**Python in Easy Steps – Book**

**Chapter 1: Getting Started**

* **Input() function** will prompt the user to input their own information specified and then save it to the input variable
* The initial input will be shown to the user and then the response filled in will change the variable assigned
  + EX
* user = input('I am Python. What is your name? :')
* print('Welcome',user)
  + first time printing ‘user’ it will return ‘I am Python. What is your name?
  + When the user enters their name, the ‘user’ variable changes to the name
  + OUTPUT
  + I am Python. What is your name? :Zac
  + >>> print(user)
  + Zac
* **Print() function** can take in multiple string variables and by default with a single space between values
  + However, Print can also take input parameters such as sep =, and end =
* lang = input('What is your favorite programming language? :')
* print(lang, 'is', 'fun', sep = '\*', end = '!\n')
  + OUTPUT
  + What is your favorite programming language? :Python
  + >>> print(lang, 'is', 'fun', sep = '\*', end = '!\n')
  + Python\*is\*fun!
  + Sep = \* means the inputs will be separated by a \* instead of the default space
  + End = !/n means end the line with a ! and move to a new line
* **Syntax Error** – occurs when the IDE encounters code that does not confirm with Python rules such as missing a quote around a string
* **Runtime Error** – occurs during execution such as an misspelling of a variable
* **Semantic Error** – when code syntactically works and runs but does not give intended result

**Chapter 2: Performing Operations**

* **‘\t’** means tab spacing
  + EX when a =8 and b=2
  + Print(‘Addition:\t’, a, ‘+’, b, ‘=’, a+b)
  + >Addition: 8+2=10
* a += b -> a = (a+b)
* a -= b -> a = (a-b)
* insert and other sign and same concept applies
* REMEMBER
  + = means assign
  + == means equality
  + != means inequality
* Logical operators
  + AND &
  + OR |
  + NOT ~
* **Most used casting**
  + Int(x)
  + Float(x)
  + Str(x)
  + Chr(x) – integer to character
  + Ord(x) – character to integer

**Chapter 3: Making Statements**

* **Lists** – a variable containing multiple values
  + Expressed in [] with , separating values
  + Fruit = [apples, banana, orange]
* Indexing lists always start with 0 as the first object
* Fruit[0] would return apples where [1] is the second object, banana
* **Popular list methods**
  + List.append(x) – adds another element to an existing list
  + List.extend(L) – adds items from one list to another
  + List.insert(I,x) – adds item x to position l
  + List.remove(x) – removes element
  + List.pop(i) – removes item at index position I and returns it
  + List.index(x) – returns index position of of item x
  + List.count(x) – returns number of times x appears in the list
  + List.sort() – sorts list
  + List.reverse() – reverses all items in list
* Can also delete using position I del(i) and can provide a range i2:i4 such as
  + Del basket[1:3]
* **Tuple** – a list that cannot have its elements or items manipulated like above
  + tuples are created using () instead of []
  + You can assign variables to items in a tuple
  + Colors = (‘red’, ‘blue’, ‘yellow’)
  + A,b,c = colors
* **Set –** similar to a tuple that cannot be manipulated AND can only have unique values
  + created using {}
* **Dictionary –** comes in the format of key:value built using {}
  + userSys = {‘name’:’Zac’, ‘sys’:’Win’}
  + Dictionaries can be merged using |
  + Dict3 = dict1 | dict2
* **Branching Ifs**
  + **Example**

If <test expression 1>:

<statement to execute when test expression 1 is True>

Elif <test expression 2>:

<statement to execute when test expression 2 is True>

Else: <statement to execute when both expressions are False>

* Remember – If, Elif, and Else must all be followed by a colon :
* Each statement underneath must be indented
* if num > 5:  
   print('Number exceeds 5\n')  
  elif num < 5:  
   print('Number is less than 5\n')  
  else:  
   print('Number is 5\n')

**Translated:**

If the number entered is greater than 5, print “Number exceeds 5”  
OR else if the number entered is less than 5, print “Number is less than 5”  
else if neither of the above, print “number is 5”   
(because if the number entered is neither great or less than 5, then it must be 5)

* **Match Case**
  + Value I specified after a match case key word and then compared to a specific pattern
  + Key here is to match a specific pattern specified

Match value:

Case pattern 1:

Statement to execute if value matches pattern 1

Case pattern 2:

Statement to execute if value matches pattern 2

#match case Sample  
cmd = input('Enter Stop or Go: ')  
match cmd:  
 case 'Go':  
 print('Started...')  
 case 'Stop':  
 print('Halted...')

**Translated:**

Cmd variables asks the user to enter either Stop or Go  
if the user matches to the input Go then print Started…  
if the user matches to the input Stop then print Halted…  
inputs must match exactly to return the specified response

**While Loops: executed a block of statements repeatedly until a given condition is satisfied**

* ‘i’ is the usual counter variable standing for iteration
* Key word ‘while’ begins the while loop function to iterate over a specified block of code until a certain criteria is met
* i = 1  
  while i < 6:  
   print(i)  
   i += 1

**Translation:**

Begin the iteration count at 1  
While i is less than 6  
print the value of i  
each time it loops through, increase the value of i by 1  
stop the loop once i reaches 6

**For Loop: allows us to run a series of instructions once mainly for elements in a list, tuple or set**

* The For keyword loops over all items in any list specified to the IN keyword

For <each item> in <list name>:

<statements to execute in each iteration>

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
 print(x)

Translated:

The list called ‘fruits’ contains apples, banana, and cherry  
For x (variable representing each item) in the list ‘fruits’  
print each value

**Another way to think of it:**

A white text with black text

Description automatically generated

**Breaking out of loops**

* Provides a specific condition to break out of the loop
* #Breaking Loops  
  i = 1  
  while i < 6:  
   print(i)  
   if i == 3:  
   break  
   i += 1  
    
  fruits = ["apple", "banana", "cherry"]  
  for x in fruits:  
   print(x)  
   if x == "banana":  
   break
* While counting 1 – 6, if i equals 3 – break out and stop the loop
* For the fruits list, when cycling through and you reach banana, break out and stop the loop

**Chapter 4: Defining Functions**

* To define your own function, begin in with keyword ‘def’ followed by the function name, () and a :
* Following the name, the actual code of the function must be indented
* EX

Def <function name> () :

Statement-to-be-executed  
 statement-to-be-executed

* Function variables scope
  + **Global** – variables created outside of function that can be referenced inside the function if specified to be a ‘global’ variable
  + **Local** – variable created inside the function that cannot be referenced outside of the function once created
* An argument or parameter can also be passed into a function if specified – instead of a blank parenthesis, put the name of the input
  + EX
  + A screen shot of a computer

    Description automatically generated
  + 
  + You can also place a default value that will appear if no alternative value is provided
  + A screen shot of a computer

    Description automatically generated
  + 
* **Return** keyword is used when we want to see the result of an equation or specified code
  + **Print** is used more when we want to print a specific output rather than the answer
  + EX
  + A screen shot of a computer

    Description automatically generated
  + 
* **Lambda Function**
  + Anonymous unnamed function that can only contain a single expression and must return a value
  + Lambdas are often used to embed a function within code
  + EX
  + A black background with white text

    Description automatically generated
* **Placeholder** 
  + Pass keyword can be used to insert into an incomplete function or code block to pass over for the time being until it is complete and it won’t interfere with testing the larger code block
  + EX
  + A black screen with white text

    Description automatically generated
  + Allows user to test the if portion without coming up with an else part for the time being
* **Exception Handling**
  + **Try –** lets you test a block of code for errors
  + **Except –** lets you handle the error
  + **Else –** lets you execute code when there is not error
  + **Finally –** lets you execute code regardless of the result of the try and except blocks

A computer screen shot of a program

Description automatically generated



**Chapter 7: Object Oriented Programmer:**

* OOP allows programmers to create their own objects with methods and attributes
  + Example of a method is using .append when you created a list
* Methods act as functions specifically for an object and use info about the object and the object itself to return results
* Purpose allows programmers to create code that is repeatable and organized where creating a function is no longer at scale for the job
  + Key here is OOP is repeatable, flexible, and organized
* **Class** is the keyword to begin constructing an object
  + Begins blueprint that defines the nature of a future object
  + Instance is a specific object created from a particular class
* **Method** has similar functionality and looks like a function, but is considered a method when it is created WITHIN an object
* **Object > Class > Instance > Attribute**
* A screenshot of a computer code

  Description automatically generated
* Class = dog, method in this case is \_\_init\_\_, self is a reference to the instance of the class, breed name and spots are the attributes
* Parameters are usually the same as the attributes
* A screenshot of a computer

  Description automatically generated
* A screenshot of a phone

  Description automatically generated