NGS analysis for gene regulation and epigenomics

Jacques Serizay, Cyril Matthey-Doret

January 12, 2020

Day 2

Python crash course for genomics

Variables and functions

- Straight forward syntax, no type declaration
- Indentation used to specify nesting
- Indices start at 0 (unlike R)

```
def print_numbers(n):
  """This functions prints numbers from 0 to n-1"""
  for i in range(n):
    print(i)
```

```
>>> x = 3
>>> print_numbers(x)
0
1
2
```

Text formatting

Embed variables in strings using f-strings:

```
def make_ucsc_range(chrom, start, end):
  """Given a chromosome, start and end coordinates,
  return a string in format chr:start-end
  """
  ucsc_range = f"{chrom}:{start}-{end}"
  return ucsc_range

>>> make_ucsc_range('chr1', 3000, 15000)
'chr1:3000-15000'
```

Basic data structures

Some useful native data structures:

Lists can contain any types of values:

```
>>> my_list = [1, "a", "a"]
>>> for i in my_list: print(i)
1
"a"
"a"
>>> my_list[0]
1
>> my_list[:2]
[1, "a"]
```

Basic data structures

Sets contain unique unordered values and can perform set operations:

```
>>> fruit_set = {'apple', 'grape', 'orange'}
>>> color_set = {'red', 'orange'}
>>> "zebra" in fruit_set
False
>>> fruit_set & color_set
{'orange'}
>>> fruit_set | color_set
{'apple', 'grape', 'orange', 'red'}
```

Basic data structures

Dictionaries allow to map keys to values:

```
>>> legs_dict = {'dog': 4, 'duck': 2}
>>> legs_dict['dog']
4
>>> for animal, legs in legs_dict.items():
>>>   print(f"{animal}s have {legs} legs")
dogs have 4 legs
ducks have 2 legs
```

Arrays

Operations are not vectorized by default in python (unlike R):

```
>>> [1, 2, 3] * 2
[1, 2, 3, 1, 2, 3]
```

The `numpy` package provides multidimensional arrays and vectorized operations:

```
>>> import numpy as np
>>> np.array([1, 2, 3]) * 2
array([2, 4, 6])
>>> np.array([1, 1]) + np.array([10, 20])
array([11, 21])
>>> mult_arr = np.array([[1, 2], [3, 4]])
array([[1, 2],
       [3, 4]])
>>> mult_arr[:, 0]
array([1, 3])
```

Dataframes

- The `pandas` package provides an equivalent to R’s dataframes.
- DataFrames are arrays with named dimensions which can hold different datatypes.

```
>>> import pandas as pd
>>> my_df = pd.DataFrame({
>>>   'animal': ['duck', 'dog', 'snail', 'ant'],
>>>   'legs': [2, 4, 0, 6]})
>>> my_df
  animal  legs
0   duck     2
1    dog     4
2  snail     0
3    ant     6
```

Dataframes

- The `pandas` package provides an equivalent to R’s dataframes.
- DataFrames are arrays with named dimensions which can hold different datatypes.

```
>>> my_df['legs']
0    2
1    4
2    0
3    6
Name: legs, dtype: int64
>>> my_df['legs'].mean()
3.0
```

Selecting in dataframes

Dataframes can be subsetted using two methods:

- `iloc`, which is uses indices
- `loc`, which uses masks

```
>>> my_df.iloc[[1, 3], :]
  animal  legs
1    dog     4
3    ant     6
>>> my_df.iloc[2:, 0]
2    snail
3      ant
Name: animal, dtype: object
```

Selecting in dataframes

Dataframes can be subsetted using two methods:

- `iloc`, which is uses indices
- `loc`, which uses masks

```
>>> my_df.loc[my_df.legs < 4, :]
  animal  legs
0   duck     2
2  snail     0
```

Data structures summary

Native structures:

- No vectorized operations
- Lists: any datatype `[x, ...]`
- Sets: any datatype, unique values `{x, ...}`
- Dictionaries: Key-value map, unique keys, any datatype `{x: y, ...}`

Data structures summary

Numpy arrays Docs:

- Single datatype `np.array([1, 2, 3])`
- Vectorized operations `arr.sum()`
- N-dimensional `np.array([[1, 2, 3], [10, 20, 30]])`

Pandas dataframes Docs:

- 1 data-type per column
- columns selected either with `df['col']` or `df.col`
- subset using `df.loc` (conditions) or `df.iloc` (indices)

In practice

- Genomic tracks are often stored in bed files, which can be read into a dataframe:

```
>>> genes = pd.read_csv('genes.bed', sep='\t', names=['chrom', 'start', 'end'])
>>> genes.head()
Out[34]:
  chrom  start    end
0  chr1   4658   5004
1  chr1  18063  18704
2  chr1  64549  65197
3  chr1  69468  69735
4  chr1  73476  73845
5  chr1  78433  79316
```

- Hi-C matrices can be represented using 2D numpy arrays of floats
- The `cooler` package provides pd.DataFrame and np.array objects