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.

In-Class Problems

01 Use control flows to print your name exactly 11 times.

02 Use control flows to print every multiple of 3 between 2 and 22.

03 Create a variable score between 0 and 100. Use control flows to assign a letter grade (A: 90-100, B: 80-89, C: 70-79, D: 60-69, F: below 60).

04 Create a list with 5 different colours. Create a boolean variable `has_red` that yields True if the list contains red, False otherwise. Print the result.

05 Make a list of boolean values representing quiz answers. Calculate and print the percentage of True answers.

06 Create a list of floating-point numbers. Modify the list so that every number is rounded up to 2 decimal places.

07 Create a set with numbers 1-20. Remove all numbers divisible by 4 and print the remaining set.

08 Create two sets, `a` and `b`, with 7 numerical values between 0 and 9 each. Now create a set `c` that contains values of `a` without the values that also occur in `b`. (Hint: Recall set operations.)

09 Create a variable `day_idx` with a value between 1 and 365. Use control flows to determine which day of the week it is. (Hint: Consider using the modulo operator to find the pattern.)

10 Create a list of strings representing file names (include endings like '.pdf', '.txt', '.jpg', '.docx' etc.). Use iterators to filter out strings not representing text files and print the result.

11 Create a list of 10 random integers. Find the second largest number.

12 Create a list of dictionaries, each representing a person with name and age. Calculate the average age.

13 Create a list of 12 random names. Split these names into triplets alphabetically and print the result as a list.

14 Use control flows to find the result of 63487 divided by 7. No division or multiplication operators!

15 Everyone create a list of 10 colours in your respective repositories. Then merge your lists into one repository using GitHub (not manually) and find their intersection.

In-Class Problems: Advanced

01 Create a list with mixed data types (strings, numbers, booleans). Sort them into separate lists by type and print the new lists.

02 Create a dictionary with city names and populations. Find all cities with population over 1 million.

03 Create a tuple representing RGB colour values (0-255). Determine if it represents a shade of grey (all values equal).

04 Let's define 'matrix' as a list of lists containing an equal number of items of the same type. Create a few sample lists of lists and use control flows to determine whether or not they are matrices.

05 Consider the list `[1, 2, 3, 4, 3, 2, 3, 7]`. Get rid of all duplicates without control flows. Print the number of duplicates. (Hint: Consider using data casting.)

06 Make a set of 10 random names. Use control flows to find all names that contain the letter 'a'.

07 Make a list of random integers between 1 and 50. Find out how many are primes and print the result.

08 Create a string with your full name. Extract and print your initials.

09 Create two lists of 7 random numbers between 0 and 9. Print a dictionary holding their union, intersection, and symmetric difference.

10 Make a list of 0s and 1s in a random order. Then use control flows to count the number of islands (connected groups of the same numbers). Print the result.