use readline to skip the first line
    print(f.read())
```

```
Emily,Wilson
Jennifer,Sanchez
Denise,Ramos
Manuel,Barnett
Ashley,Hampton
```

Checkpoint

What happens when I try to open a file that doesn't exist in read mode?

A. Python will create the file

B. Nothing

C. Error

D. idk

Checkpoint Solution

In [20]:
```
with open('files/nonexistent.txt', 'w') as f:
    pass
```

In [19]:
```
with open('files/nonexistent.txt', 'r') as f:
    pass
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[19], line 1
----> 1 with open('files/nonexistent.txt', 'r') as f:
      2     pass

File /opt/anaconda3/envs/tutor/lib/python3.10/site-packages/IPython/core/interactiveshell.py:284, in _modified_open(file, *args, **kwargs)
    277 if file in {0, 1, 2}:
    278     raise ValueError(
    279         f"IPython won't let you open fd={file} by default "
    280         "as it is likely to crash IPython. If you know what you are doing, "
    281         "you can use builtins' open."
    282     )
--> 284 return io_open(file, *args, **kwargs)
```

List Comprehension

- Fancy, shorthand method of writing for loops
- Syntax changes depending on use case
- can be nested in each other, just like lists
- Can contain multiple for loops in one list comp
- Can also be a nested loop

  Syntax

  - [x for x in iterable]
  - [x for x in iterable if (condition)]
  - [x if (condition) else y for x in iterable]
  - [x for sub_val in iterable for x in sub_val]

In [15]:
```
with open('files/names.csv', 'r') as f:
    f.readline() # use readline to skip the first line
    data = f.readlines()
data
```

Out[15]:
```
['Emily,Wilson\n',
 'Jennifer,Sanchez\n',
 'Denise,Ramos\n',
 'Manuel,Barnett\n',
 'Ashley,Hampton']
```

In [16]:
```
# If I want to remove the newline from the previous output
data = [name.strip() for name in data[1:]]
data
```

Out[16]:
```
['Jennifer,Sanchez', 'Denise,Ramos', 'Manuel,Barnett', 'Ashley,
Hampton']
```

In [17]:
```
data= [[1,2], [3,4]]

[x*2 for lst in data for x in lst]
```

Out[17]:
```
[2, 4, 6, 8]
```

In [18]:
```
output = []
for lst in data:
    for x in lst:
        output.append(x*2)
output
```

Out[18]:
```
[2, 4, 6, 8]
```

Checkpoint

What is the result of the following list comp?

In [ ]:
```
[(x*2, x**2) for x in range(0,10) if (x**2) % 2 == 1]
```

A.[(0, 0), (2, 1), (4, 4), (6, 9), (8, 16), (10, 25), (12, 36), (14, 49), (16, 64), (18, 81)]

B. [(0, 0), (4, 4), (8, 16), (12, 36), (16, 64)]

C. [(2, 1), (6, 9), (10, 25), (14, 49), (18, 81)]

D. []

Checkpoint Solution

In [20]:
```
# C
[(x*2, x**2) for x in range(0,10) if (x**2) % 2 == 1]
```

Out[20]:
```
[(2, 1), (6, 9), (10, 25), (14, 49), (18, 81)]
```

Dictionary Comprehension

- Fancy, shorthand method of populating dictionaries
- Syntax changes depending on use case

Syntax

- basically the same as list comp, but now it expects key:value
- can include a list comp!

In [20]:
```
# If I want to now create a dictionary of first name:last name
{x.split(',')[0]:x.split(',')[1] for x in data}
```

Out[20]:
```
{'Emily': 'Wilson',
 'Jennifer': 'Sanchez',
 'Denise': 'Ramos',
 'Manuel': 'Barnett',
 'Ashley': 'Hampton'}
```

Checkpoint

What is the result of the following dict comp?

In [ ]:
```
{x:[x + 2 for x in range(x)] for x in range(0,5) if x%2==0}
```

A. {0: [], 2: [2, 3], 4: [2, 3, 4, 5]}

B. {1: [2], 3: [2, 3, 4], 5: [2, 3, 4, 5, 6]}

C. {}

Checkpoint Solution

In [13]:
```
{x:[x + 2 for x in range(x)] for x in range(0,5) if x%2==0}
```

Out[13]:
```
{0: [], 2: [2, 3], 4: [2, 3, 4, 5]}
```

Assert Statements

- Used to evaluate written code
- **asserts** -> input validation (are the arguments the correct types?)
- Often combined with boolean functions (any(), all(), etc.)

In [54]:
```
def check_inputs(lst, word, number):
    assert isinstance(lst, list)
    assert isinstance(word, str)
    assert isinstance(number,int)
```

In [55]:
```
check_inputs([], 'a', 2)
```

In [56]:
```
check_inputs('a', 'a', 2)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent
call last)
Cell In[56], line 1
----> 1 check_inputs('a', 'a', 2)

Cell In[54], line 2, in check_inputs(lst, word, number)
      1 def check_inputs(lst, word, number):
----> 2     assert isinstance(lst, list)
      3     assert isinstance(word, str)
      4     assert isinstance(number,int)
```

Checkpoint

Given the following code, select the incorrect assert statement

In [12]:
```
def foo(input_lst):
    """
    function to flatten a nested list of strings.

    Args:
        input_lst (list): nested list of strings
    Returns:
        a "flattened" list.
    """
    return [x for lst in input_lst for x in lst]
foo([['a'], ['b'], ['c']])
```

Out[12]:
```
['a', 'b', 'c']
```

A. assert isinstance(input_lst, list)

B. assert all([isinstance(lst, list) for lst in input_lst])

C. assert all([isinstance(x, str) for x in input_lst])

D. assert all([isinstance(x, str) for lst in input_lst for x in lst])

Checkpoint Solution

In [28]:
```
# C
input_lst = [['a'], ['b'], ['c']]
```

In [24]:
```
assert isinstance(input_lst, list)
```

In [25]:
```
assert all([isinstance(lst, list) for lst in input_lst])
```

In [26]:
```
assert all([isinstance(x, str) for lst in input_lst for x in lst])
```

In [27]:
```
assert all([isinstance(x, str) for x in input_lst])
```

```
---------------------------------------------------------------
-----------
AssertionError                            Traceback (most recent
call last)
Cell In[27], line 1
----> 1 assert all([isinstance(x, str) for x in input_lst])

AssertionError:
```

Thanks for coming!