**Practice 02 - Control and Environment**

I**nstructions:** Solve the questions on the paper first. You can verify your solution using the Python shell.

### Part 1 - Control

**Practice 1:** What Would Python Display?

|  |  |
| --- | --- |
| >>> **def** **xk**(c, d):  ... **if** c == 4:  ... **return** 6  ... **elif** d >= 4:  ... **return** 6 + 7 + c  ... **else**:  ... **return** 25  >>> xk(10, 10)  23  >>> xk(10, 6)  23  >>> xk(4, 6)  6  >>> xk(0, 0)  25 | >>> **def** **how\_big**(x):  ... **if** x > 10:  ... print('huge')  ... **elif** x > 5:  ... **return** 'big'  ... **elif** x > 0:  ... print('small')  ... **else**:  ... print("nothin'")  >>> how\_big(7)  big  >>> how\_big(12)  huge  >>> how\_big(1)  small  >>> how\_big(-1)  nothin |
| >>> n = 3  >>> **while** n >= 0:  ... n -= 1  ... print(n)  2  1  0  -1 | >>> positive = 28  >>> **while** positive:  ... print("positive?")  ... positive -= 3  Positive?  Positive?  Positive?  Positive?  Positive?  Positive?  Positive?  Positive?  Positive?  Positive?  Positive?  (Infinite loop as it never reaches 0) |

*Hint*: Make sure your while loop conditions eventually evaluate to a false value, or they'll never stop! Typing Ctrl-C will stop infinite loops in the interpreter**Practice 2**: Boolean Expressions

|  |  |
| --- | --- |
| >>> **True** **and** 13  13  >>> **False** **or** 0  0  >>> **not** 10  False  >>> **not** **None**  True  >>> **not** 0  True  >>> (1 + 1) **and** 1  1  >>> 1/0 **or** **True**  Error (Divide by zero)  >>> (**True** **or** **False**) **and** **False**  False | >>> **True** **and** 1 / 0 **and** **False**  Error (Divide by zero)  >>> **True** **or** 1 / 0 **or** **False**  **True**  >>> **True** **and** 0  False  >>> **False** **or** 1  1  >>> 1 **and** 3 **and** 6 **and** 10 **and** 15  15  >>> 0 **or** **False** **or** 2 **or** 1 / 0  2 |

**Practice 3**: What is the result of evaluating the following codes?

|  |  |
| --- | --- |
| def square(x):  print("here!")  return x \* x  def so\_slow(num):  x = num  while x > 0:  x = x + 1  return x / 0  **square(so\_slow(5))** | >>> positive = -9  >>> negative = -12  >>> **while** negative:  ... **if** positive:  ... print(negative)  ... positive += 3  ... negative += 3  \_\_\_\_\_\_  \_\_\_\_\_\_  \_\_\_\_\_\_ |
| Answer:  (Blank)  Infinite Loop of x + 1 as x never reaches 0 and so doesn’t return | -12  -9  -6 |

### 

### Part 2 - Environment Diagram

**Practice 5**: def statements create function objects and bind them to a name. To diagram def

statements, record the function name and bind the function object to the name.

It's also important to write the parent frame of the function, which is where the

function is defined.

1. Draw the function object to the right-hand-side of the frames, denoting the intrinsic name of the function, its parameters, and the parent frame   
   (e.g. func square(x) [parent = Global].
2. Write the function name in the current frame and draw an arrow from the name to the function object.

Use these rules and the rules for assignment statements to draw a diagram for the code below.

|  |
| --- |
| def double(x):  return x \* 2  def triple(x):  return x \* 3  hat = double  double = triple |