

**Chapter 1 and 2 Quiz:**

* Write an expression that equals to 200. For example, 100 + 100. Trying using more than one operator and submit only one line of code.

m=[300-100],[600/3],[200\*1]

m

**Output: ([200], [200.0], [200])**

* My phone bill for the last three months has been $35, $40, and $54. What is the average monthly electricity bill over the three-month period? Write an expression to calculate the mean and use print () to view the result.

bill= [35,40,54]

bill

**Output: [35, 40, 54]**

sum(bill)

**Output: 129**

**avg = sum(bill)/len(bill)**

**print(avg)**

**Out:** 43.0

* Which of these lines of Python code are well formatted? How would you improve the readability of the codes that don't use good formatting? (Choose all that apply)

print(((3+ 32))+ -15//2)

print((17 - 6)%(5 + 2)) ...... well formatted

print ((1 + 2 + 4) / 13) ......

print(4/2 - 7\*7)

**Correct way:** print(((3+ 32)) + (-15//2))

**Correct way:** print((4/2) - (7\*7))

* Now it is your turn to work with variables. The comments in this quiz (the lines that begin with #) have instructions for creating and modifying variables. After each comment write a line of code that implements the instruction.

Note that this code uses scientific notations to define large numbers. 4.445e8 is equal to 4.445 \* 10 \*\* 8 which is equal to 444500000.0.

# The current volume of a water reservoir (in cubic metres)

* reservoir\_volume = 4.445e8

# The amount of rainfall from a storm (in cubic metres)

* rainfall = 5e6

# decrease the rainfall variable by 10% to account for runoff

rainfall =5000000.0

5000000.0 - (5000000 \* 10 / 100.0 )

**Output:** rainfall\_variable = 4500000.0

# add the rainfall variable to the reservoir\_volume variable

reservoir\_volume = 444500000.0

rainfall\_variable = 4500000.0

**Output:** Total\_variable = (4500000.0) + (444500000.0)

Total\_variable = 449000000.0

Total\_volume # increase reservoir\_volume by 5% to account for stormwater that flows into the reservoir in the days following the storm

**Output:** Total\_volume =  **(**444500000.0 \* 5) / 100.0 + 445000000.0 #(22225000.0)

Total\_volume = 467225000.0 (total increase in reservior volume)

# decrease reservoir\_volume by 5% to account for evaporation

**Output:** reservoir\_volume = 467225000.0

444500000.0 - (467225000.0 \* 5 / 100.0)

#(23361250.0)

reservoir\_volume = 422275000.0

# subtract 2.5e5 cubic metres from reservoir\_volume to account for water that's piped to arid regions.

**Output**: 2.5e5 = 250000.0

reservoir\_volume = 422275000.0- 250000.0

**reservoir\_volume = 422025000.0**

# print the new value of the reservoir\_volume variable

**Output**: **print(reservoir\_volume) = 422025000.0**

* How does changing the value of a variable affect another variable that was defined in terms of it? Let's look at an example.

We're intentionally not providing a place to execute the code here, because we want to help you practice the important skill of walking through lines of code by hand.

Each line of code executes in order, one at a time, with control going from one line to the next.

>>> carrots = 24

>>> rabbits = 8

>>> crs\_per\_rab = carrots/rabbits

**Output**: **3.0**

Now we add a new 4th line to this code, that assigns a new value to the rabbits variable:

>>> rabbits = 12

If we now add this new 5th line of code to the above, what will the output be?

>>> print(crs\_per\_rab)

* 0.5
* 2.o
* 3.0
* None of the above

**Output: cars\_per\_rab = 24/12**

**print (cars\_per\_rab) = 2.0**

* In Python 3 what is the output of ½?

**Output: 0.5**

* Guess the correct output of following code.

str1 = "PYnative"

print(str1[1:4], str1[:5], str1[4:], str1[0:-1], str1[:-1])

 PYn PYnat ive PYnativ vitanYP

 Yna PYnat tive PYnativ vitanYP

 Yna PYnat tive PYnativ PYnativ

**Output: Yna PYnat tive PYnativ PYnativ**

* Python does not support a character type; a single character is treated as strings of length one.

 False

 True

**Output: True**

* List all the python string methods with their definition. You can refer the internet for this.

**Output:**

**Captalize()** : Returns a copy of the string with its 1st character capitalized and the rest in lower case

**Casefold()** :Returns a casefold copy of the string.Casefolded strings may be used for caseless matching.

**lower() ,s.upper()** : Both returns the lowercase or uppercase version of the string

**isupper(),islower():**  Returns True if all characters in the string are upper case or lowercase

isalpha(), isnumeric(): Returns True if all characters in the string are in the alphabet or numeric

**swapcase():** Swaps cases, lower case becomes upper case and vice versa

**title():** Converts the first character of each word to upper case

**strp():** Returns a whitespace removed from the start and end

**count():** Returns the number of times a specified value occurs in a string

**index():** Searches the string for a specified value and returns the position of where it was found

**startswith(), s.endswith():** tests if the string starts or ends with the given string

**find() :** searches for the given string and returns the first index,basically finds the position(index)

**replace('oldvalue','newvalue'):** Returns a string where all occurrences of 'old' have been replaced by 'new'

**split():** Returns a list of substrings separated by the given delimiter.

**spitlines():** Splits the string at line breaks and returns a list

**translate():** Returns a translated string

**join() :** Joins the elements of an iterable to the end of the string

**lstrip()** : Returns a left trim version of the string

**rstrip()** : Returns a right trim version of the string

**format()** : Formats specified values in the string

**len():** returns length of the string

* Write one line of code giving an example of indexing. Foe example ‘Hello World’[0] returns ‘H’. Make any two examples of your own one with normal indexing and other one with reverse indexing.

**Output: "welcome to the jungle"[5] = m**

**"welcome to the jungle"[-5] = u**

* Reverse this ‘This is an easy quiz’ sentence without using string reverse method.

**Output: str = "This is an easy quiz"**

**print ('This is an easy quiz'[:: -1])**

* Use slicing techniques with providing 4 different examples each one using different slicing techniques learned in the video.

**Output:**

* Are strings immutable?

**Output: Yes, String is immutable**

* Change the name of the two strings from ‘Hero’ to ‘Zero’ using the slicing and string concatenation method. Use least lines of code as you can.

**Output: str = "Hero"**

**"Z" + str [1:] Out: 'Zero'**

* Write two expressions using format() method and f-string method.

**Output:**

**format() method**

**"{} {} is hard work".format ("Learning", "Python")**

**Out: 'Learning Python is hard work'**

**f-string method**

**name= "Manisha"**

**age = 16**

**flag = True**

**s = f' {name} is {age} and {flag}**

**print s**

**Out: Manisha is 16 and True**

* Write code to compare these densities. Is the population of San Francisco more dense than that of Rio de Janeiro? Print True if it is and False if not.

sf\_population, sf\_area = 864816, 231.89

rio\_population, rio\_area = 6453682, 486.5

san\_francisco\_pop\_density = sf\_population/sf\_area

rio\_de\_janeiro\_pop\_density = rio\_population/rio\_area

# Write code that prints True if San Francisco is denser than Rio, and False otherwise

->

**Output:**

**d** = {"sf\_population**" :** 864816, "sf\_area " : 231.89, "rio\_population" : 6453682, "rio\_area " : 486.5}

sf\_population = d [ "sf\_population" ]

rio\_population = d [ "rio\_population" ]

sf\_area = d [ "sf\_area " ]

rio\_area = d [ "rio\_area " ]

if (san\_francisco\_pop\_density > rio\_de\_janeiro\_pop\_density ):

print (True)

else:

print (False)

**Out:** san\_francisco\_pop\_density = 3729.423433524516 rio\_de\_janeiro\_pop\_density = 13265.533401849949

False

* Give examples of all the datatypes you learned.

**Output:**

**Text Type: str**

**e.g: x= "Hello World"**

**Numeric Types: Float,int**

**e.g: x= 5 Print(type(x))**

**Sequence Types: list,tuple,range**

**e.g: list = ["banana","shoes,"orange", 1,2,3]**

**tupple = ("apple", "banana", "cherry)**

**x = range(5)**

**Mapping Type: dictionary**

**e.g: x = {"name": "John", "age:" 36}**

**Set Types: Set**

**e.g: x = {"apple", "banana", "cherry"}**

**Boolean Type: bool**

**e.g: x = true**

* What function do you use to find length of a string?

**Output: x = len()**

* What type does this object have? "hippo" \*12

**Output: string**

* Create an example of list, with mixed data types. The answer should be just one line of code.

**Output:** my\_list = ['hippo', 2, 3, 2.50, 'true']

* If list1 = [1,2,3,4,5]. What is the output of list1.pop ()? And what would be the result of list1[1:]?

**Output:** list1.pop() = [1, 2, 3, 4]

list1[1:] = [2, 3, 4]

* Create a Dictionary where all the keys are strings and values are integers.

**Output:** dict **=** {'shoes':'35.00', 'hat': '20.00', 'chair': '52.00'}

dict

* Create a dictionary within a dictionary and write the code to find the values in the inside dictionary.

**Output:** dict = {'shoes':'35.00', 'hat': '20.00', 'chair': '52.00', 'dress' : {'insidekey red' : 25.00},}

dict

**Out:** {'shoes': '35.00', 'hat': '20.00', 'chair': '52.00', 'dress': {'insidekey dress': 25.0}}

**Output:** new\_dict1 = {'dress': {'insidekey dress' : 25.00}}

new\_dict1.values()

**Out:** dict\_values([{'insidekey dress': 25.0}])

* Do dictionaries retain order and are they a sequence?

**Output: No, Dictionaries do not retan order and are not in sequense as they are unordered, changeable and indexed**

* Given d = { ‘k1’:[1,2,3]}. What is the output of d[‘k1’][3]?

**Output**: **IndexError: list index out of range**

* Are dictionaries immutable?

**Output: No, In python all objects can be either mutable or immutable... Objects of built in types like [list,set,dictionaries] are mutable and [int,float,bool,str,tupple] are immutable**

* Do tuples have lots of methods associated with them?

**Output**: **No, Tuples have only 2 methods ...count() and index()**

* When is the best time to use tuples over lists?

**Output**: **Lists has a variable size and tuples have a fixed size.We use a list when we want to contain similar items, but we can use tuple only when we know what information goes into it**

* Are tuples immutable?

**Output**: **yes.**

* Which of the following is a tuple?
* [1,2,3]
* (1,2,[1,2,3])
* {1,2,4,)
* None of the above?

**Output: (1,2,[1,2,3])**

* Write an expression to turn the string ‘Mississippi’ into a set of unique

characters.

**Output:**

**str = "Mississippi"**

**my\_set = set(str)**

**print(my\_set )**

**Out: {'p', 'i', 's', 'M'}**

* {1,2,3,4} Is this a set?

**Output: No, It is dictionary**

* What method do you use to add an element to a string?

**Output: Concatination**

* What is the result of: set ([1,2,2,3])

**Output: {1,2,3}**

* An error
* [1,2,3]
* {1,2,2,3}
* {1,2,3}
* What is the output of following lines of code? What would the length be?

a = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]

b = set(a)

print(len(a) - len(b))

**Output**: **6**

* Consider:

a = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]

b = set(a)

b.add(5)

b.pop()

After executing this code, will the number 5 be a part of the set b?

**Output: Yes**

**a = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]**

**b = set(a)**

**b.add(5)**

**b.pop()**

**print (b)**

**Out: {2, 3, 4, 5}**

* Define a Dictionary, population,

# that provides information

# on the world's largest cities.

# The key is the name of a city

# (a string), and the associated

# value is its population in

# millions of people.

# Key | Value

# Shanghai | 17.8

# Istanbul | 13.3

# Karachi | 13.0

# Mumbai | 12.5

**Output**: population = {"Shanghai" : 17.8, "Istanbul" : 13.3, "Karachi" : 13.0, "Mumbai" : 12.5}

print (population)

**You can always find more questions online and try to attempt those too. I tried keeping it basic and less questions. But try finding more quizzes online and try to solve those!**

**---X---**