Week 2

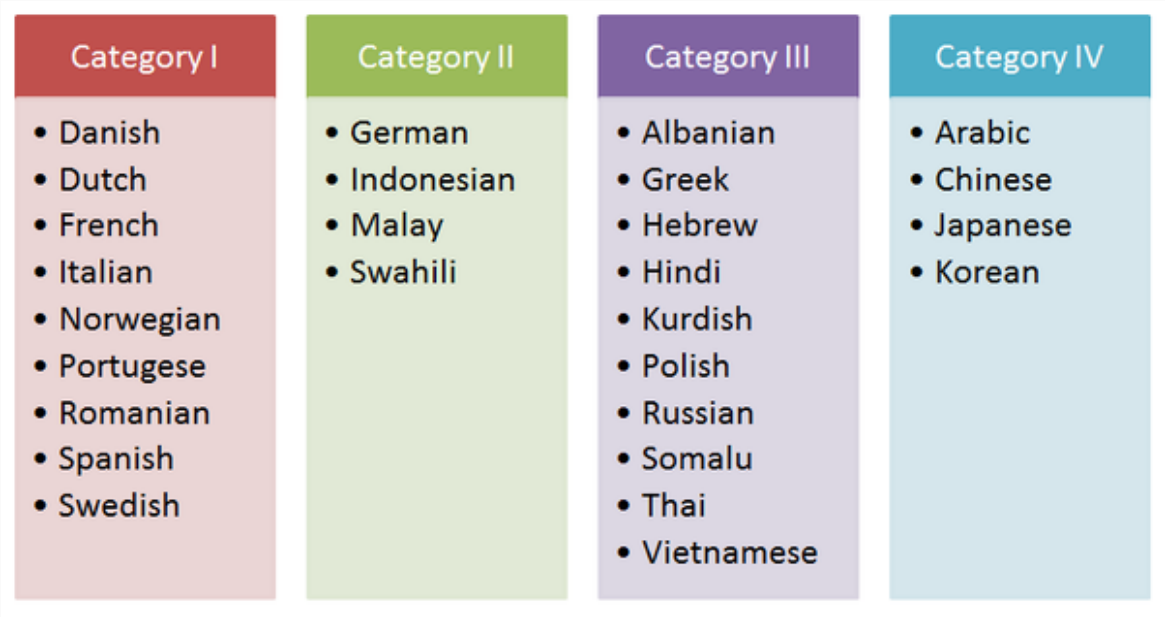
operators, types, and data structures

STAT 198/298 Fall 2020

Send your phone here

Or send a browser to `slido.com`, event `#Z837`.

Learning a second language



Learn Python to learn R.

Poll

We can create the object `a` in R:

```
a <- 1
```

What line of code will augment `a` by 1? (increase its value by 1)

```
a <- a + 1
a
```

```
## [1] 2
```

Operators: what is it?

Take whatever object is on either side of the operator, perform an operation on it, and return the result...

Sounds like a *function* with two arguments.

```
`+`(a, 1)
```

```
## [1] 3
```

Operators: Assignment

Python has *update* operators to make common assignment tasks more streamlined.

```
r.a += 1
r.a
```

```
## 3.0
```

For any operator `#`, the expression `a #= b` is equivalent to `a = a # b`.

Poll

What will this code return?

```
a = 2
b = 3
a -= b
a
```

```
## -1
```

Poll

What will this code return?

```
a = 2
b = 3
a **= b
a
```

```
## 8
```

Operators: Comparison

Comparisons of two objects that yeilds `True` or `False`.

Operation	Description
`a == b`	a equal to b
`a != b`	a not equal to b
`a < b`	a less than b
`a > b`	a greater than b
`a <= b`	a less than or equal to b
`a >= b`	a greater than or equal to b

Operators: Comparison

Notes

- Boolean values: `True` and `False` and nothing else
- Group operations with `(` and `)`
- Validity of comparisons depends on object type

Operators: Boolean

Operations that compose values of `True` and `False`.

Notes

- Only two operators: `and` and `or`
- Useful in conditionals (if-then)
- *Not* the same as `&` and `|` (bitwise operations)

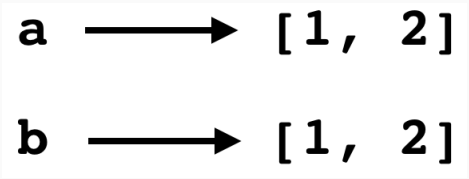
Operators: Sets

```
code
```

Operators: Sets

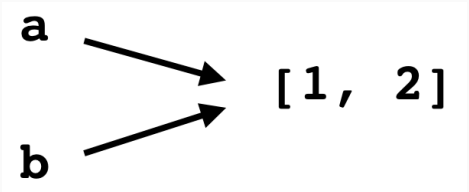
Equality

Two variables are *equal* if they point to two objects that have the same value.



Identity

Two variables are *equal* if they point to the same object.



Poll

```
a <- c(1, 3, 5)
b <- 3
```

Write the code to check to see if `b` is an element of set `a`.

Poll

```
a <- c(1, 3, 5)
b <- 3
```

Write the code to check to see if `b` is an element of set `a`.

```
b %in% a
```

```
## [1] TRUE
```

Operators: Membership

`b in a` checks if `b` is in `a`

```
3 in [1, 3, 5]
```

```
## True
```

`b not in a` checks if `b` is not in `a`

```
3 not in [1, 3, 5]
```

```
## False
```

Check for Q & A

Types

The most basic form of how a piece of data can be stored.

- Integer
- Floating-point number
- String
- Boolean (logical)
- [Complex]
- [NoneType]

Types

Notes

- *Integers* are precise numbers
- *Floats* are approximate fractional numbers, so only check for approximate equality.
- *Strings* allow some arithmetic operations, direct indexing.

Data Structures

Data structures are *compound types* that act as containers for simple types. The ones built into Python 3:

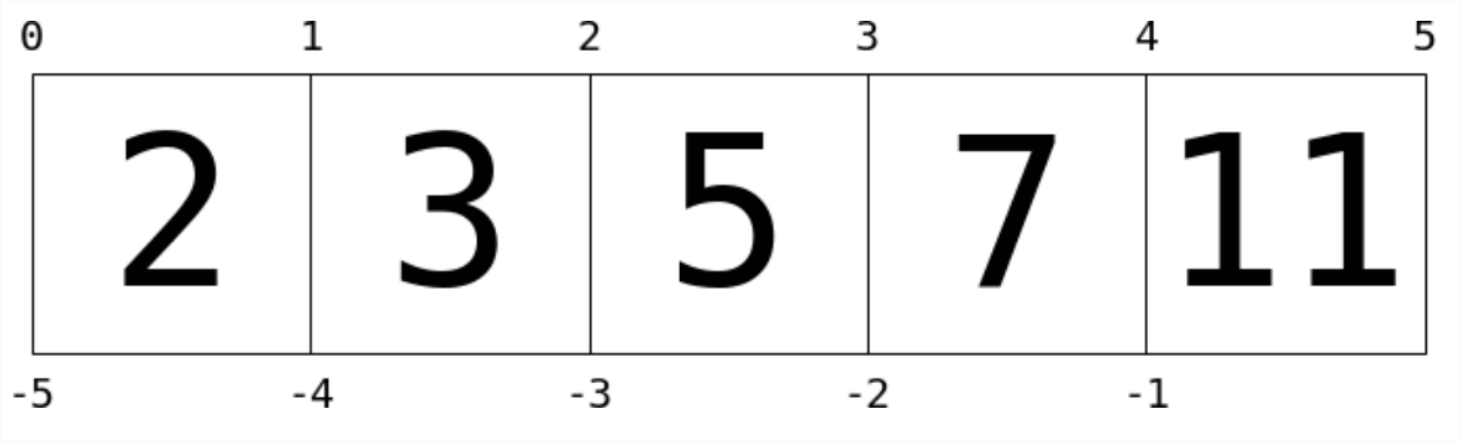
- List
- Tuple
- Dictionary
- Set

When working with a data structure, ask:

1. Is it ordered? (index by integer)
2. Is it heterogenous? (different types)
3. Is it mutable? (change elements)

Code

List indexing



code

Data Structures Summary

Type Name	Example	Description
`list`	`[1, 2, 3]`	Ordered collection
`tuple`	`(1, 2, 3)`	Immutable ordered collection
`dict`	`{'a':1, 'b':2, 'c':3}`	Unordered (key,value) mapping
`set`	`{1, 2, 3}`	Unordered collection of unique values

Assignments

Homework 2

Posted end of the day today, due Friday 8 pm

Lab 2

Posted end of the day today, due Sunday 8 pm