Python Basics Cheat sheet

Basics and Python Assignment 1:

Python Version

$ python –version

Commenting : Use the “#” to start a commented line, it will not be executed by the python interpreter

```
# This is a comment will not get executed in python
```

Variables: Are place holders for various types of data such as the ones given below

Numbers:

```
# Assign integer to a variable
x = 3

# print the variable
print(x)       # Prints "3"

# print the type of variable
print(type(x)) # Prints "<class 'int'>"


# Basic operations on Numbers

print(x + 1)   # Addition; prints "4" (3 plus 1)
print(x - 1)   # Subtraction; prints "2" (3 minus 1)
print(x * 2)   # Multiplication; prints "6" ( 3 times 2)
print(x ** 2)  # Exponentiation; prints "9" (3 raised to power 2)

# Operations to update a varible
# To increment x by 1 and assign the value back
x += 1 # (this can also be done by x = x + 1)
print(x)  # Prints "4"

# To assign a multiple of x back to variable x
x *= 2 # (this can also be done by x = x * 2)
print(x)  # Prints "8"


# Assign a decimal to a variable (the data type is called float)
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # Prints "2.5 3.5 5.0 6.25"
```

Python Basics Cheat sheet

Strings :

```
# Assign string to a variable
hello = 'hello'     # String literals can use single quotes
world = "world"     # or double quotes; it does not matter.

# print the variable
print(hello)        # Prints "hello"

# print length of the variable
print(len(hello))   # String length; prints "5"

# Concatenation operation on string
hw = hello + ' ' + world  # String concatenation
print(hw)  # prints "hello world"
```

String formatting methods :

```
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints
"HELLO"
print(s.lower())       # Convert a string to lowercase; prints
"hello"
print(s.rjust(7))      # Right-justify a string, padding with
spaces; prints "  hello"
print(s.center(7))     # Center a string, padding with spaces;
prints " hello "
print(s.replace('l', '(ell)'))  # Replace all instances of one
substring with another;
                                # prints "he(ell)(ell)o"
print('  world '.strip())  # Strip leading and trailing whitespace;
prints "world"
```

Reading user inputs to store into variables:

```
# Read input from console
x = input()    # reads the input as a string
example input: hello
print(x)    # prints 'hello'

# Read an integer
y = int(input())    # reads the input as a string and converts it to
integer
```

```
example input: 5
print(y)    # prints 5

# Read a floating (decimal) number
z = float(input())    # reads the input as a string and converts it
to floating point number (decimal)
example input: 1.7
print(z)    # prints 1.70

# Print instructions before seeking input from user
Print("Please enter an integer between 1 and 100")
inp = int(input())    # user now knows the input to provide based on
instruction
print(inp)    # prints the number provided
```

Changing type of data:

```
# change between string, float and integer
x = input()    # reads the input as a string
example input: 8
print(x)    # prints '8'

y = int(x)
print(y)    # prints 8

z = float(y)
print(z)    # prints 8.0

x2 = str(z)
print(x2)    # prints '8.0' which is a string with three characters
taken from the variable z
```

Comma separated printing across data types:

```
# print multiple data types in a single print statement
sent1 = "I bought"   # reads the input as a string
count = 5   # reads the input as a string
sent2 = "chocolates at a price of"
price = 1.2   # reads the input as a string
sent3 = "dollars per piece"   # reads the input as a string

print(sent1, count, sent2, price, sent3)   # prints "I bought 5
chocolates at a price of 1.2 dollars per piece"
```

Python Basics Cheat sheet

Conditional statements:

```
# if, else and elif (else if) statements to check conditions
x = int(input())    # reads an integer input
if x > 0:
    print(‘positive’) # prints ‘positive’ if x is greater than 0
elif x < 0:
    print(‘negative’) # prints ‘negative’ if x is lesser than 0
else:
    print(‘zero’)    # prints ‘zero’ if x is exactly 0
```

Functions: The same logic above can be implemented in a reproducible way. Functions are used in order to do this as shown below.

```
# if, else and elif (else if) statements to check conditions
def sign(x):
    if x > 0:
        return ‘positive’ # returns ‘positive’ if x is more than 0
    elif x < 0:
        return ‘negative’ # returns ‘negative’ if x is lesser than 0
    else:
        return ‘zero’    # returns ‘zero’ if x is exactly 0

input_num = int(input())    # reads an integer input
Example input: -6
sign_of_num = sign(input_num)    # calls the function, which can be
done for different set of numbers again and again

print(sign_of_num) # prints ‘negative’
```

While loops: repeatedly does operations as long as the condition meets

```
# consider a simple print statement
i = 1
while i<6:      # repeatedly goes back to the print and increment
statements until the condition breaks (that is when i becomes 6)
    print(i)    # prints the value of i
    i+=1       # increments the value of i to next number
```

Python Assignment 2:

Accessing parts of strings

```
# use the index to access part of string
var_str = “hello world”
print(var_str[0])    # prints “h”
print(var_str[4])    # prints “o”
print(var_str[0:5])    # prints “hello” (selects 0 to 4)

# find the index (first occurrence) of any letter
var_str = “hello world”
ind_str = var_str.index(“e”)
print(ind_str)       # prints “1” (“h” has index 0 and “e” has 1)
```

Lists: Creation, Access, Appending, Inserting and Deleting

```
# Creation
xs = [3, 1, 2]     # Create a list

# Access
print(xs, xs[2])   # Prints "[3, 1, 2] 2"
print(xs[-1])      # Negative indices count from the end of the list;
prints "2"

# Insert
xs[2] = 'foo'      # Lists can contain elements of different types
print(xs)          # Prints "[3, 1, 'foo']"

# Append
xs.append('bar')   # Add a new element to the end of the list
print(xs)          # Prints "[3, 1, 'foo', 'bar']"

# Delete last element
x = xs.pop()       # Remove and return the last element of the list
print(x, xs)       # Prints "bar [3, 1, 'foo']"

## Extend list of languages
languages = ['French', 'English']

languages1 = ['Spanish', 'Portuguese']

# appending language1 elements to language
languages.extend(languages1)
```

```
print('Languages List:', languages) # Prints ['French', 'English',
'Spanish', 'Portugese']

# appending language1 elements to language (option 2)
Languages2 = languages + languages1
print('Languages List:', languages2) # Prints ['French', 'English',
'Spanish', 'Portugese']


## Reverse the list
# Operating System List
systems = ['Windows', 'macOS', 'Linux']
print('Original List:', systems)  # Prints ['Windows', 'macOS',
'Linux']


# List Reverse
systems.reverse()

# updated list
print('Updated List:', systems) # Prints ['Linux', 'macOS',
'Windows']

## sorting a list
cars = ['Ford', 'BMW', 'Volvo']

cars.sort()

print(cars)  # Prints ['BMW', 'Ford', 'Volvo']
```

Loops:

```
# Looping through existing list
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
# Prints "cat", "dog", "monkey", each on its own line.


# Create a numerical list and loop through it using functions
for i in range(1, 10, 2): # range function is used to create a list
which starts from 1, increases in steps of 2 and maximum number is
lower than 10
    print(i)
# Prints 1 3 5 7 9
```

Python Assignment 3:

Reading text files

```
# [ ] Run to download file to notebook
!curl
https://raw.githubusercontent.com/MicrosoftLearning/intropython/mast
er/poem1.txt -o poem1.txt

# [ ] review and run example
# open address to file
poem1 = open('poem1.txt', 'r')

# read only the first line
poem_line1 = poem1.readline()

# readlines and print as a list
poem_lines = poem1.readlines()
poem_lines

# [ ] review and run example
for line in poem_lines:
    print(line)

# [ ] review and run example: Close poem1
poem1.close()

# To get rid of the new line character '\n'
poem_lines2 = []
For line in poem_lines:
    poem_lines2.append(line.rstrip('\n'))

# to split the data into a list having delimiters
poem1 = open('poem1.txt', 'r')
poem_line1 = poem1.readline()
poem_line1_list = peom1.split(',')

# write a list to file
f = open("myfile.txt", "w") # "w" indicates the file is in the write
mode
L = ["This is Singapore \n","This is Paris \n","This is London \n"]

# to write just a single string
f.write("Hello \n")
# to write a list of lines
f.writelines(L)
f.close()
```

Python Basics Cheat sheet

```
# List of modes for handling files in python
# “ r “, for reading.
# “ w “, for writing.
# “ a “, for appending.
# “ r+ “, for both reading and writing
# “ a+ ”, for reading, writing and appending
```