Week 8 Tutorial: Fundamentals of Python Programming I

POP77001 Computer Programming for Social Scientists

Naming Conventions

It is a good practice to follow usual naming convention when writing code.

- Use **UPPER_CASE_WITH_UNDERSCORE** for named constants (e.g. variables that remain fixed and unmodified)
- Use **lower_case_with_underscores** for function and variable names
- Use **CamelCase** for classes (more on them later)

Extra

PEP 8 – Style Guide for Python Code

Code Layout

- Limit all lines to a maximum of 79 characters.
- Break up longer lines:

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]

result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
    )

income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
          - ira_deduction
          - student_loan_interest)
```

Extra

PEP 8 – Style Guide for Python Code

Reserved Words

There are 35 reserved words (keywords) in Python (as of version 3.9) that cannot be used as identifiers.

`and`	`continue`	`finally`	`is`	`raise`
`as`	`def`	`for`	`lambda`	`return`
`assert`	`del`	`from`	`nonlocal`	`True`
`async`	`elif`	`global`	`None`	`try`
`await`	`else`	`if`	`not`	`with`
`break`	`except`	`import`	`or`	`while`
`class`	`False`	`in`	`pass`	`yield`

```python
try = 5 # Watch out for reserved words
```

```
  Cell In[4], line 1
    try = 5 # Watch out for reserved words
        ^
SyntaxError: expected ':'
```

Extra

Defining Variables

- Assignment statement binds the variable name and an object.

```python
x = 5 # Variable 'x' is bound to object 5 of type integer
x
```

```
5
```

- The same object can have multiple names (⚠ more on aliasing and copying below)

```python
y = x
y
```

```
5
```

```python
# Note that x was overwritten even with addition operation as integers are immutable
x += 3
x
```

```
8
```

```python
y
```

```
5
```

Strings

- String (`str`) - <u>immutable ordered</u> sequence of characters.
- **Immutable** - individual elements cannot be modified.
- **Ordered** - strings can be sliced (unlike in R).

```python
s = 'test'
s
```

```
'test'
```

```python
s[0] = 'r' # immutability
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 s[0] = 'r' # immutability

TypeError: 'str' object does not support item assignment
```

```python
s[0] # slicing (indexing starts from 0!)
```

```
't'
```

String Methods

```
s.capitalize()
s.title()
s.upper()
s.lower()
s.find(some_string)
s.replace(one_string, another_string)
s.strip(some_string)
s.split(some_string)
s.join(some_list)
```

Extra

Python string methods

Method Chaining

- Recall from the lecture that methods can be **chained**
- E.g. `s.strip().replace('--', '---').title()`
- It provides a shortcut (does not necessitate intermediate objects)
- However, it can reduce code legibility! 📜

Exercise: Working with Strings

- Remove trailing whitespaces (before and after the sentence) in the string below;
- Replace all double whitespaces with one;
- Format it as a sentence with correct punctuation;
- Print the result.

```
s = "   truth  can  only be  found in  one place:  the  code "
```

Lists

- List (`list`) - mutable ordered sequence of elements.
- **Mutable** - individual elements can be modified.
- **Ordered** - lists can be sliced (like strings).

```python
l = [1, 2, 3]
l
```

```
[1, 2, 3]
```

```python
l[1] = 7 # mutability
l
```

```
[1, 7, 3]
```

```python
l[0] # slicing
```

```
1
```

List Methods

```
l.append(some_element)
l.extend(some_list)
l.insert(index, some_element)
l.remove(some_element)
l.pop(index)
l.sort()
l.reverse()
l.copy()
```

Extra

Python list methods

Aliasing vs Copying - Immutable

- Having multiple names for the same object doesn't usually create a problem with immutable types, as the entire object just gets overwritten.

```python
x = 5
y = x # Object 5 of type integer is not copied, y is just an alias!
```

```python
x
```

```
5
```

```python
y
```

```
5
```

```python
id(x) # function id() prints out unique object identifier
```

```
11760808
```

```python
id(y)
```

```
11760808
```

```python
x += 3
```

```python
print(x)
print(y)
print(id(x))
print(id(y))
```

```
8
5
11760904
11760808
```

Aliasing vs Copying - Mutable

```python
l = [1, 2, 3]
```

```python
# Object [1, 2, 3] of type list is not copied, l1 is just an alias!
l1 = l
```

```python
# Both [:] slicing notation and copy method create copies
l2 = l[:]
l3 = l.copy()
```

```python
l1.pop(0) # Remove (and return) first element of the list
l2.insert(0, 0) # Insert 0 as the first element of the list
l3.append(4) # Append 4 to the end of the list
```

```python
print(l)
print(l1)
print(l2)
print(l3)
```

```
[2, 3]
[2, 3]
[0, 1, 2, 3]
[1, 2, 3, 4]
```

Exercise: Working with Lists

- Below is a shuffled version of the first 11 elements of Fibonacci sequence.
- Create a copy of the shuffled list;
- Remove the last element;
- Sort it from smaller integers to larger;
- Select the second smallest and the third largest integers in the sequence; Print them out;
- Replace them in the list with the string, containing word corresponding to that number (e.g. ‘two’ for 2);
- Print out the results.

```python
fib_shuffled = [34, 5, 3, 1, 13, 55, 21, 2, 8, 0, 1]
```

Week 8 Exercise (unassessed)

- Practice working with built-in Python data structures