**Programming Concepts Mon/Wed 4-6pm**

**Mid Term Exam Study Guide**

**All homework assignments**

**Flow Chart: why use & symbols we have learned**

**Chapter 2:**

**Graphically depicts the steps that take place in the program**

**Program Development Cycle Phases: Name them and what happens during each phase**

**Algorithm: understand how to write one and how to use one**

**Pseudocode: understand the handout you were given**

**“fake code” used to focus on programs design. Words, not mathematical operators**

**Ex: Input the hours worked**

**Input the hourly pay rate**

**Calculate the gross pay as hours worked multiplied by pay rate**

**Display the gross pay**

**Input Output Processing: understand the three-part process; be able to explain**

***Print function*- you should know everything about this!!**

**Strings, String literals -- know all about them!**

**Comments: why use them, what they are**

**Variables: What it is; creating with assignment statements, what is an assignment statement**

**Variable declaration, initialization, uninitialized variables**

***print function with 2 arguments*, a string literal and a variable. Creating a statement that includes the variable but prints the expression.**

**Room=503**

**Print(‘I am staying in room number’ , Room)**

***Variable reassignment*: Garbage collection: python automatically removes a variable from memory after it has been reassigned a new value.**

**Green = 10**

**Green= 20 python will no longer use 10 as the value for Green**

***Numeric literals* - a number that is written into a program’s code**

**When a numeric literal is written as a whole number with no decimal it is considered an int 7, 24, -9**

**When a numeric literal is written with a decimal point it is considered a float 1.5, 3.1465**

**\*\* you need to be mindful when writing programs**

**Using built in *type function***

**type(1)**

**< class ‘int’>**

**type(2.5)**

**< class ‘float’>**

***Real numbers*: numbers with a fractional part are stored different than integers, and similar operations are performed differently on them as well**

***Data types* are used to categorize values in memory: integer classified as in and real number (decimal) is a float**

**Python also has data type str, which is used for storing strings in memory last\_name = “Zinn”**

**A variable in python CAN CHANGE types! It can be reassigned**

**X = 99**

**X = “Take me to your leader”**

**Variable goes from type int to type string**

**Input from Keyboard: reads data from the keyboard, usually it stores that data as a variable so it can be used by the program**

***Input function* reads input from the keyboard and returns that piece of string data back to the program. Normally use input function in an assignment statement**

**Variable = input(prompt)**

**The strings purpose is to instruct the user to enter a value; variable (above) is simply the name of a variable that references the data that was entered on the keyboard**

**Name = input(‘What is your name? ‘)**

**The input function Does NOT automatically display a space after the prompt. If you want a space before the input from the user you must put a space in your code!**

**Input function always returns the user’s input as a string, even if the user enters numeric data!! This can be a problem if you want to use the value in a math operation. Math operations can only be performed on numeric values, not strings. Perform Data Conversion Functions to get the desired results:**

**Hours = int(input(‘How many hours did you work? ‘))**

***Nested Functions*: hours = int(input(‘How many hours did you work? ‘))**

**The value that is returned from the input function is then passed as an argument to the int() function –( inner parentheses and then to the outer int function)**

**\*\* Understand how to explain the above nested function:**

**-It calls the input function to get a value entered at the keyboard**

**-Value that is returned from the input function is a string which is passed as an argument to the (int) function**

**-Then int value that is returned from the int () function is assigned to the hours variable**

***Int() and float() functions* only work if the item being converted is a valid numeric value**

**An error called an exception will occur while the program is running if not**

**Hand Tracing- Handout!! What is it? Why use it? Be able to use it for the midterm.**

**Calculations: *math operators*, *operands*, *math expression*, variables in expressions**

**/ operator performs floating point division // performs integer division**

**5 / 2**

**2.5**

**5 // 2**

**2 (no remainder, whole numbers)**

**\*\* // operator**

**When the result is positive, it is truncated, fractional part thrown away**

**When the result is negative, it is rounded AWAY FROM ZERO to the nearest integer**

**-5 // 2**

**-3 actual answer is -2.5 but it is rounded away from 0. 0 -1 -1.5 -2 -2.5 -3**

**PEMDAS in Python**

1. **Exponentiation: \*\***
2. **Multiplication, division, remainder: \* / // %**
3. **Addition and Subtraction: + -**

**\*\* when they are all present and equal in precedence go left to right !**

**Example: outcome = 12.0 + 6.0 / 3.0**

**Division has higher precedence and would be done first**

**\*\* always an exception in Math: when two \*\* operators share an operand you go right to left**

**Example 2\*\*3\*\*4 is evaluated as 2\*\*(3\*\*4)**

**Use parentheses to force math to be performed as you want it. Result= (a + b) / 4**

***Exponent Operator* : purpose is to raise a number to a power**

**Area = length\*\*2**

***Remainder operator* (also *modulus operator*) % performs division but instead of returning the quotient it returns the remainder**

**Commonly used in calculations that convert time or distances, detect odd or even numbers**

**As well as other specialized operations**

**Converting algebraic expressions for python \*\* GO OVER THESE !!!**

**Example a + b would be x = (a + b) / c**

**x= c**

***Mixed-Type Expressions* and *Data type conversions***

**When performing math operation on two operands, data type of the result will depend on the data type of the operands**

**Two int values, int**

**Two float values, float**

**Int and float, int will be temporarily converted to float and result will be float**

**Example: my\_number = 5 \* 2.0**

**Value of 5 will be converted to 5.0 \* 2.0 and the answer will be 10.0**

**This example is how the conversion happens implicitly**

**To explicitly perform a conversion: use int() or float() functions**

**Fvalue= 2.6**

**Ivalue = int(fvalue)**

**\*\* Know how to explain the above in words!! *Call, pass, argument, function, converts*, *truncates* are all examples**

***Line continuation character \* When you need it and when you don’t**

***Special argument end=’ ‘* end = ‘ ‘ how use, why use**

***Newline character***

**Specifying an *item separator***

***Escape characters* - how many? What are they? What do they do?**

***String concatenation* \*\*\* when concatenating you should NOT have quotes print on the screen (unless told to have them)**

***Format function*: Formatting numbers: know why, and where they go ‘.2f**

***First argument, second argument, format specifier, Scientific notation, comma separators, field width***

****

***Floating point percentage*: why and how to use**

***Formatting integers* ‘,d’**

**You use d as the type designator**

**You CANNOT specify precision**

***Named constants*; value that cannot be changed during the programs EXECUTION!**

**Make programs more understandable**

**Widespread changes can be made easily**

**Help prevent typographical errors**

**Chapter 7**

***Sequences*: an object that holds multiple items of data, stored on after the other. You can perform operations on a sequence to examine and manipulate the items stored in it.**

***Lists:* list is an object that contains multiple data items. Lists are mutable, dynamic data structures. You can index and slice lists. Use brackets []**

***Elements*: the items in a list are called elements**

***Repetition Operator*: makes multiple copies of a list and joins them all together. List \* n**

**\*You cannot create arrays in Python, python uses lists.**

***Indexing*: a way to access individual elements in a list. Indexing starts at 0**

***Len function* len(list)**

***Concatenating* lists use the + character. Same as chapter 2**

***Slicing expression* selects a range of elements from a sequence; a slice is a span of items**

**General format list\_name[start : end] BRACKETS**

***in Operator*: allows you to search for an item in a list general format:  *item* in list**

**Python Built in functions to work with Lists: know what they are and which are methods, statements and functions**

* **append method**
* **index method**
* **insert method**
* **sort methond**
* **remove method**
* **reverse method**
* **del statement**
* **min and max functions**
* **pop method**

**Copying lists: you must copy the list’s elements**

**Processing lists: ways that a program can process the data held in a list**

**Returning a List from a Function: a function can return a reference to a list. Gives you the ability to write a function that creates a list and adds elements to it, then returns a reference to the list so other parts of the program can work with it.**

**Save contents of a list to a file; read data from a file into a list. One way is using *sort method***

***Two-Dimensional Lists*: a list that has other lists as its elements; aka *nested lists***

**Useful when working with multiple sets of data… pg 377**

***Tuple:* immutable sequence uses ()**

**Does not support: append, remove, insert, reverse, or sort.**

**Why use? Tuples process faster; good choice when processing big data that doesn’t change**

**Converting between Lists and Tuples: built in list () function; tuple() function**

**Packages:**

***matplotlib package*: library for creating two-dimensional charts and graphs. NOT part of the standard python library, needs installed with a *pip* command.**

**Contains a *module* named *pyplot***

**A *function* named *plot***

**Unless you create an alias that is shorter you will need to type matplotlib.pyplot to import the module**

**The books suggests an alias of plt page 383**

**Plotting a graph: horizontal X and vertical Y axis.**

***title function* : allows you to add a title to your graph**

**Default: X axis begins at the lowest x coordinate in your set of data points and ends at highest**

**Y axis configured same way**

**You can change the lower and upper by *calling* the *xlim* and *ylim* *functions (and in bar graph)***

**You can customize tick mark labels with the *xticks* and *yticks functions* *(and in bar graph)***

**You can display round dot marker using the *keyword argument marker=’o’* (page 391 more)**

**Use the *bar function* in the *module* to create a bar chart**

**Default width in bar graph is 0.8 along the X axis. Can change by passing a *3rd argument* to *bar function* (create a variable for the desired width, and pass it Ex: plt.bar(left\_edges, heights, bar\_width)**

**Bar function allows to change the bar colors using a *tuple of color codes* as a *key-word argument*:**

**Pie Chart circle represents the whole and the slices represent percentages of the whole**

***pie function* : you pass a list of values as an argument. Pie function will calculate the sum of the values in the list, then use that sum as the value of the whole. Each element in the list will become a slice in the pie chart.**

***labels parameter* can be used to display labels for the slices in the pie. The argument you pass into the parameter is a list containing the desired labels, as strings. Ex: keyword argument labels=slice\_labels**

***\*pie function* uses colors automatically in order( page398) ; can specify different set using a *tuple* as an *argument* to *pie function’s colors parameter*.**

**Page 673 has a list of predefined color names that will work also.**