Illinois Python Cheat Sheet
by Elizabeth de Sa e Silva, Tamara Nelson-Fromm, Wade Fagen-Ulmschneider

Basic Data Types

Integers are whole numbers
\[
\begin{align*}
\text{int1} &= 8 \\
\text{int2} &= -5 \\
\text{int3} &= 0 \\
\text{int4} &= \text{int}(4.0)
\end{align*}
\]
Floats have a decimal point
\[
\begin{align*}
\text{float1} &= 5.5 \\
\text{float2} &= 0.0 \\
\text{float3} &= 1e6 \\
\text{float4} &= \text{float}(2)
\end{align*}
\]

Strings
A string literal has quotes: ‘CS101’, ‘CS107’, ‘5.67’

- (it’s literally the exact characters of the string)
- A variable name does not: course_name, stat107, my_string
- A string can be indexed the same way as a list

Example
\[
\text{my_string} = 'literal' \\
\text{print('my_string')} \\
\text{print(literal)}
\]

Booleans
Booleans are True or False values
\[
\begin{align*}
\text{x} &= \text{y} \quad \text{is True if } x \text{ is equal to } y \\
\text{x} \in \text{y} &= \text{True if } x \text{ is an element of } y
\end{align*}
\]

And
\[
\begin{align*}
\text{True and True} &= \text{True} \\
\text{True and False} &= \text{False} \\
\text{False and False} &= \text{False}
\end{align*}
\]
Or
\[
\begin{align*}
\text{True or True} &= \text{True} \\
\text{True or False} &= \text{True} \\
\text{False or False} &= \text{False}
\end{align*}
\]

Slicing
Strings, lists, and other iterable data types (data with many elements) can be indexed over a range of values, or sliced

Replace any [ ] with a range to select many elements at once:

\[
\begin{align*}
\text{[start:stop:step]} & \quad \text{Selects position start through position stop, not including stop, but only elements step positions apart; start defaults to zero, so } [0:7] \text{ starts at } 0 \\
& \quad \text{stop defaults to one past the last index, so } [10:2] \text{ selects through the end of the data} \\
& \quad \text{step defaults to one, so } [1:5] \text{ steps by 1 (a negative step will count backwards)}
\end{align*}
\]

Example
\[
\begin{align*}
\text{my_string} &= \text{'abcdefgihj'} \\
\text{my_string}[2:4] &= \text{'cd'} \\
\text{my_string}[5:] &= \text{'abcde'} \\
\text{my_string}[\text{start:stop:step}] &= \text{fghijk'} \\
\text{my_string}[\text{start:stop:step}] &= \text{abcdefgihj'} \\
\text{my_string}[\text{start:stop:step}] &= \text{'cd'} \\
\text{my_string}[\text{start:stop:step}] &= \text{c'eg'} \\
\text{my_string}[\text{start:stop:step}] &= \text{'ige'}
\end{align*}
\]

Lists

Creating a new list
\[
\text{empty_list} = []
\]
\[
\text{my_list} = [\text{1,2,3}]
\]

Adding to a list (appending)
\[
\text{list_name} += [\text{v1,v2}]
\]

Indexing
\[
\text{list[i]} \quad \text{is equal to the element in list at zero-based index } i
\]
- Negative index values count from the end of the data
  \[
  \text{list[-i]} \quad \text{is equal to list[ len(list) - i ]}
  \]

Changing a list
\[
\text{list[i]} = \text{v} \quad \text{# changes the element in list at position } i \text{ to the value } v
\]

Example
\[
\begin{align*}
\text{my_list} &= [10,20,30] \\
\text{my_list} &= [50,60] \\
\text{my_list} &= [30] \\
\text{my_list} &= [60]
\end{align*}
\]

Dictionaries

Creating a new dictionary
\[
\text{my_dict} = \{\text{key1:value1, key2:value2, ... , keyn:valuen}\}
\]
\[
\text{empty_dict} = {}
\]

Adding to a dictionary (appending)
\[
\text{dict_name[key]} = \text{value}
\]
\[
\text{# adds key:value to dict_name}
\]

Indexing
\[
\text{dict[key]} \quad \text{is equal to the value in dict with key key}
\]

Changing a dictionary
\[
\text{dict_name[key]} = \text{value} \quad \text{# changes key's value to v so dict_name now has the pair key:v}
\]

Example
\[
\begin{align*}
\text{my_dict} &= \{\text{'a':5, 'b':6}\} \\
\text{my_dict} &= \{\text{'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\}
\end{align*}
\]

Getting Keys and Values
\[
\text{dict_name.keys()} \\
\text{dict_name.values()}
\]
\[
\text{# returns a list of keys in dict_name} \\
\text{# returns a list of values in dict_name}
\]

Example
\[
\begin{align*}
\text{my_dict} &= \{\text{'a':5, 'b':6}\} \\
\text{my_dict} &= \{\text{'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\} \\
\text{my_dict} &= \{\text{'a':5, 'b':6, 'c':4}\}
\end{align*}
\]
# If Statements

**if**  
Indicates a block of code that only runs if its boolean condition is True  
**elif**  
Short for "else if"; this block is associated with an if block and has a condition; it only runs if its condition is true and the original if block condition was false  
**else**  
This block has no condition and runs only if the associated if statement and any of its elif blocks did not run

**Example**  
if x < 5:  
  #this indented code only runs if x is less than 5  
elif x < 10:  
  #this only runs if x is greater than 5 and less than 10  
elif x == 13:  
  #this only runs if x is equal to 13  
else:  
  #this only runs if x is greater than 10 and is not 13

# For Loops

**for**  
Repeats a block of code for every element of an iterable data type  
**for i in iterable:**  
#code block to repeat  
Does not require you to advance the variable i

**Example:**  
```python  
list = ['CS101','CS107','ILL']  
for item in list:  
  #loops over every element  
  print(item)  
```

This code prints:
CS101  
CS107  
ILL

**range(start, stop, step)**  
Generates a list of all integers from start to stop, jumping by step  
**start**  
The very first integer of the sequence. This defaults to 0 if not specified  
**stop**  
The boundary for the end of the sequence. This number is not included in the actual sequence of number. Has no default value and must always be specified  
**step**  
The spacing between numbers included in the sequence. This defaults to 1

# While Loops

**while**  
Repeats a block of code while some condition is true  
Often requires you to change the variables the condition relies on in the code block to get the loop to ever stop

**Example:**  
```python  
while this_is_true:  
  #code block to repeat  
```

**Example:**  
```python  
while n > 0:  
  result = result * n  
  n = n - 1  
```

# Accumulator Patterns

**Example:**  
```python  
package_weights = [2, 6.5, 1, 10]  
total = 0  
for weight in package_weights:  
  total += weight  
print(total)  
```

# Pandas

**Example:**  
```python  
data = []  
for i in range(50):  
  coin = randint(0,1)  
  d = {'coin': coin}  
  data.append(d)  
df = pandas.DataFrame(data)  
```