ECE 20875
Python for Data Science
Milind Kulkarni and Chris Brinton

Data Structures
coding in python

- Standard Integrated Development Environments (IDEs)
  - IDLE: Python’s own, basic IDE
  - PyCharm: Code completion, unit tests, integration with git, many advanced development features
  - Many more!
- Jupyter Notebook ([https://jupyter.org/](https://jupyter.org/))
  - Contains both computer code and rich text elements (paragraphs, figures, …)
  - Supports several dozen programming languages
  - Very useful for data science development!
  - You can download the notebook app or use Jupyter Hub available on RCAC
basic variables

• No “declaration” command as in other programming languages
  • Variable is created when a value is assigned to it
  • Can change type after they have been set
• Few rules on naming: Can make them very descriptive!
  • Must start with a letter or underscore
  • Case-sensitive (purdue & Purdue are different)
• Combinations (+) work on all types
  “xyz ” + “abc” = “xyz abc”
  3.2 + 1 = 4.2
control statements

• Logical conditions
  
a == b, a != b, a < b,
  
a <= b, a > b, a >= b

• If, elif, else
  
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")

• while loop: Execute while condition is true
  
i = 1
  while i < 6:
    print(i)
    i += 1

• for loop: Iterate over a sequence
  
for x in "banana":
  print(x)

• break: Stop a loop where it is and exit

• continue: Move to next iteration of loop
lists

• One of the four collection data types

• Also tuples, sets, and dictionaries

• Lists are ordered, changeable, and allow duplicate members

```python
thislist = ["apple", "banana", "apple", "cherry"]
```

• Can pass in an integer index, or a range of indexes

```python
thislist[0] = "apple"
thislist[-1] = "cherry"
thislist[1:3] = ["banana", "apple"]
```

• Length using len() method

```python
print(len(thislist))
```

• Adding items to a list

```python
thislist.append("orange")
thislist.insert(1, "orange")
```

• Removing items from a list

```python
thislist.remove("banana")
thislist.pop(1)
```

• Defining lists with shorthand

```python
new_list = 5 * [0]
new_list = range(5)
```
lists in for loops

• In other programming languages, for loop variables are integers

• In Python, can use any ‘iterable’ object

```
fruits = ['apple', 'banana', 'cherry']
for x in fruits:
    if x == 'banana':
        continue
    print(x)
```

• Nested loops can be used too

```
adj = ['red', 'big', 'tasty']
fruits = ['apple', 'banana', 'cherry']
for x in adj:
    for y in fruits:
        print(x, y)
```

• Can also iterate through a list of lists

```
data_list = [[1,2],[2,6],[5,7]]
for point in data_list:
    [x,y] = point
    z = x ** 2
    print(x,y,z)
```

• Can use the range function to iterate through integers

```
for x in range(2, 30, 3):
    print(x)
```

• Can use a list to index another list

```
ind = [1, 3, 5, 7]
values = [0] * 8
for i in ind:
    values[i] = i / 2
```
functions

• Block of code which runs when called

• Defined using `def` keyword
  ```python
  def my_function():
      print("Hello from a function")
  ```

• Call a function using its name
  ```python
  my_function()
  ```

• Parameters can be passed as input to functions
  ```python
  def my_function(country):
      print("I am from " + country)
  ```

• To return a value, use the `return` statement
  ```python
  def my_function(x):
      return 5 * x
  ```
  ```python
  print(my_function(3))
  print(my_function(5))
  ```

• For multiple arguments, can use keywords to specify order
  ```python
  def arithmetic(x,y,z):
      return (x+y)/z
  ```
  ```python
  print(arithmetic(z=3,x=2,y=4))
  ```
tuples

• Another of the four collection data types

• Tuples are ordered, unchangeable, and allow duplicate members

  thistuple = ("apple", "banana", "apple", "cherry")

• Indexed the same way as lists

  thistuple[0] = "apple"
  thistuple[-1] = "cherry"
  thistuple[1:3] = ("banana", "apple")

• Once a tuple is created, items cannot be added or changed

• Workaround: Change to list, back to tuple

• Check if item exists

  if "apple" in thistuple:  
    print("Yes, 'apple' is in the fruits tuple")

• Tuple with one item needs comma

  thistuple = ("apple",)  #Tuple
  thistuple = ("apple")  #Not a tuple

• Built in functions

  thistuple.count("apple")
  thistuple.index("apple")
sets

• Collection which is **unordered**, (half) **changeable**, and does **not** allow duplicates

• Written with curly brackets

```python
def thisset = {"apple", "banana", "cherry"}
```

• Cannot access items by index, but can loop through and check for items

```python
for x in thisset:
    print(x)
print("banana" in thisset)
```

• Cannot change existing items, but can add and remove items

```python
thisset.add("orange")
thisset.update(["orange", "mango", "grapes"])
thisset.remove("banana")
```

• Also have set operations just like mathematical objects

```python
set1 = {"a", "b", "c"}
set2 = {1, "b", 3}

set1.union(set2)  # Union
set1.intersection(set2)  # Intersection
set1.difference(set2)  # set1 \ set2
set1.issubset(set2)  # Testing if subset
```
dictionaries

• Collection which is unordered, changeable, and indexed

• Also written with curly brackets, but have keys and values

```python
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
```

• Access/change/add values of items by referring to the key name

```python
thisdict["model"]
thisdict["year"] = 2019
thisdict["color"] = "red"
```

• Can iterate through the keys, values, or both:

```python
for x in thisdict:
    print(thisdict[x])
for x in thisdict.values():
    print(x)
for x, y in thisdict.items():
    print(x, y)
```

• Like other collections, can create a dictionary of dictionaries

```python
child1 = {
    "name": "Emil",
    "year": 2004
}
child2 = {
    "name": "Tobias",
    "year": 2007
}
child3 = {
    "name": "Linus",
    "year": 2011
}
myfamily = {
    "child1": child1,
    "child2": child2,
    "child3": child3
}
```

• Use the copy method (not direct assignment) to make a copy of a dictionary

```python
mydict = dict(thisdict)
```