Miscellaneous Topics

Final exam preparation: wrapping up details that may not have been covered or not covered in sufficient detail.

James Tam

Logic: Alternative Representations

Alternatives
0 in place of False
1 in place of True

		Logical-AND
0	0	1
0	1	0
1	0	0
1	1	1

Both are the same

		Logical-AND
F	F	F
F	T	F
T	F	F
T	T	T

James Tam

Functions: Use Of Return

- Functions can return a tuple:
  - **Empty ()**
  - **Non-empty (1,2,3)**
  - Nothing or the 'None' type (**no return**)

```
def ageMap(age):
    if((age>0)and(age<=2)):
        return("Baby")
    elif((age>2)and(age<=4)):
        return("Toddler")
    elif((age>5)and(age<=12)):
        return("Pre-teen")
    elif((age>12)and(age<=19)):
        return("Teen")
    #Last cases are stylistically poor but included for learning
    elif((age>19)and(age<=64)):
        return()
    else:
        pass #No explicit return is possible for this case
```

James Tam

Files And Paths

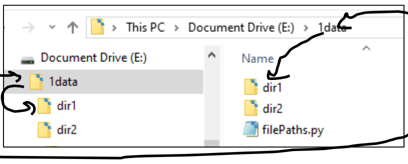
- **New definition, Absolute path**: specifies the full path in the directories/folders e.g. `C:\Program Files\Java\jdk-24\bin` (installation location of a version of java)
  - Rough real world analogy when giving directions: starting at the Calgary Tower you then provide directions to your house.
  - It's a rough analogy because while the 'C' drive contains the "Program Files" folder, the Calgary Tower doesn't contain the entire city.

James Tam

Files And Paths (2)

New definition, Relative path: specifies a path relative to the current location.

- Example:
  - Current location: `E:\1data`
  - Location of a file (to be accessed via file input or output): `E:\1data\dir1`
  - Because `dir1` is contained within the `1data` folder it is visible at this point so the full path doesn't have to be specified.



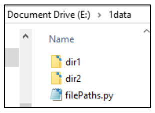
dir1 inside of 1data

  - Instead a path relative to the current location can be used instead i.e. a prompt for a location of a file to open can just specify the name of the folder (and then the name of the file) e.g. **dir1\input1.txt** (assuming file is here).
  - Rough real world analogy when giving directions: given the person is currently at the U of C your first step will begin there ("walk to the Bio. building and take overpass to the LRT that is heading downtown:).

James Tam

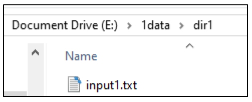
Example: Absolute Path

- File system for this example path to the program.



```
path = input("Path to file (e.g. 'C:\data\'): ")
fileName = input("Name of file (e.g. 'input.txt'): ")
aFile = open(path+fileName)
print("File", fileName, "contains:", aFile.read())
```

- File system for this example path to input file



- Run of program (**full absolute path**)

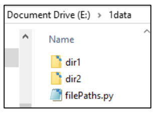
```
Path to file (e.g. 'C:\data'): E:\1data\dir1\
Name of file (e.g. 'input.txt'): input1.txt
File input1.txt contains: This is input1's contents
```

JT: what you enter for the path MUST match with where you downloaded this example

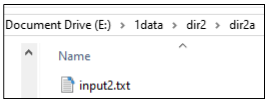
James Tam

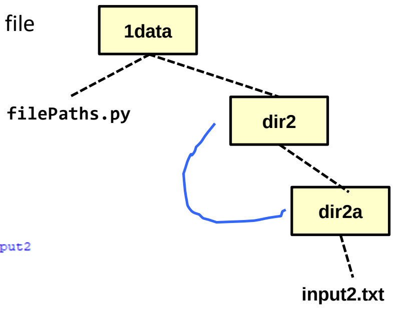
Example: Relative Path

- File system for this example path to the program (same).



- File system for this example path to input file





- Run of program (**relative path**)

```
Path to file (e.g. 'C:\data'): dir2\dir2a\
Name of file (e.g. 'input.txt'): input2.txt
File input2.txt contains: This is whats in file input2
```

James Tam

Creating Different Types

```
a = ""
print(type(a))
b = []
print(type(b))
c = ()
print(type(c))
d = {}
print(type(d))
e = set()
print(type(e))
```

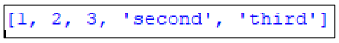
```
<class 'str'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'set'>
```

James Tam

Some List Methods

- Append: you have seen it applied many times
- Extend:

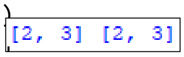
```
aList = [1,2,3]
aList.extend(["second","third"])
print(aList)
```



```
[1, 2, 3, 'second', 'third']
```

- Copy (reference: https://docs.python.org/3/library/copy.htm)

```
aList1 = [2,3]
aList2 = aList1.copy()
print(aList1,aList2)
```



```
[2, 3] [2, 3]
```

  - This method only performs a shallow copy, use `deepcopy()` for a full deep copy).

- Remove

```
aList1.remove(2)
```

```
[3]
```

- These are the list methods of importance for this course (there are others).

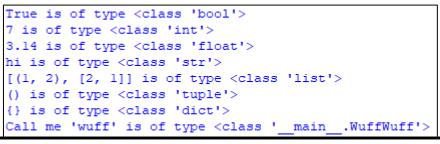
James Tam

A Composite's Elements Can Be Composite

- A list's elements can consist of another list (e.g. 2D list = sequence of 1D lists).
- Example (list is used but this applies to other composites):

```
class WuffWuff:
    def __init__(self):
        self.name = "Call me 'wuff'"
    def __str__(self):
        return(self.name)

w = WuffWuff()
aList =  [True,7,3.14,"hi",[(1,2),[2,1]],(),{},w]
for element in aList:
    print(element, "is of type", type(element))
```



```
True is of type <class 'bool'>
7 is of type <class 'int'>
3.14 is of type <class 'float'>
hi is of type <class 'str'>
[(1, 2), [2, 1]] is of type <class 'list'>
() is of type <class 'tuple'>
{} is of type <class 'dict'>
Call me 'wuff' is of type <class '__main__.WuffWuff'>
```

James Tam

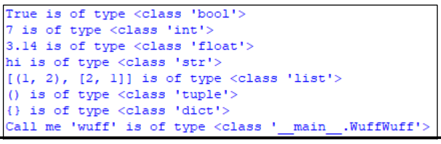
A Composite's Elements Can Be Composite

- A list's elements can consist of another list (e.g. 2D list = sequence of 1D lists).
- Example (list is used but this applies to other composites):

```
class WuffWuff:
    def __init__(self):
        self.name = "Call me 'wuff'"
    def __str__(self):
        return(self.name)

w = WuffWuff()
aList =  [True,7,3.14,"hi",[(1,2),[2,1]],(),{},w]
for element in aList:
    print(element, "is of type", type(element))
```

JT: (advanced concept beyond 217) A true deep copy doesn't just iterate the list and copy elements but instead it must determine if each element is immutable.



```
True is of type <class 'bool'>
7 is of type <class 'int'>
3.14 is of type <class 'float'>
hi is of type <class 'str'>
[(1, 2), [2, 1]] is of type <class 'list'>
() is of type <class 'tuple'>
{} is of type <class 'dict'>
Call me 'wuff' is of type <class '__main__.WuffWuff'>
```

James Tam

Class Attributes: FYI Class Attributes Can Be Composite

```
class Person():
    def __init__(self, aName):
        self.myName = aName
        self.myFriends = []

aPerson = Person("JT")
```

James Tam

Copyright Notification

- Unless otherwise indicated, all images in this presentation were provided courtesy of James Tam.

slide 13

James Tam