Python Master Reference

LIN 301: Computation for Linguists

Compiled by Dr. Andrew M. Byrd

Table of contents

1. Python Setup & Basics

Variables & Data Types

- Python is dynamically typed: variable types are inferred.

```
x = 10          # int
y = 3.14        # float
name = "Alice"  # str
is_ready = True # bool
```

- Use `type(x)` to check a variable's type.
- Basic types: `int`, `float`, `str`, `bool`, `list`, `tuple`, `dict`.

Converting Types

```
int("42")   # 42
float(3)    # 3.0
str(3.14)   # '3.14'
```

Printing & Input

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

---

2. Strings, Lists, and Loops

Strings

```
s = "Linguistics"
print(s[0])      # 'L'
print(s[-1])     # 's'
print(s[0:4])    # 'Ling'
```

- `len(s)` returns the length.
- Methods: `.lower()`, `.upper()`, `.replace()`, `.split()`.

Lists

```
words = ["cat", "dog", "rabbit"]
words.append("mouse")
print(words[1])
```

- `list(range(5))` → `[0, 1, 2, 3, 4]`
- Slicing works just like strings.

Loops

```
for w in words:
    print(w)

for i in range(5):
    print(i)
```

Conditionals

```
if x > 10:
    print("Large")
elif x > 5:
    print("Medium")
else:
    print("Small")
```

List Comprehensions

```
# Basic
squares = [x**2 for x in range(10)]

# With condition
evens = [x for x in range(20) if x % 2 == 0]

# Nested (flatten a list of lists)
matrix = [[1,2,3],[4,5,6]]
flat = [n for row in matrix for n in row]
```

---

3. Dictionaries & Functions

Dictionaries

```
ages = {"Alice": 25, "Bob": 30}
print(ages["Alice"])
ages["Carol"] = 22
for name, age in ages.items():
    print(name, age)
```

- `.keys()`, `.values()`, `.items()`

Dict & Set Comprehensions

```
# Dict
word_lengths = {w: len(w) for w in ["queen","rabbit","tea"]}

# Invert a dict (only if values are unique)
d = {"a":1,"b":2}
inv = {v:k for k,v in d.items()}

# Set (unique squares)
unique_squares = {x**2 for x in [1,2,2,3,3,3]}
```

Functions

```
def greet(name):
    return f"Hello, {name}!"

def add(x, y=0):
    return x + y
```

- Default arguments and returns.
- Use docstrings to document:

```
def square(n):
    '''Return the square of n.'''
    return n ** 2
```

---

4. Text Files

Text Files (UTF-8)

```
from pathlib import Path
p = Path("data") / "alice_en.txt"

# Read whole file
text = p.read_text(encoding="utf-8")

# Write whole file
out = Path("out.txt")
out.write_text(text, encoding="utf-8")
```

Reading & Writing Files

```
with open("data.txt", "r", encoding="utf-8") as f:
    text = f.read()

with open("output.txt", "w", encoding="utf-8") as f:
    f.write("Hello world!")
```

5. Pandas

Pandas Basics

```
import pandas as pd

df = pd.read_csv("data.csv")
print(df.head())

# Filtering
subset = df[df["count"] > 10]

# New column
df["double"] = df["count"] * 2
```

CSV Basics with pandas

```
import pandas as pd

df = pd.read_csv("data/words.csv")            # read
df.to_csv("data/words_clean.csv", index=False) # write

# Useful options
df = pd.read_csv("data/words.csv", encoding="utf-8", na_filter=True)
```

Value Counts & Top-N

```
import pandas as pd
s = pd.Series(["alice","queen","alice","rabbit"])
top = s.value_counts().head(10)
```

Groupby + Aggregate

```
df = pd.DataFrame({"lang":["en","en","fr"], "count":[5,3,4]})
agg = df.groupby("lang", as_index=False)["count"].sum()
```

Merge / Join

```
left = pd.DataFrame({"id":[1,2], "lemma":["queen","rabbit"]})
right = pd.DataFrame({"id":[1,2], "freq":[7,3]})
merged = left.merge(right, on="id", how="left")
```

Apply (row-wise function)

```
def is_long(w): return len(w) > 5
df["is_long"] = df["lemma"].apply(is_long)
```

---

5. spaCy

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Alice chased the rabbit."
doc = nlp(text)

for tok in doc:
    print(tok.text, tok.lemma_, tok.pos_, tok.dep_)
```

- `tok.text`: actual token
- `tok.lemma_`: base form
- `tok.pos_`: part of speech
- `tok.dep_`: syntactic dependency

Named Entities

```
for ent in doc.ents:
    print(ent.text, ent.label_)
```

---

6. WordNet & Semantics

```
from nltk.corpus import wordnet as wn
wn.synsets("dog")[:3]
```

Synonyms & Hypernyms

```
dog = wn.synset("dog.n.01")
print(dog.definition())
print(dog.hypernyms())
```

Similarity

```
cat = wn.synset("cat.n.01")
print(dog.wup_similarity(cat))
```

---

7. Visualization

Matplotlib

```
import matplotlib.pyplot as plt
x = [1,2,3,4]
y = [10,20,25,30]
plt.plot(x, y)
plt.title("Simple Line Plot")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()
```

Bar Chart

```
langs = ["English","French","Finnish"]
counts = [1200, 950, 870]
plt.bar(langs, counts)
plt.title("Word Counts per Language")
plt.show()
```

Word Cloud

```
from wordcloud import WordCloud
text = "Alice was beginning to get very tired of sitting by her sister"
wc = WordCloud(width=400, height=200).generate(text)
plt.imshow(wc)
plt.axis("off")
plt.show()
```

---

8. Common Errors & Fixes

Reading Tracebacks

```
# Example
# Traceback (most recent call last):
#   File "...", line 1, in <module>
# NameError: name 'df' is not defined
```

Error	Meaning	Fix
`NameError`	Variable not defined	Check spelling, define before use
`TypeError`	Wrong type in operation	Cast to correct type (`int()`, `str()`)
`KeyError`	Missing key in dictionary	Use `.get()` or check with `in`
`IndexError`	List index out of range	Use `len(list)` before accessing
`ModuleNotFoundError`	Missing library	`pip install <library>`

---

9. Best Practices

- Write readable variable names: `word_count`, not `wc`.
- Use comments sparingly but clearly.
- Keep functions short and single-purpose.
- Use version control (GitHub) and commit early.
- Check for UTF-8 encoding when handling text.
