Python Cheat Sheet

General Functions

`input(prompt)`
Pauses and reads a line of text typed by the user.
```
name = input("Your name: ")
```

`len(obj)`
Returns the number of items in a string, list, dict, or other collection.
```
len("hello") -> 5
len([1,2,3]) -> 3
```

`range(start, stop, step)`
Generates a sequence of integers. start and step are optional.
```
for i in range(1, 6): # produces 1 2 3 4 5
```

`int() / float() / str()`
Converts values between number and string types.
```
int("42") -> 42
str(99) -> "99"
```

`type(obj)`
Returns the data type of any object.
```
type(3.14) -> <class 'float'>
```

`sum() / min() / max()`
Computes the total, smallest, or largest value in a sequence.
```
sum([1,2,3]) -> 6
max([5,2,9]) -> 9
```

`abs(x) / round(x, n)`
Absolute value and rounding to n decimal places.
```
abs(-7) -> 7
round(3.14159, 2) -> 3.14
```

Strings

`s = "hello"` — text written inside quotes

`str.upper() / str.lower()`
Converts the entire string to upper or lower case.
```
"hello".upper() -> "HELLO"
```

`str.split(sep)`
Splits a string into a list using sep as the delimiter.
```
"a,b,c".split(",") -> ["a", "b", "c"]
```

`str.strip()`
Removes leading and trailing whitespace from a string.
```
" hi ".strip() -> "hi"
```

`str.replace(old, new)`
Replaces every occurrence of old with new.
```
"cat".replace("c", "b") -> "bat"
```

`str.join(iterable)`
Joins a list of strings, inserting the string between each item.
```
", ".join(["a","b","c"]) -> "a, b, c"
```

`str.find(sub)`
Returns the index of the first occurrence of sub, or -1.
```
"hello".find("ll") -> 2
```

`str.startswith(p) / str.endswith(s)`
Returns True if the string begins or ends with the given value.
```
"hello".startswith("he") -> True
```

`str.count(sub)`
Counts the number of non-overlapping occurrences of sub.
```
"hello".count("l") -> 2
```

`str.title()`
Capitalises the first letter of every word.
```
"hello world".title() -> "Hello World"
```

`str.isdigit() / str.isalpha()`
Returns True if every character is a digit or letter, respectively.
```
"123".isdigit() -> True
"abc".isalpha() -> True
```

`s[::-1]`
Slice notation reverses a string. Also works on lists.
```
"hello"[::-1] -> "olleh"
```

Lists

`fruits = ["apple", "banana", "cherry"]` — items inside square brackets, separated by commas

`list.append(x)`
Adds a single item to the end of the list.
`fruits.append("date") -> ["apple", "banana", "cherry", "date"]`

`list.pop(i)`
Removes and returns the item at index i. Defaults to the last item.
`fruits.pop() -> removes and returns "cherry"`

`del list[i]`
Deletes the item at index i. This is a statement, not a method.
`del fruits[0] # removes "apple", the first item`

`list.insert(i, x)`
Inserts x into the list at index i, shifting later items right.
`fruits.insert(1, "kiwi") -> ["apple", "kiwi", "banana", "cherry"]`

`list.remove(x)`
Removes the first item equal to x. Raises an error if not found.
`fruits.remove("kiwi") # deletes the value "kiwi"`

`list.sort()`
Sorts the list in place. Use sorted(list) to get a new sorted list.
`fruits.sort() -> ["apple", "banana", "cherry"]`

`x in list`
Checks whether a value exists in the list. Returns True or False.
`"banana" in fruits -> True`

`list[::-1]`
Slice notation creates a reversed copy of the list.
`fruits[::-1] -> ["cherry", "banana", "apple"]`

`len(list)`
Returns the number of items currently in the list.
`len(fruits) -> 3`

`sorted(list)`
Returns a new sorted list. The original list is left unchanged.
`sorted(["cherry", "apple", "banana"]) -> ["apple", "banana", "cherry"]`

Dictionaries

`age = {"sam": 12, "ana": 11}` — key/value pairs inside curly braces

`dict[key] = value`
Adds a new key-value pair, or updates the value if the key exists.
`age["max"] = 10 # adds "max" to the dictionary`

`dict[key]`
Retrieves the value stored under key. Raises an error if missing.
`age["sam"] -> 12`

`dict.get(key, default)`
Retrieves the value for key, or returns default if it is missing.
`age.get("zoe", 0) -> 0 # "zoe" not found`

`del dict[key]`
Deletes the key and its value from the dictionary.
`del age["max"] # removes the "max" entry`

`dict.keys()`
Returns a view of all the keys in the dictionary.
`age.keys() -> dict_keys(["sam", "max"])`

`dict.values()`
Returns a view of all the values in the dictionary.
`age.values() -> dict_values([12, 10])`

`dict.items()`
Returns key-value pairs, often used in a for loop.
`for k, v in age.items(): print(k, v)`

`key in dict`
Checks whether a key exists in the dictionary.
`"sam" in age -> True`

Random Module (add: import random)

`random.randint(a, b)`
Returns a random integer between a and b, inclusive.
`random.randint(1, 6) -> 4 (like rolling a die)`

`random.choice(seq)`
Returns a randomly selected element from a list.
`random.choice(["rock","paper","scissors"]) -> "rock"`

`random.shuffle(lst)`
Shuffles a list in place. Modifies the original; returns nothing.
`random.shuffle(cards) # deck is now in random order`