More Built-In Data Types

Name	Stores	Examples
`list`	an ordered collection of (any) values	`[3, 7, 42]`
`tuple`	an ordered collection of a fixed number of values	`(159, 222, 200)`
`range`	a collection of discreet values between start and finish	`range(2, 10, 2)`
`dict`	a collection of unique keys and their respective values	`{name: "John", age: 13}`
`set`	an unordered collection of unique values	`{"apple", "pear", "banana"}`

Learn more about other data types here.

Operations on Data Types

Lists

Operation	Syntax
access list item	`listname[1]; listname[-2]; listname[2:5]; listname[2:]; listname[:4]`
change list item	`listname[idx] = x; listname.insert(position, new_value)`
extend list	`listname.append(new_value); listname.extend(other_listname)`
sort list	`listname.sort()`

Learn about more list operations here.

Tuples

Operation	Syntax
access tuple item	`tuplename[idx]`
unpack tuple	`x, y, z = tuplename`

Learn about more tuple operations here.

Dictionaries

Operation	Syntax
access dict item	`dictname[key]; dictname.keys(); dictname.values()`
add items	`dictname[new_key] = new_value`
remove items	`dictname.pop(key)`

Learn about more list operations here.

Sets

Operation	Syntax
add items	`setname.add(new_item)`
remove items	`setname.remove(new_item)`

Learn more about set theory to better understand operations here. Learn about more list operations in Python here.

Basic Logic: Control Flows

General structure of control flows, commented-out parts (marked by ##) are optional:

`if-else`

```
if [condition]:
    consequent
##elif [condition]:
    ##consequent
##else:
    ##consequent
```

`match-case`

```
match [variable]:
    case [value_1]:
        consequent_1
    ##case [value_2]:
        ##consequent_2
    ##case [value_3]:
        consequent_3
    ...
```

`while` loops

```
while [condition]:
    action
```

`for` loops

```
for [item] in [sequence]:
    action
```

Control Flows: General Notes

1. **All control flow statements <u>must</u> be properly indented** as shown above (otherwise Python won't be able to parse them). Indent using the 'tab' key.
2. **Conditions typically contain logical, comparison, identity, or membership operators.** (Revise the different operators here.)
3. **Make sure there is *an* end to loops.** Otherwise they're fairly good at crashing computers. Consider including an action that modifies the condition and/or adding a `break` statement.
4. **For `if-else` and `match-case` statements: consequents cannot be empty**—consider using a `pass` statement that fulfils pretty much the same function.
5. **Loops are iterative statements that can step through any *iterable* data type.** Iterable data types include lists, sets, tuples, dictionaries, strings, or ranges.