# Introduction to Python

Bingbing Yuan
BaRC Hot Topics
Bioinformatics and Research Computing
Whitehead Institute
Nov. 4th 2021
http://barc.wi.mit.edu/hot_topics/

## JupyterLab

1. Login to notebook.wi.mit.edu

2. Download the exercises to your home folder:

   Click on “New” -> Terminal -> Type “setup_Intro_Python_talk” command

   This will create a folder named as “Intro_to_Python”

3. Go to the Intro_to_Python folder, and click on Introduction_to_Python.ipynb

# About Python

- Object oriented language; easy to read
- Scripting language; quick to write
- Massive community support
  - Biopython
- Whitespaces are important. Python uses indentation, no braces are needed
- We will learn Python 3 in this class. Many libraries in Python 2 are not compatible with Python 3.

Whitehead Institute

BaRC Hot Topics

# Objectives

- Go over the basics of Python 3 to get you started on writing your own code
- Covered in this workshop:
  - Introduce variables
  - Read/Write files
  - Create Function and modules
  - Flow Control: if-else and loops
  - Create a single script

Whitehead Institute

BaRC Hot Topics

# Data types

- Single variable:
  - Number: integer, float
  - String: characters
- Ordered:
  - strings: immutable; surrounded by quotation marks

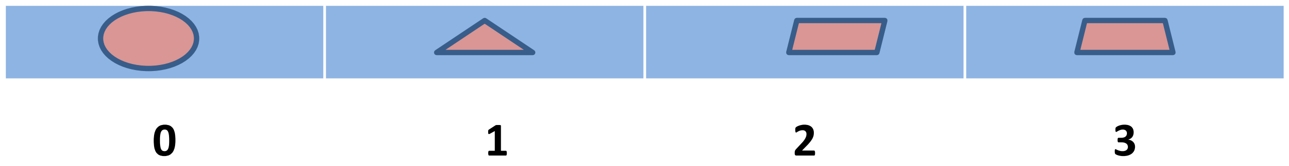
    >>> my_class = “Introduction to Python”
  - lists: mutable; surrounded by square brackets

    >>> my_list = [ 1, 4, 6 ]
  - tuples: immutable; surrounded by parentheses

    >>> my_tuple = ( 1, 4, 6 )
- dictionary:
  - mutable;  surrounded by curly brackets
    - Key is immutable
    - >>> codon = { 'CUU': 'leu', 'UCU':'ser', 'AAA':'lys‘ }
- Use function “type” to find out a variable type

# Common Properties for Sequences strings, lists, tuples

- seq=



- Index
- Slice
- Conditional tests
- Concatenation
- Length, maximum, minimum, etc.

```
>>> len(seq)
>>> 4
```

Whitehead Institute

BaRC Hot Topics

# Index for strings, lists, tuples

– begin with 0

```
>>> seq = “ATGCT”
>>> seq[0] # A
>>> tuple1 = ( 5,6,7 )
>>> tuple1[0] # 5
>>> list1 = [“Tgfb1”, “Bmp4”, “Cdh1”]
>>> list1[1] # Bmp4
```

– Access from right with negative number

```
>>> seq[-1] # T
>>> tuple1[-2] # 6
>>> list1[-1] # Cdh1
```

WHITEHEAD INSTITUTE

BaRC Hot Topics

# slice for strings, lists, tuples

- slicing: [start:stop:step]
  - Stopping value not included
  - Omit start: start from first position
  - Omit stop: go to the last position
  - step default is 1, negative number counting backward

```
>>> seq = "BaRCHotTopics“
>>> len(seq) # 13
>>> seq[0:4] # BaRC
>>> seq[:4] # BaRC
>>> seq[7:] # Topics
>>> seq[0:4:2] # BR
>>> seq[::-1] # scipoTtoHCRaB
```

# Conditional tests for strings, lists, tuples

```
>>> expr = (20, 45, 60)
>>> 55 in expr
False
>>> 60 in expr
True
>>> sum(expr) > 100
True

>>> myseq = "ACTTA"
>>> "G" in myseq
False
>>> myseq.lower() == "actta"
True
```

Conditional tests commonly used inside if/else work flow

# Concatenation for strings, lists, tuples

Concatenate sequences from same type with “+” sign

```
>>> expr1 = (10, 20, 30)
>>> expr2 = (15, 25, 35)
>>> expr1and2 = expr1 + expr2
>>> expr1and2
(10, 20, 30, 15, 25, 35)

>>> seq1 = "ACT"
>>> seq2 = "TGC"
>>> seq1and2 = seq1+seq2
>>> seq1and2
'ACTTGC'
```

Whitehead Institute

BaRC Hot Topics

# Other functions for strings, lists, tuples

- len(): amount of elements

```
>>>List1 = [ "A", "C", "T", "G" ]
>>>print (len(List1))
4
```

- max() and min():

```
>>> expr = (5, 10, 200)
>>> max(expr)
200
```

Whitehead Institute

BaRC Hot Topics

# Dictionaries

- Dictionaries are lookup tables
- It includes key:value pair, and surrounded by curly brackets {}

```
>>> codon = {'CUU': 'leu', 'UCU':'ser', 'AAA':'lys'}
```

- Mutable; unordered data types
- Keys:
  - Unique
  - Immutable
  - Strings, tuples, numbers
- Values:
  - Duplication is allowed

| **type** | **string** | **list** | **tuple** | **dictionary** |
|---|---|---|---|---|
| example | "AACTGC" | [1, 21, 48] | (1, "ATG", 4, 8) | { 'CUU': 'leu', 'UCU':'ser'} |
| mutable | No | Yes | No | Yes* |
| Ordered** | Yes | Yes | Yes | No |

*Key is not mutable
** index and slice are applicable

# JupyterLab

1. Login to notebook.wi.mit.edu

2. Download the exercises to your home folder:

   Click on "New" -> Terminal -> Type "setup_Intro_Python_talk" command

   This will create a folder named as "Intro_to_Python"

3. Go to the Intro_to_Python folder, and click on Introduction_to_Python.ipynb

4. Do exercises: 1 to 8

# Code Sharing and Reuse

Advantages:

Re-usability and readability, easy debug, improve performance

Includes:

a. Functions
b. Modules
c. Packages
d. Classes
e. Objects

# Functions:

- Create function

  *def* functionName(arg1, arg2, …):

  """ your description (optional)"""

  function Code …

  return DATA

- Return types:
  - Single (None, string, etc.)
  - Multiple

```
def calculate_GC_percentage(seq):
   """Calculate GC percentage of a nucleotide sequence. """
   # count the number of C
   C = seq.upper().count("C")

   # count the number of G
   G = seq.upper().count("G")

   # calculate gc%
   # use len to get the total number of nts
   gc_per = 100*(C+G)/len(seq)

   # only keep two digits after decimal
   gc_per = round (gc_per, 2)

   # return gc percentage
   return(gc_per)
```

```
# call a function:
calculate_GC_percentage(“actgtgg”)
```

WHITEHEAD INSTITUTE

## Modules/Packages

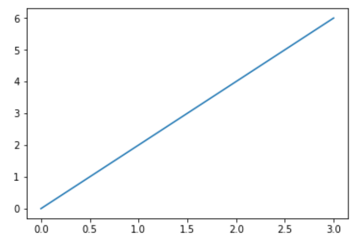
- Module is a file with functions, constants and objects.
- packages is group of modules
  - Should include __init__.py

```
>>> import math
>>> math.pi
3.141592653589793

>>> from math import pi
>>> pi
3.141592653589793

>>> import os
>>> os.getcwd()
/nfs/BaRC_training/Python/Python_Part1

>>> import matplotlib.pyplot as plt
>>> plt.plot([0,2,4,6])
>>> plt.show()
```



WHITEHEAD INSTITUTE

BaRC Hot Topics

# Open/Write files

## Function: open

1. Open file:
   - Two parameters:
     1) File name
     2) Opening mode:
        - r : read (default)
        - w : write
        - a: append
2. Read the file:
   1) read() : read file
   2) readline(): read one line
   3) readlines(): return a list with lines

   *1) and 3): not for large files*

3. Close the file with close()

## Examples:

```
>>>fh = open (“foo.txt”)
>>>print (fh.readline())
>>>fh.close()
```

- Write to file

```
>>> fh = open("out.txt", "w")
>>> fh.write("Python is fun!")
>>> fh.close()
```

- Another way with “with open”
  - No need to close the file

```
>>>with open(“foo.txt") as fh:
>>>    lines=fh.readlines()
```

WHITEHEAD INSTITUTE

BaRC Hot Topics

# Flow Control: If-Else

- *if* expression1:

        Block1
    *elif* expression2:

        block2
    *else*:

        block3

```
>>>WT = 100
>>> KO = 56
>>> if WT > KO:
>>>     print("WT is greater than KO")
>>> elif WT == KO:
>>>     print("WT and KO are equal")
>>> else:
>>>     print("KO is greater than WT")
```

- Comparisons:
  - logical conditions:

    x<y  x>y  x==y  x!=y
  - Use ***and*** or ***or*** combine conditional statements

    *if* x>y *and* x>z:

    *if* x>y *or* x>z:

Whitehead Institute

BaRC Hot Topics

## Loops

**for loop**

for VAR in ITERABLE:
    BLOCK

ITERABLE:

lists, tuples, strings, dictionaries, files

```
>>>seq = "ACTG"
>>>for base in seq:
>>>       print (base)
.
```

**while loop**

while EXPRESSION:
    BLOCK

```
>>>a = 10
>>>while a<30:
>>>    print (a+1)
>>>    a=a+10
```

Avoid infinite loop

## Create a script

Python file name:  ends with .py

The file name for this script is get_GC_percentage.py

Run the script:

The script asks for an input fasta file, and prints out  gc% of the sequence inside the input file.

```
$ ./get_GC_percentage.py
What is your input sequence file?sample_seq.fa
52.68
```

```
#!/usr/bin/python

input_file = input ( "What is your input sequence file?")


def calculate_GC_percentage(seq):
    """Calculate GC percentage of a nucleotide sequence. """
    # count the number of C
    C = seq.upper().count("C")
    # count the number of G
    G = seq.upper().count("G")

    # calculate gc%
    # use len to get the total number of nts
    # only keep two digits after decimal
    return(round(100*(C+G)/len(seq), 2))


# set an empty string to hold sequences
seq = ""
# read a sequence file

with open(input_file) as fh:
    content = fh.readlines()

    # concatenate the sequences
    seq = "".join(content[1:])
    # delete end of line character
    seq = seq.replace("\n", "")

# calculate the GC percentage
print ( calculate_GC_percentage(seq) )
```

WHITEHEAD INSTITUTE

BaRC Hot Topics

# References

- Python Official website:
  - https://www.python.org/
- Python Tutorials from programiz website
  - https://www.programiz.com/python-programming/#tutorial
- GeeksforGeeks
  - https://www.geeksforgeeks.org/
- Software Carpentry:
  - https://swcarpentry.github.io/python-novice-gapminder/

- Cheat sheets:
  - https://www.docsity.com/it/beginners-python-cheat-sheet-pcc-all/653739/
  - https://tmont.es/images/sheet-of-python-v1.pdf

# JupyterLab

## Introduction_to_Python.ipynb:

Do exercises: 9 to 12

## Coming up next:

*Introduction to Python part 2: Python for Analysis*

*Thur. Nov. 18, 2021*