Introduction to Python (Part I)

learnpython.org

1

Introduction

Course Objectives

- Give biologists comfort and confidence with computation
- Basic ability to learn and practice computational skills in your future research
- Basic Python Programming
  - Learn how to use IDLE to execute Python commands and use Wing 101 to write Python scripts
  - Create and manipulate variables
  - Use loops and conditional statements in a Python program

2

Introduction

This course is….

- A formal introduction to computer science but it is

This course is NOT….

- A systematic course in bioinformatics
- A substitute for understanding how the statistics/programs/analyses you use work

3

Introduction

What you DON'T need to succeed

- Deep math background
- Previous programming experience

What you DO need to succeed

- Logical and organized way of thinking (or at least organized notes for your future self!)
- Determination
- Practice

4

Basic Python

Monday

- Install Python and Pycharm
- Interactive and batch mode
- Docstrings and comments
- Learn the basics of Python syntax
  - Variables
  - Data types
  - Strings

Tuesday

- Review the basics of Python syntax
  - Variables
  - Data types
  - Strings
- Gene Sequence Exercise

You don't need to understand what these mean yet!

5

Let's get started!

6

Introduction

Am I cut out for Programming?

Programming Skills

- Logical thinking
- Detail oriented
- Able to solve problems based on incorrect results
- Biologists are natural programmers!

7

Introduction

What is a program (script)?

A Detailed Set of Instructions for How to Do Something

- Definitions/symbols (assignments)
  - pi = 3.1416
- Actions
  - Area = pi x radius2
- Loops
  - Repeat 2 times
  - Repeat until …
- Conditional
  - If (something) do (something)
- Results (Output)

8

Introduction

What is a program (script)?

Computers are Like Very Hard Working but Very Stupid Lab Helpers

» Instructions must be exact – computers are quite happy to do the wrong thing over and over

» All possible alternatives must be covered – when undefined situations occur computers either

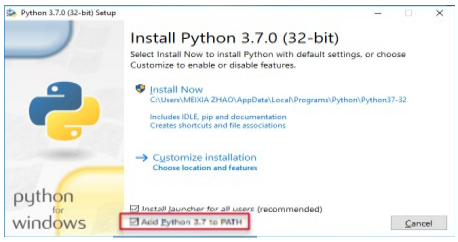
- Do the wrong thing
- Stop and wait (forever)
- Fail catastrophically (starved because they never finished and missed dinner)

9

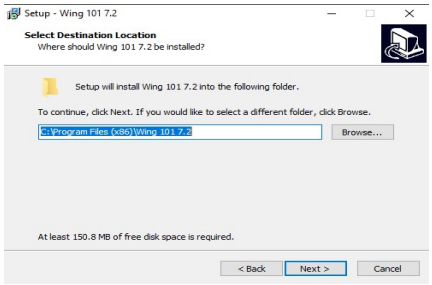
Basic Python

Install Python and Wing 101

» Install Python (must be version 3)



» Install Wing 101



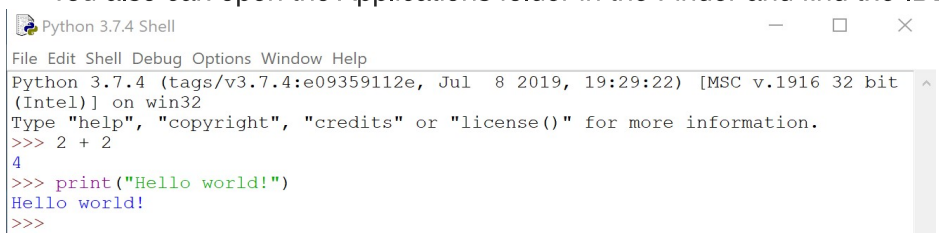
If run into problems:
https://wingware.com/get&prod=wing-101&target=pub/wing-101/7.2.3.0/wing-101-7.2.3.0.exe

10

Basic Python

IDLE (Integrated Development and Learning Environment)

- IDLE is a program used to write and execute Python code
- For windows:
  - Navigate to the Start Menu in the bottom left hand corner of the screen
  - Type in the search bar IDLE and click on the application IDLE
- For Mac:
  - Open the Finder and type in IDLE in the search menu
  - You also can open the Applications folder in the Finder and find the IDLE program



11

Basic python

Python Programming/Interactive mode

Two Basic Python Modes

- **Interactive mode (Front-end)**
- **Batch Mode (a script)**

12

Basic python

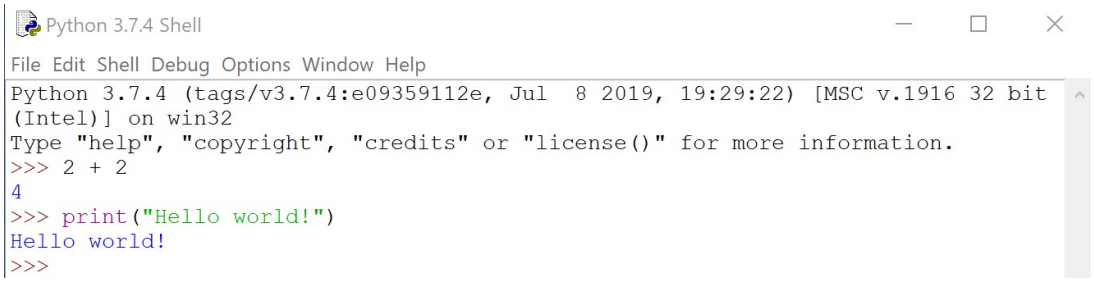
Python Programming/Interactive mode

Two Basic Python Modes

» **Interactive mode (Front-end)**

- Interactive mode is a command line shell which gives immediate feedback for each statement

```
>>> print ('Hello World!')
Hello World!
>>> 1 + 1
2
```



13

Basic python

Python print( ) function

» The print( ) function prints the given object to the standard output device (screen) or to the text stream file.

» By default it prints all inputs separated with a whitespace, but you can change the separator with a parameter named sep.

» print (objects, sep=' ', end = '\n')

```
>>> print('Hello', 'World!', sep='-')
Hello-World!
>>> print('Hello', 'World!', 'World!', sep='-', end='\n')
Hello-World!-World!
```

14

Basic python

What is a program (script)?

Computers are Like Very Hard Working but Very Stupid Lab Helpers

» Instructions must be **exact** – computers are quite happy to do the wrong thing over and over

» **All possible alternatives must be covered** – when undefined situations occur computers either

- Do the wrong thing
- Stop and wait (forever)
- Fail catastrophically (starved because they never finished and missed dinner)

15

Basic python

Python Programming/Batch mode

Two Basic Python Modes

» **Batch Mode (script)**

- The Python Shell is convenient for executing a few simple commands. If you need to give the computer more complex instructions, it's a good idea to write a script. A Python **script** is a set of instructions that you can easily edit and run.
- When a program is executed from such a text file, rather than line by line in an interactive interpreter, it is called batch mode.

1. Go to **File >> New File** to create a new script.
2. In your script, write **print("Hello!")**.
3. Go to **File >> Save As…** to save your script. Choose a location and a name. Be sure to add **.py** to the end of the file name. This will save the file as a Python script.
4. Go to **Run >> Run Module** to run your script. The Python shell will reopen and display **Hello!**

16

Basic python

Python Programming/Batch mode

Two Basic Python Modes

» **Batch Mode (script)**

- These are regular text files usually with the ".py" extension. They are also known as python scripts.
- These files can be generated with any standard text editor, such as Pycharm, Notepad++, EditPlus, **Wing 101** etc.

17

Basic Python

First program

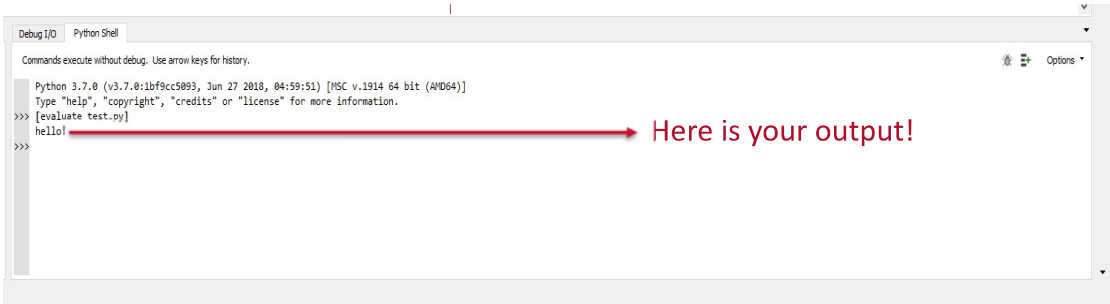
Create a New Project

» Click "File" --- "New" (Or click on the button)

» Write your code (Eg. **print("hello!")**)

» Click "File" --- "Save as" --- Name your file

» Click on the button to run your program



18

Basic Python

Docstrings and Comments

Programming Best Practices: Commenting and Clarity

» The top priority is making your program easy to understand
- Makes it possible to verify correctness
- Simplifies modifying and updating
- Promotes reuse

» Requirements
- Docstrings – Block quotes within triple “”” are called docstrings. They are used in the interpreter to provide help. For programs in this course
  - ✓ You must supply a header as a docstring. The header must include
  - ✓ Purpose of the program
  - ✓ Your name
  - ✓ Date
- Comments –comments are introduced by #. Use comments to describe the steps in your program.

19

Basic Python

Docstrings and Comments

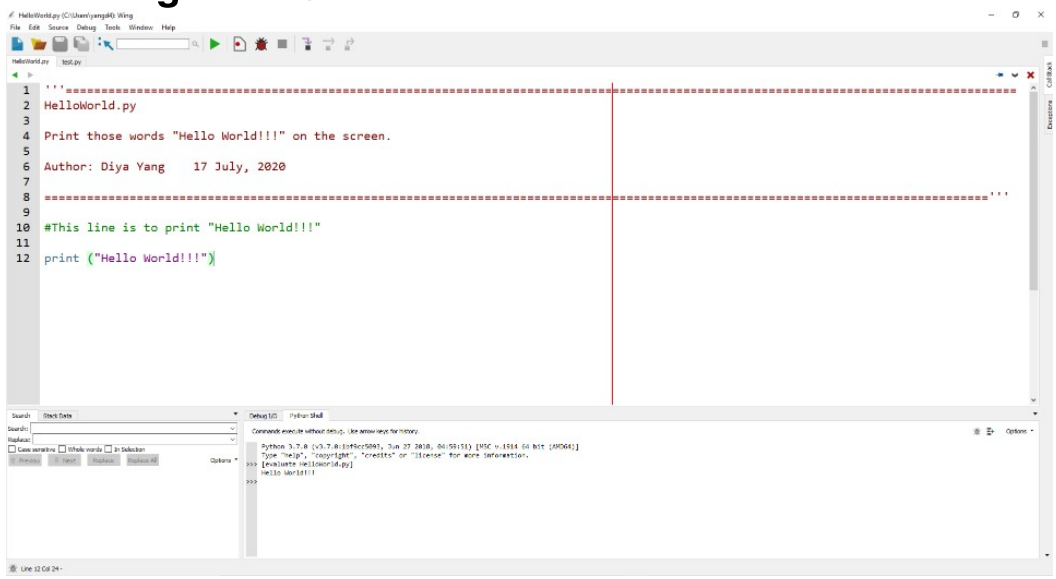
Programming Best Practices: Commenting and Clarity

» The header and comments should be written before you start writing any code
- Start with the header
- Add comments describing the actions
- Only then start programming

20

Basic Python

Docstrings and Comments

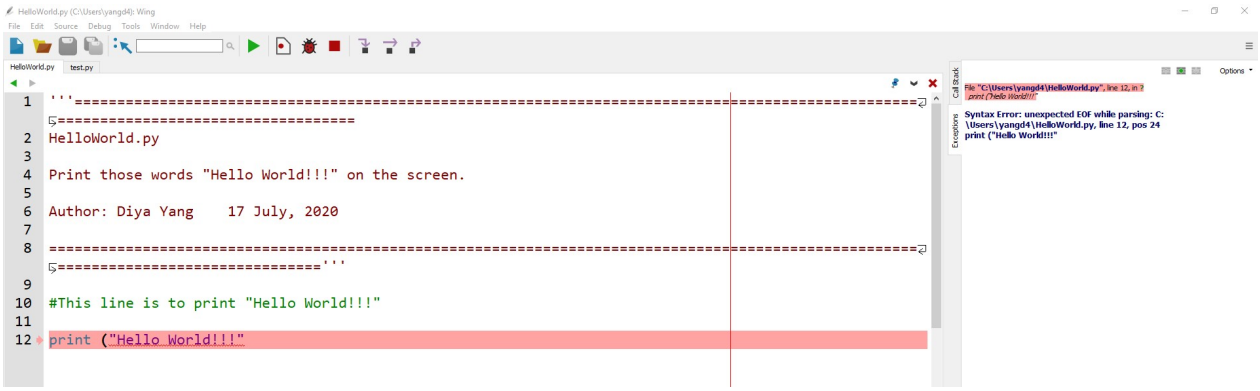


21

Basic Python

Debugging Option

» Click on the button if your code doesn't run
- Hints and comments would be shown on the right side.



22

Basic Python

Variables

Define a Variable

» A variable is a value that you can change and access throughout your script (*variables are just names*)

» Variable names can only contain these characters:
- Lowercase letters (a through z)
- Uppercase letters (A through Z)
- Digits (0 through 9)
- Underscore (_)

» **Names cannot begin with a digit**
- 1
- 1a
- 1_

23

Basic Python

Variables

Define a Variable

» Pick variable names that are distinctive and memorable
- course = 'Python 101'
- dnaseq = 'ATGCCTGAG'
- num = 100

» Don't use any of these for variable names, because they are Python's reserved words!

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

24

Basic Python

Basic variable types

- **Integer numbers**
  - 100
- **Floating point numbers**
  - 3.1415926
- **Strings**
- A sequence of zero or more characters that are enclosed within either a pair of single quotes or a pair of double quotes. String is immutable.
  - A string can include letters, digits, punctuation, white spaces and other characters (Eg. dnaseq = 'ANTGCTG')
- **Boolean**
  - True
  - False
- There is also a special no-value value called **None** (When the value is None, nothing is printed, since None means "nothing")

```
>>> print ('single quote')
single quote
>>> print (''double quote'')
double quote
>>> print (''''triple single quote''')
triple single quote
>>> print ("""triple double quote""")
triple double quote
```

25

Basic Python

Operators

An operator is a symbol that indicates a calculation using one or more operands.

Numeric Operator

Operator	Description	Example
+ Addition	Adds values on either side of the operator	>>> 5 + 2 7
- Subtraction	Subtracts right hand operand from left hand operand	>>> 5 – 2 3
* Multiplication	Multiplies values on either side of the operator	>>> 5 * 2 10
/ Division	Divides left hand operand by right hand operand	>>> 5 / 2 2.5
// Integer Division	The whole number smaller than the floating point result	>>> 5 // 2 2
% Modulus	Integer remainder after b/a	>>> 5 % 2 1
** Exponent	Performs exponential (power) calculation on operators	>>> 5 ** 2 25

26

Basic Python

Operators

Numeric Operator

Operator	Description	Example
=	Assigns values from right side expression to left side operand	c = a + b assigns c the values of a + b
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a c += 1 is equivalent to c = c + 1
-= Subtract AND	It subtracts right operand to the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Subtract AND	It multiplies right operand to the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand to the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
//= Integer Division AND	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
** Exponent	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a

27

Basic Python

Operators

Box Model

» Beginning programmers are often puzzled by syntax such as

```
b = 2
b = 4 + b
print (b)
```

BEFORE you run this,
what do you think the answer is?

28

Basic Python

Operators

Box Model

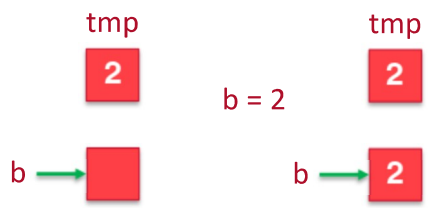
» Beginning programmers are often puzzled by syntax such as

b = 2

b = 4 + b

» Think of variables as boxes

- a box is a location in memory
- the variable name is a label on the box, or the name of the box if you like
- b = 2 copies a value from a temporary location into the box



29

Basic Python

Operators

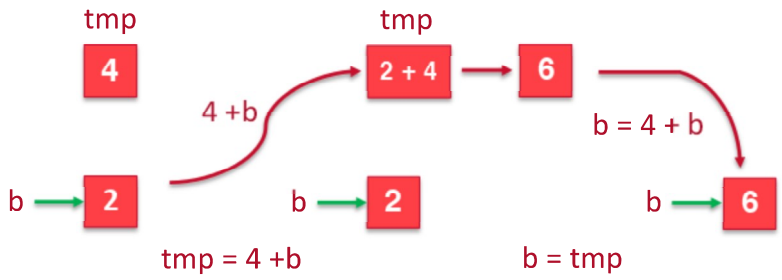
Box Model

» Think of variables as boxes

b = 2

b = 4 + b

- 4 is loaded in a temporary space
- b is added to 2 in the temporary space
- the result is copied back into b



30

Basic Python

Operators

Box Model

- The box model is conceptual
- You can check where the variable is stored using the id() function

```
>>> n = 6
>>> id (n)
1349663056
>>> n += 4
>>> n
10
>>> id (n)
1349663120
```

- When I change the value of n, its memory address changes
  - 4 is loaded in a temporary space

```
>>> x = 2
>>> y = x
>>> y += 1
```

x =? x = 2 Does not change x!!!
y =? y = 3

31

Basic Python

Convert Data Types

Type Conversions with int(), float() and str()

- **int()** – Convert other data types to integer
  - Convert Boolean "True" and "False" to integer 1 and 0
  - Convert a floating-point number to an integer just lops off everything after the decimal point
- **float()** – Covert a string or integer to a floating point number
  - Convert an integer to a float just makes it the proud possessor of a decimal point
  - Convert a string containing characters to a real float
- **str()** -- Convert other data types to strings

```
>>> 'DNA' + 'RNA'
'DNARNA'
>>> 5 + 'DNA'
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> str(5) + 'DNA'
'5DNA'
```

32

Basic Python

Strings

String Index

- Extract a character with [ ]
- The first character is at position 0, not 1 (Remember!!!)

```
>>> dnaseq = 'ANTGCTG'
>>> dnaseq[0]
A
>>> dnaseq[1]
N
```

- The index can also be negative. The last character is at index -1

```
                0 1 2 3 4 5 6
>>> dnaseq = 'ANTGCTG'
>>> dnaseq[-1]  -7 -6 -5 -4 -3 -2 -1
G
>>> dnaseq[-2]
T
```

33

Basic Python

Strings

String Slicing ***[start:end:step]***

- Slicing extracts a series of characters from a string
- The character positions of a slice are specified by two or three integers inside square brackets, separated by colons

```
                0 1 2 3 4 5 6
>>> dnaseq = 'ANTGCTG'
                -7 -6 -5 -4 -3 -2 -1
>>> dnaseq[1:4]
NTG
```

The second index indicates where the slice ends. The character at that position is not included in the slice!!!

The first index indicates the position of the first character to be extracted.

34

Basic python

Strings

String Slicing ***[start:end:step]***

» A third number indicates a number of characters to skip, known as a *step*

```
0 1 2 3 4 5 6
>>> dnaseq = 'ANTGCTG'
-7 -6 -5 -4 -3 -2 -1
>>> dnaseq[0:6:2]
ATC
```

» When the third number is omitted, as it often is, the default is 1

» Negative step takes characters in reverse order

```
>>> dnaseq[6:0:-2]
GCT
```

35

Basic python

Strings

String Functions and Methods

» A function is a piece of code written to carry out a specified task, and called by name.

- print() and input()
- int(), float() and str()

» Get length with len()

```
DNASeq = 'ANTGCTG'
DNASeq_length = len(DNASeq)
print(DNASeq_length) → 7
```

36

Basic Python

Strings

String Methods

» **str.count()** – count how many times a particular substring occurs in a string

```
dnaseq = 'ANTGCTG'
print(dnaseq.count('A'))  # 1
```

» **str.replace(old, new, count)** – replace a particular substring occurs in a string

```
dnaseq = 'ANTGCTG'
print(dnaseq.replace('N', 'G'))  # AGTGCTG
```

» **str.lower()** – return a copy of the string with all the cased characters converted to lowercase.

```
dnaseq = 'ANTGCTG'
print(dnaseq.lower())  # antgctg
```

str.upper()

```
dnaseq = 'antgctg'
print(dnaseq.upper())  #ANTGCTG
```

37

Basic Python

In Class Exercise

» dnaseq = "ATgTCtCATTcAAAGCANNNNNATGCGAGTTATGA",

» Write a simple python script to…

- Replace "N" into "G" in the variable dnaseq, and print the sequence.
- Capitalize all the lowercase in the variable dnaseq, and print the sequence.
- Get the length of dnaseq and print the length.
- Calculate the number of "G" and print the number.
- Calculate the number of "G" in the replaced sequences and print the number.

» Hints:

- Write docstrings and comments.
- Use .replace string method.
- Use .upper string method.
- Use len() function.
- Use .count string method.

Expected results

```
ATgTCtCATTcAAAGCAGGGGGATGCGAGTTATGA
ATGTCTCATTCAAAGCANNNNNATGCGAGTTATGA
35
5
10
```

38

Lists

What is a List?

» A list is made from zero or more items, separated by commas, and surrounded by square brackets.

```
>>> empty_list = [ ]
>>> weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
>>> birds = ['emu', 'ostrich', 'cassowary']
>>> first_names = ['Graham', 'John', 'Terry', 'Terry', 'Michael']
```

» Items can be any data type.

```
>>> my_list = [123, 'John', 'Terry', 1.23]
```

39

Lists

How to Access List Elements or Items?

» As with strings, you can extract a single value from a list by specifying its offset.

```
                    0      1      2
>>> sequences = ['AAA', 'TTT', 'GGG']
>>> sequences[0]   -3     -2     -1
'AAA'
>>> sequences[1]
'TTT'
>>> sequences[2]
'GGG'
```

```
>>> sequences[-1]
'GGG'
>>> sequences[-2]
'TTT'
>>> sequences[-3]
 'AAA'
```

40

Lists

Change an Item by [offset]

» **Lists are mutable.**

```
>>> sequences = ['AAA', 'TTT', 'GGG']
>>> sequences[2] = 'CCC'
>>> sequences
['AAA', 'TTT', 'CCC']
```

Strings are immutable

```
>>> DNASeq = 'ANTGCTG'
>>> DNASeq[1]  = 'G'
```

```
>>> sequences = ['AAA', 'TTT', 'GGG']
>>> sequences[3] = 'CCC'
```

IndexError: list assignment index out of range

```
>>> sequences += ['CCC']     (or sequences.append['CCC'])
>>> sequences
['AAA', 'TTT', 'GGG', 'CCC']
```

41

(Part I)

Review Questions!

Here are some sample questions to review what you have learned at Python I course. You can find all the answers in the powerpoint slides. If you need more explanation please don't hesitate to email the course instructors or TAs and remember some of these questions are designed to make you think deeper about the code so it's completely normal to not get them right on the first try. Happy studying!
Suggestion: If you have difficulty with a question write down your train of thought about how the code will run and what it will return. It is a great exercise and it will help when you encounter more complicated pieces of code in the future.

Please choose the best answer for each multiple choice question.

1. Which variable name is NOT legal in python?
A. __file__
B. 5prime_end
C. DNAseq
D. ZzZ

2. Which of the following ways can create a legal docstring in python?
A. """your docstring"""
B. #your docstring
C. '''your docstring'''
D. ###your docstring###

3. What is the output for the following python script?

```
x=2
x+=2
print(x)
```

A. 2
B. 0
C. 4
D. "2"

4. Which line of code produces an error?
A. "7" + "eight"
B. 3 + 4
C. "one" + "2"
D. '5' + 6

5. What is the output for the following python script? (suggestion: run the code in python)

```
DNA="ATGC"
print(DNA[-4::2])
```

A. AT
B. ATGC
C. AG
D. GC

6. What is the output for the following python script?

```
a=[1,2,3,4,5]
print(a[3:0:-1]
```

A. Syntax error
B. [4, 3, 2]
C. [4, 3]
D. [4, 3, 2, 1]

7. (T/F) python is not case sensitive.

Section2

Write the output of the following codes.

8. Fill in the blank with the output of this code.

```
>>> 1 + 2 + 3 + 4.0 + 5
```

9. Fill in the blank with the output of this code.

```
print(12 + 34)
```

10. Fill in the blank with the output of this code.

```
print("12" + "34")
```

11. What is the result of this code?

```
DNAseq = 'AACCGGTT'
DNAseq2 = DNAseq[::-2]
print(DNAseq2)
```

12. What does the following code print?

```
DNASeq = 'ATgCtTcNNNTGA'
DNASeq = DNASeq.replace('N', 'G')
DNASeq = DNASeq.upper()
print(DNASeq)
```

Notes