Week 3: Expressions, Data Types, Sequences

Data 8 Tutoring

# 1 Expressions and Data Types

## Key Concepts

**Function calls**

Done by writing the name of the function and passing in arguments in the parenthesis. Functions can have any number of arguments.

*Example:* min(4,5,6)

**Assignment Statements**

Assign the variable on the left hand side of the equal side to the value of the expression on the right hand side

*Example*. var = max(1,3) first evaluates the right hand side to 3, then assigns it to the variable var

**Arithmetic Operations**

* +, -, \*, and /
* % for remainder
* \*\* for exponentiation
* >,<,==,!=,>=,<= for comparison

**Data Type Conversion Functions**

* str(...)
* int(...)
* float(...)

## Practice Problems

**1.1** Evaluate the following code snippets

**a.** str(8)+str(24)

‘824’

**b.** min(10%2, 10%3, 10%1)

0

**c.** abs(1-(4\*\*2))

15

**d.** int(‘4’)\*6

24

**1.2** Jaylen Brown challenges you to a modified Three-Point Contest. Jaylen shoots 10 shots and his score is equal to the number of baskets he makes. Assume Jaylen makes a number of shots stored in the variable jaylen\_makes. On the other hand, you get 3 tries to shoot 10 shots each, and your score comes from whichever of the 3 tries has the most shots made. Whoever has the highest score wins.

Assume your attempts made in your three tries are stored in try1*,* try2*,* and try3. Assume there are no ties. Write a line of code which returns whether you won the game. (*Hint: this should be* True *or* False)

max(try1,try2,try3) > jaylen\_makes

**1.3** Assume the variable name eight has been assigned to the string ‘8’ . Using only this string, the string methods, arithmetic, and any type conversion functions (*int, str, etc.)*, print the square of 88. You may want to use variable assignments so you don’t have to reuse code.

eighty\_eight = eight + eight print(int(eighty\_eight)\*int(eighty\_eight))

**1.4** Write a line of code that evaluates whether 111\*43 is even.

(111\*43)%2 == 0

# 2 Arrays

## Key Concepts

**Array**

Data type which can hold sequences of data, as long as they all have the same type.

Useful for arithmetic operations, as it allows for mathematical expressions to be applied to all elements to an array at one time.

Useful array methods/functions include:

* *make\_array(...)*
* len(array)
* *np.arange(start, end, step)*
* *arr.item(x)*
* *np.cos, np.log, np.sin, np.sqrt,* etc.

## Practice Problems

**2.1** Using different approaches, write two separate lines of code that evaluate to the first 10 multiples of 3 (starting at 3).

np.arange(1,11,1)\*3, np.arange(3,31,3)

**2.2** Assume shopping is an array of dollar amounts spent in a store (before tax) by 5 different customers. Write a line of code to answer the following questions.

**a.**  What is the total amount spent by the customers?

sum(shopping)

**b.** What was the largest amount spent by a customer?

max(shopping)

**c.**  Did person 2 spend more than person 4? Assume there’s no person 0. Your code should evaluate to either True or False.

shopping.item(1)>shopping.item(3)

**d.** Assume tax is 10 percent on these items. What are the prices spent by the customers after tax?

shopping \* 1.10

**e.**  What was the absolute difference between Person 1’s expenditure before tax, and Person 5’s expenditure after tax?

abs(shopping.item(0) - (shopping\*1.10).item(4))

OR abs(shopping.item(0) - shopping.item(4)\*1.10)

**2.3** Use one line of code to figure out what every number from 1 to 10 to the power of one more than that respective number is. So, the first number in your output should be 1^2 = 1, the second number should be 2^3, and the last number should be 10^11.

np.arange(1,11,1)\*\*np.arange(2,12,1)

**2.4** Assume we have an array of strings called str\_arr

**a.** Find the length of the array.

len(str\_arr)

**b.** Find the length of the third string in this array.

len(str\_array.item(2))

**2.5** Assume we have arrays first\_arr and second\_arr, which are arrays of floats. Use one line of code to answer the following questions: How much larger (or smaller) is the sin of the first element in first\_arr than the cosine of the last item in second\_arr.

np.sin(first\_arr.item(0)) -

np.cos(second\_arr.item(len(second\_arr)-1))