**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **21 may 2020** | **Name:** | **Veronica gudagur** |
| **Course:** | **python** | **USN:** | **4al16ec091** |
| **Topic:** | **Function** | **Semester & Section:** | **8-B** |
| **Github Repository:** | **Veronica-g** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  **C:\Users\Nelson\Desktop\01.PNG**  **C:\Users\Nelson\Desktop\z2.PNG** |
| **Function Basics & Advanced (Recursion, optional arg,**  **key word arg)**  Functions Basics and little Advanced:  Function takes input and spits off an output  Inputs are called argument, parameters, inputs, passing values/reference.  These functions don’t return anything.  After you write a simple code you convert it into functions.  def print two(arg1, arg2):   1. print(f"arg1: {arg1}, arg2: {arg2}") 3. # this just takes one argument 4. def print one(arg1): 5. print(f"arg1: {arg1}") 7. # this one takes no arguments 8. def print none(): 9. print ("I got nothing' to return.")  12. print two("S", "J") 13. print one("S!") 14. print none()   Below are the functions that would return the calculation.  Create errors and give only one or three arguments.   1. def add(a, b): 2. print(a,b) 3. return a + b 5. def subtract(a, b): 6. print(a,"minus",b) 7. return a - b 9. def multiply(a, b): 10. print("multiple",a,b) 11. return a \* b 13. def divide(a, b): 14. print("DIVIDING",a,b) 15. return a / b 17. print("Let's see some functions!") 18. age = add(30, 5) 19. height = subtract(78, 4) 20. weight = multiply(90, 2) 21. iq = divide(100, 2)   Try to see if you can find out the order of execution for the above...   1. def xxx(\*args): 2. for xy in args: 3. print(xy) 4. xxx('aaa','bbb','ccc','eee')   This taken in multiple arguments.    Search yourself key word arguments and how they use keys in the function input. These key word arguments are used a lot in Django!   1. x = "I like the way python works" 2. y = x.split() 4. def print\_first\_word(words): 5. """Prints the first word after popping it off.""" 6. word = words. Pop(0) 7. print(word) 9. print\_first\_word(y)   First recursive function:  What is recursion? There is theory that when we find out about this matrix we fund recursion in fabric of code for this universe.  Don’t worry if you don’t get recursion, it is rarely used in day to day programming unless you are too worried about speed and efficiency when writing big code for big data!   1. def factorial( n ): 2. if n <1: # base case 3. return 1 4. else: 5. returnNumber = n \* factorial( n - 1 ) # recursive call 6. print(str(n) + '! = ' + str(returnNumber)) 7. return returnNumber 8. def foo(\*args): 9. for a in args: 10. print (a) 11. foo(1,2,3,5,6)   Lambda function- create functions on the fly in python in minimal code. Only JS allows you to the same. And now scala.   1. def short function(x): 2. return x \* 2 4. f = lambda x: x \* 2 6. f(6) 8. short function(6)   Passing function as an  argument   1. def apply\_to\_list(some list, short function): 2. return [short function(x) for x in some list] 4. apply\_to\_list([1,2,3],short function )  7. ints = [4, 0, 1, 5, 6] 8. apply\_to\_list(ints, lambda x: x \* 2)   strings = ['foo', 'card', 'bar', 'aaaa', 'abab']  strings. Sort(key=lambda x: len(set(list(x))))   1. strings   Try to convert, if you cannot convert return back what you have.  def attempt\_float(x):   1. try: 2. return float(x) 3. except: 4. return x   Difference between printing and returning.  def joshif(x, y, z):   1. print("function ran - message from inside of function")  4. print ("end of the function") 5. return (x + y) \* z   Optional argument with if else:  def add\_and\_maybe\_multiply(a, b, c=None):  result = a + b    2. if c is not None: 3. result = result \* c 5. return result   We have looked at functions that are:  taking input vs not taking input  spitting output vs not spitting output  fixed number of arguments vs variable number of arguments  optional arguments And we see that most of times when we do something we convert it into functions so that we can use it later when needed.  **Another way to look at Functions - Input and Output**  One input and three output!  Capturing three output and unpacking the output.   1. def secret formula(started): 2. samosa = started \* 500 3. dosa = samosa / 1000 4. idli = dosa / 100 5. return samosa, dosa, idli 7. start point = 10000 8. s,d,i = secret formula(start point) 9. tupleoutput = secret formula(start\_point) 10. # remember that this is another way to format string 11. print(f"We'd have {s} , {d} , and {i} .")    print("We'd have {s} , {d} , and {i}".format(\*tupleoutput ))   If we don’t want to capture 3 outputs in separate variables   1. start\_point = 10000 2. print ("We'd have %d beans, %d jars, and %d crabapples." % secret formula(start\_point))   This was another function to understand function. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |