PYTHON - DATA STRUCTURES

Bendik Egenes Dyrli

UiA University of Agder

DATA STRUCTURES

```
{}

[]

()

set()
```

OVERVIEW

1. Data structures
2. Lists
3. Del
4. Sets
5. Dictionary
6. Looping Techniques
7. ...code?

DATA STRUCTURES

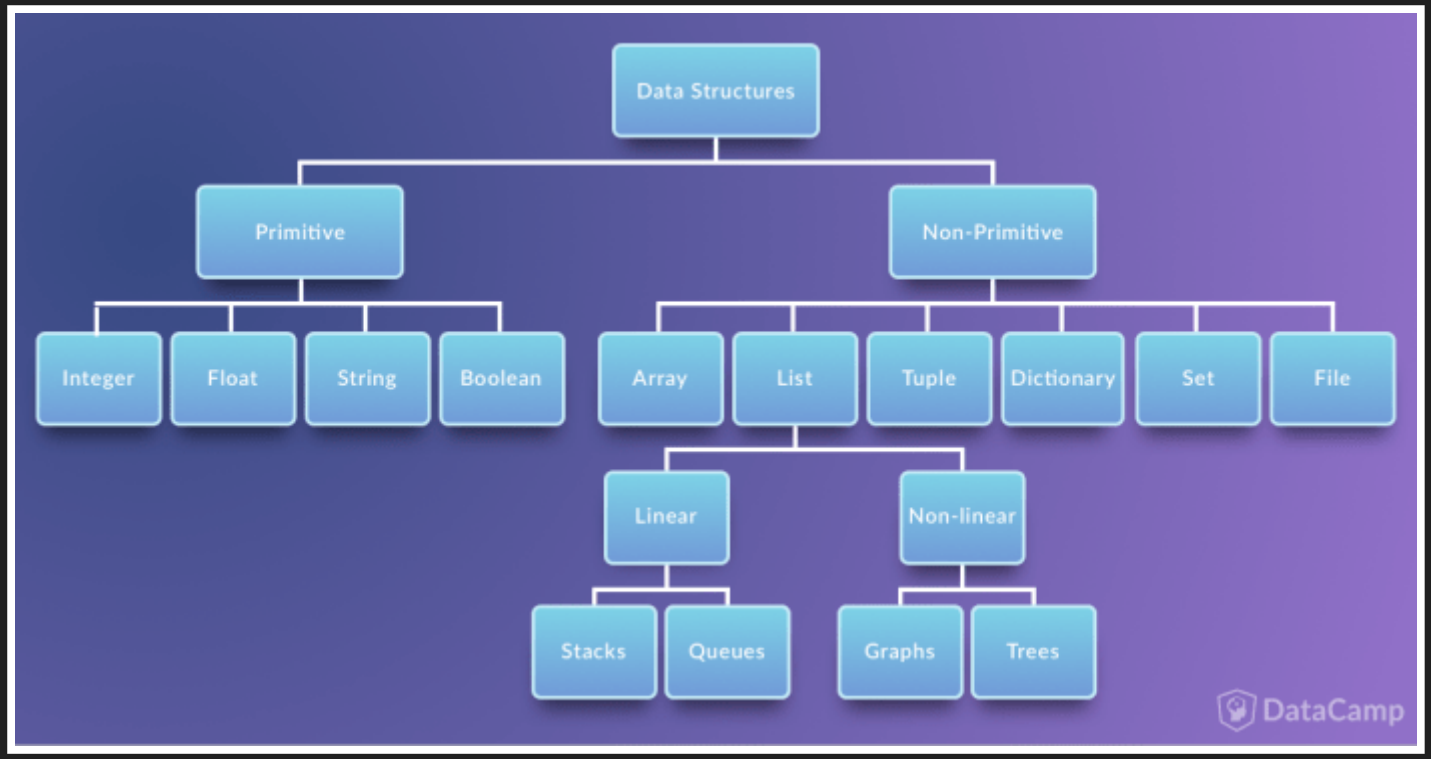

image source: datacamp.com

LISTS

a way of storing data in python

```
my_list = []
```

KEYWORDS

USINGS LISTS AS STACKS

- LIFO - Last-in First-Out

```
stack = [13,21,34,55,89,144]

stack.append(233)
stack.append(377)

print (stack) # we now have two new numbers

print(stack.pop()) # pops the last item from the list
print(stack.pop())

print (stack)
```

LIST COMPREHENSIONS

A more concise way to create a list

```
square = []
for x in range(10):
    square.append(x**2)

print (square)
```

LIST COMPREHENSIONS

```
squares = list(map(lambda x: x**2, range(10)))
```

or, equivalently:

```
squares = [x**2 for x in range(10)]
```

LIST COMPREHENSIONS

```
[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]

[fixtures for fixtures in range(100, 106)]
```

10 MINUTES TASK

Find out how you would write this without the use of list comprehensions

```
[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]
```

LIST COMPREHENSIONS

```
[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]
```

```
combinatorics = []
for x in [1,2,3]:
    for y in [3,1,4]:
        if x != y:
            combinatorics.append((x,y))

print (combinatorics)
```

DEL

The way of deleting an item or clearing a list in python

```
my_list = [1,2,3,4,5] # initialize list with values
print (my_list)

del [0] # delete the value stored in the first index in the li
print (my_list)

del my_list # this will delete everything in the list
print (my_list)
```

SET

```
my_set = set("interdimensional cable")
```

- Unordered collection
- No duplicate elements
- Supports mathematical operations such as
  - union
  - intersection
  - difference
  - symetric difference

SET

```
A = set("interdimensional cable")
B = set("earth cable")

# Set operations

print (A - B) # letters in A but not in B

print (A | B) # letters in A or B or both

print (A & B) # letters in both A and B

print (A ^ B) # letters in a or b but not both
```

DICTIONARY

```
my_dict = {'Rick':-1, 'Morty':137, 'Jerry':'0'}
```

- Dictionaries are awesome !
- can be seen as a key, value store
- keys need to be unique

DICTIONARY

```
state = {'Rick':'Lab', 'Morty':'School', 'Jerry':'Don't know}
print (state)

state['Beth'] = 'Surgery'
print (state)

del state['Jerry']
state['Summer'] = 'Home'

print (list(state)) # prints the list "version" of the dict
print (sorted(state)) # sorts the dictionary

print('Jerry' in state)
print('Rick' in state)
```

DICT COMPREHENSIONS

```
print ({x: x**2 for x in (2,4,6,8)})
```

LOOPING TECHNIQUES

LOOPING LISTS

```
my_list = [1,2,3,4,5,6,7,8,9]

for item in my_list:
    print (item)
```

LOOPING DICTS

```
esolangs = {'brainfuck':'brain damage',
            'lolcode':'lol',
            'unlambda':'needs more lambda'}

for key, value in esolangs.items():
    print (f"Key: {key}\nValue: {value}\n")
```

LOOPING SEQUENCES

```
for index, value in enumerate(['paper',
                               'scissor',
                               'rock',
                               'lizard',
                               'spock'])

    print (f"index: {key}\nValue: {value}\n")
```

LOOPING OVER TWO SEQUENCES AT THE SAME TIME

```
questions = ['name', 'quest', 'favorite color']
answers = ['lancelot', 'the holy grail', 'blue']

for quest, answer in zip(questions, answers):
    print (f"What is your {quest}? It is {answer}.")
```

LOOPING OVER LIST IN REVERSED ORDER

```
for rev in reversed(range(1,10,2)):
    print (rev)
```

RESOURCES

https://docs.python.org/3/tutorial/datastructures.html