

Name: _____ **Section:** 1 2

1 = Mutchler, 2nd-3rd periods. **2** = Mutchler, 4th-5th periods.

Use this quiz to help make sure you understand the videos/reading. **Answer all questions.** Make additional notes as desired. **Not sure of an answer?** Ask your instructor to explain in class and revise as needed then.

Throughout, where you are asked to “underline your choice”, you can underline or circle it (whichever you prefer).

Video: **Introduction** [3:14 minutes]

1. What is the relationship between the live coding example on the video and the exercise you will do in class during Session 2?

Handout (and associated exercise in a Python console): **Objects, Types, Values and Variables.** As described in the assignment page, work through items 1-20 (page 1 to top of page 8), answering these questions as you go through it.

2. Consider the following statements:

```
x = 54
y = 3.713
z = 'my best friend'
```

- a. What is the **type** of the object to which the variable **x** refers?
What is the **value** of that object?
 - b. What is the **type** of the object to which the variable **y** refers?
What is the **value** of that object?
 - c. What is the **type** of the object to which the variable **z** refers?
What is the **value** of that object?
3. Write a statement that assigns the variable `my_friend` the value `'Betty Bop'`
 4. Suppose that you run the following **one-line** program. What happens?

```
print('hello')
```
 5. Suppose that you run the following **one-line** program. What happens?

```
print(hello)
```

6. What do the following expressions evaluate to?

`3 * (4 + 1)`

`3 * ('hi' + 'bye')`

7. What is the value of `y` after the following set of statements executes?

`y = 5`

`y = y * 3`

`y = y + 1`

8. Assume that you have a variable `x` that has already been given a numeric value. Assume that you have put `import math` at the top of your program. Write a statement that sets the variable `y` to the sum of the sine of `x` and the cosine of `x`.

Textbook Reading: **Section 2.1 Variables** [pages 29-37]

9. The same variable name can occur on both sides of `=` (the assignment operator).

True **False** (underline your choice)

10. `_2_` is a valid Python variable name. **True** **False** (underline your choice)

11. Valid or not (not wanting to give away the answer to the previous question), why would `_2_` be a bad choice for a variable name?

12. It is okay to use a variable before we give it an initial value. **True** **False** (underline your choice)

13. What is a *magic number*?

Video: **Input-Compute-Output Programs** [8:36 minutes]

The following questions refer to the annotated program shown in the handout.

14. Where does **execution** traditionally begin?

15. What is the:

- first **statement** that executes?
- second **statement** that executes?
- Third **statement** that executes?

16. Where is the `convert_to_celsius` function **called** (i.e., **invoked**, i.e., made to run)?

Where is it **defined** (i.e., where are the statements that execute when the function is called)?

17. Where is a **doc-comment**?

What is the purpose of doc-comments?

18. Where is an **internal comment**?

What is the purpose of internal comments?

19. What keyword marks the beginning of a **function definition**?

20. What notation marks the **body** of the function (that is, how can we tell when one function ends and another starts)?

21. How can you tell the **name** of the function being defined?

22. What is the name of the function that causes this program to pause and wait for the user to type some **input**?

23. What is the name of the function that converts that input from a **string** (i.e., a sequence of characters) to a **floating-point number** (i.e., a number that has a decimal point)?

24. List three **variables** in this program.

25. What symbol **assigns** a value to a variable?

26. What side of the assignment is the assigned variable – the **left side** or the **right side**? (Circle your choice.)

27. What is that name of the function that **prints** things (i.e., displays them on the console)?

Video: **Calling Functions** [3:22 minutes]

28. Consider the function definition shown to the right:

```
def f_to_c(fahrenheit):  
    celsius = (fahrenheit - 32) * (5 / 9)  
    return celsius
```

- What is the **name** of the function?
- How many **parameters** does the function have?

29. List 2 reasons why functions are powerful.

- Reason 1:
- Reason 2:

30. Write 2 statements that together call the above function twice, each time doing a different Fahrenheit-to-Celsius conversion:

31. When the statement

```
c = f_to_c(10.5)
```

is executed (where the **f_to_c** function is as defined above), what are the 4 steps that occur (be concrete in your answer, referring explicitly to this example wherever you can):

- Step 1:
- Step 2:
- Step 3:
- Step 4:

From your **Session01** project: **Examples**

32. In your example file **m2e_hello_and_gooby.py**:

- a. How many times does the word **Ciao** appear in the program's code?
- b. How many times does **Ciao** get printed when you run (execute) the program?
- c. Explain your answer to (b) briefly:

33. Using your example file `m4e_input_compute_output.py` as an example, write statement(s) that together: prompts for an integer from the user, gets a string from the user, converts the string into an integer, and stores the integer in the variable `n`.
- Hint: Instead of using `float` (which converts a string to a floating point number), use `int`.

Textbook Reading: **Section 2.2 Arithmetic** [pages 37-45]

34. Show the Python values of the following expressions:

`17 / 5`

`17 // 5`

`17 % 5`

35. What line of code must be executed before you can execute `y = math.sin(0.357)` ?

36. What is a *round-off error*?

37. What is wrong with this statement? `x=(-b-sqrt(b**2-4*a*c))/(2*a)`

38. If the last character in a line of Python code is backslash (`\`), what is the effect on the program?

Video: **Preview of Session 2 Exercise**[19:39 minutes]

There are no quiz questions for this video. The “quiz” is that in the Session 2 class, you will be asked to do the things that are in this video. If you have seen the video first and heard our explanation, you will most likely find that exercise much easier to do. But there is a lot to do in preparation for Session 2, so watching this video is completely optional.

Reminder: Don't forget to do the TODO's in the `m8` and `m9` files of your **Session01** project (if you did not already do them) and to commit them to your Subversion repository (use the **SVN** pull-down menu, and the **Commit** item). You can do the Commit at the beginning of Session 2 if you prefer.