Python Cheat Sheet

Simple Operators	+, -, *, /, **
Basic Data Types	int, float, string, bool
Declaring Variables	pi = 3.14159
Simple Output	print()
Simple Input	raw_input()
Get Data Type	type()
Type Casting Variables	str(), int(), float()
Generate a List of Numbers	range(a, n-1)
For Loop	for <variable> in <list>:
While Loop	while <condition>:
Appending to a List	<list name>.append(<data>)
Random Number Generator	import random random.randint(a, n-1)

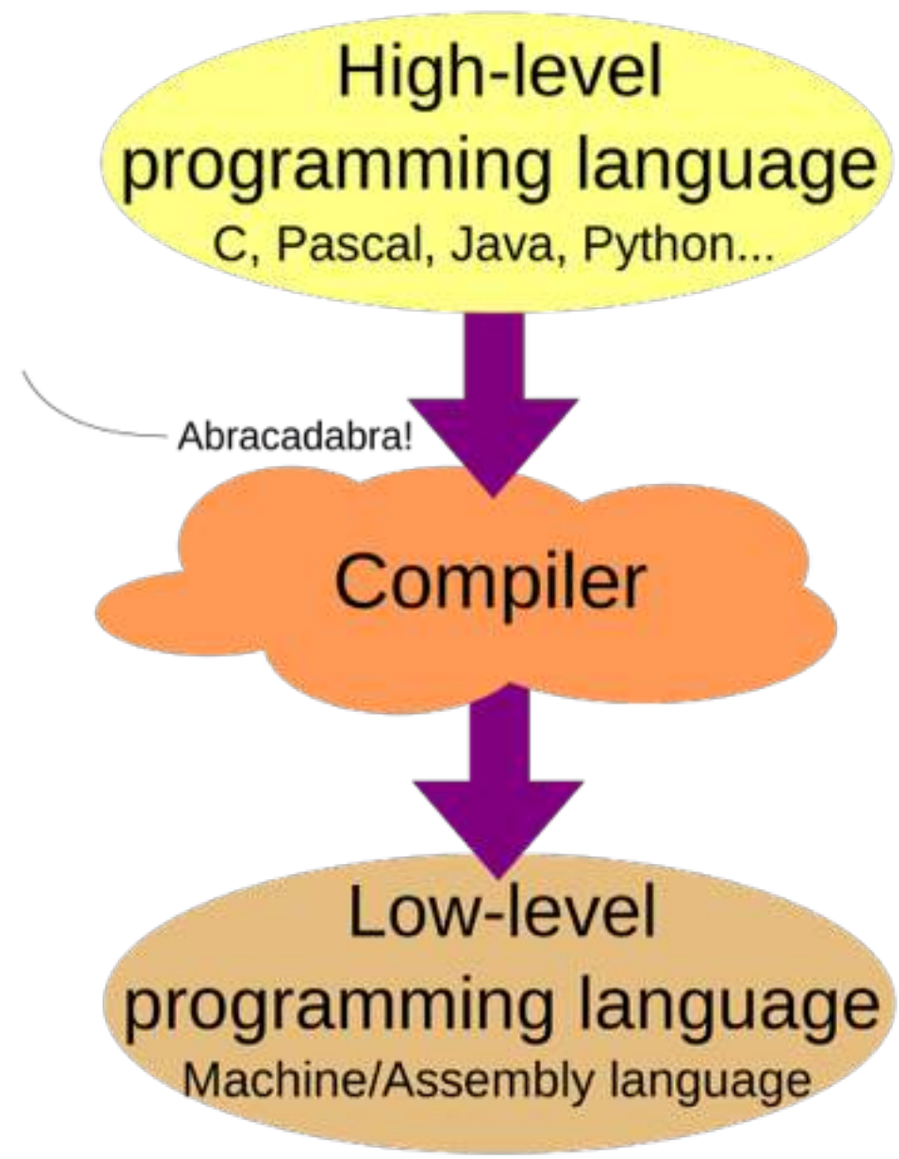
UNIX Cheat Sheet

Change Directory	cd
Move Up a Directory Level	cd ..
Home Directory	~/
List Directory Contents	ls
Move a File or Directory	mv
Copy a File or Directory	cp
Delete a File	rm
Display Help File	man
Retrieve Previous Command	up/down arrows
Autocomplete	<tab key>
Kill a running process	<cntrl>C
Open a secure shell	ssh <account>@<IP>

Launch ROS	roscore
See all ROS topics	rostopic list
See the telemetry from a ROS topic	rostopic echo <topic>
Launch a file	roslaunch <package> <file>

Types of Languages

- **Machine** Language
  - Binary, Hexadecimal
- **Assembly** Language
  - X86
- **High-Level** Language
  - 3rd Gen: BASIC, C++, Java
  - 4th Gen: Python



Using Python

- Python is an **interpreted** language
- Runs in an **I**ntegrated **D**evelopment **E**nvironment
- Python 2.x and 3.x
    - **F1/10 uses Python 2.6/2.7**

Launching a Python Environment

- Use **repl.it**
  - simple online compiler, IDE, interpreter, and REPL
    REPL - Read–Eval–Print Loop
- Launching **iPython**
  - Open a shell in *Terminal* or *Terminator*
  - Type “iPython” to launch the Python environment

Basic Math

+ − * / **

```
In [1]: 1+1
Out[1]: 2

In [2]: 4*3
Out[2]: 12

In [3]: 4/2+6
Out[3]: 8
```

Try finding 2x4+6/3

Is order of operations preserved?

What is the output of 5/2?

What's happening there?

Basic Variables

int, float, string

```
In [6]: x = 11

In [7]: y = 12.0

In [8]: x+y
Out[8]: 23.0

In [9]: myStr = "Hello World!"
```

Are 12 and 12.0 different? How?

What is the output of 5.0/2?

What happens when you sum two strings?

When you multiply a string by a number?

Basic I/O

print(), raw_input()

```
In [15]: print("Hello World!")
Hello World!

In [16]: x = 11

In [17]: print(x)
11

In [18]: name = raw_input("Please enter your name: ")

Please enter your name: Marvin

In [19]: print("Hello " + name + "!")
Hello Marvin!
```

Other important commands

type(), str(), int(), float()

```
In [21]: myString = "Hello"

In [22]: myInt = 12

In [23]: myFloat = 6.0

In [24]: type(myString)
Out[24]: str

In [25]: type(myInt)
Out[25]: int

In [26]: type(myFloat)
Out[26]: float
```

Use raw_input to collect two numbers, *a* and *b*

What happens when you add *a* and *b*?

When you multiply?

What is the type of data collected using raw_input?

Program 1: Circle

- **Write a program to calculate the diameter, circumference, and** area of a circle. Ask the user to enter a value for the radius.
- Echo out (print) the value of the radius as well as the results of your calculations
- Useful Formulas (watch out for type (float/int) errors)
  - d = 2 * r
  - c = d * 3.14
  - a = 3.14 * r^2

Creating a program in Terminal

- Open any plain text editor (i.e. gedit or vim)
- Write your code in a text editor and save with the extension ".py"
- In Terminal, navigate to the file's directory
- Make the file executable with: `chmod +x <script_name>.py`
- Run the file with: `python <script_name>.py`
- Refer to the UNIX cheat sheet at the beginning.

Program 2: Sphere

- Write a program to calculate the diameter, circumference, surface area, and volume of a sphere. Ask the user to enter a value for the radius.
- Write the program in a VM and run in a Terminal shell
- Echo out the value of the radius as well as the results of your calculations
- Useful Formulas (watch out for type errors)
  - d = 2 * r
  - c = d * 3.14
  - sa = 4 * 3.14 * r^2
  - V = 4/3 * 3.14 * r^3

Lists

[w, x, y, z]

```
In [1]: ["Hello", "World!"]
Out[1]: ['Hello', 'World!']

In [2]: x = 2

In [3]: y = 3

In [4]: z = 4

In [7]: myList = [x, y, z]

In [8]: myList
Out[8]: [2, 3, 4]
```

Lists are a complex data type denoted by square brackets [ ]

Lists are lists of some simple data type

You can have lists of variables or raw data or both

Try making a list of floats.
Can you make a list including multiple data types?

Lists

[w, x, y, z]

```
In [1]: ["Hello", "World!"]
Out[1]: ['Hello', 'World!']

In [2]: x = 2

In [3]: y = 3

In [4]: z = 4

In [7]: myList = [x, y, z]

In [8]: myList
Out[8]: [2, 3, 4]
```

Items in a list can be accessed by calling the list name followed by the **index**

Indexes begin at 0 and run through length-1

You can also assign specific indexes particular values by calling the index:

myList[2] = 5

Range

range(a, n-1)

```
In [1]: range(0, 3)
Out[1]: [0, 1, 2]
```

The range command takes two parameters, a lower limit and an upper limit

It creates a list starting at a and ending at n-1

For Loops

for *<item>* in *<list>*:

```
for count in range(0, 10):
    <code here>
```

Keyword “for”

Counting variable

Keyword “in”

Range command
(a –> b-1)

Colon

For Loops

for count in range (*a, n-1*):

```
In [2]: for count in range(0, 5):
   ...:     print("Hello World!")
   ...:
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Loops run in the range (a –> b-1),
In this case, (0, 5-1) = (0, 4)

Count is a changing variable

Try printing count inside the loop

Try finding the sum of 1-10
1-100?
10-100?

For Loops

for item in myInterval:

```
In [7]: myInterval = range(0, 11)

In [8]: for item in myInterval:
   ...:     print(item)
0
1
2
3
4
5
6
7
8
9
10
```

You can parse through the items in a list, regardless of the data type

Program 3: Sum of an Interval

- Write a program to find the sum of numbers on an interval entered by a user. The program should ask the user to enter an a and b value and then find the sum of of all numbers a–b ***inclusive***.

- Sample input/output should similar to this:
  - [in] Enter a lower limit: 5
  - [in] Enter an upper limit: 10
  - [out] The sum is 45

- Write the program in iPython or repl.it

If-Else Statements

if <condition>: … else:

Condition

Keyword “if”

Code to run

```
if x < 10:
    <code here>

else:
    <code here>
```

Keyword “else” (if not fulfilled)

Code to run

If Statements

if <condition>: … else:

```
In [1]: x = 10

In [2]: if (x < 12) and (x > 8):
   ...:     print("x between 8 and 12")
   ...:
x between 8 and 12

In [3]: x = 10

In [4]: if (x % 2) == 0:
   ...:     print("x is even")
   ...:
x is even
```

Mathematical comparisons can be used as follows:

>, <, >=, <=, ==

You can make “compound” if statements using the keywords *and, or*

You can put expressions into the condition field as well. Try determining even vs. odd numbers

Booleans

True, False

```
In [1]: sentinel = True

In [2]: if sentinel:
   ...:     print("Hello World!")
   ...:
Hello World
```

Booleans are variables holding a simple true/false

When using a boolean variable in an if statement, you don't need to make an explicit comparison, since the output of a comparison is a boolean

While Loops

while *<condition>:*

Control variable

Condition

Keyword “while”

```
x = 0

while x < 10:
    <code here>
    x += 1
```

CONTROL UPDATE

While Loops

while *<condition>:*

```
sentinel = 0

while sentinel:
    <code here>
    if <condition>:
        sentinel = False
```

Condition (boolean)

Stopping state

Program 4: Checking a List

- Write a program to parse through a list of numbers and check if the items in the list are in order. The program should output a “yes, list is in order” or a “no, list is not in order”
- The program should account for repeated numbers; the list should be still be considered in order if repeats of numbers are adjacent to each other.
- Write the program in iPython or repl.it

Random Number Generator

import random, random.randint(a, n-1)

Keyword “import”

Package to import

```
import random

random.randint(0, 10)
```

Package name

Package function

Limits

Making a random list

myList.append(<data>)

```
In [1]: myList = []

In [2]: for count in range(0, 10):
   ...:     myList.append(random.randint(0, 10))
   ...:

In [3]: myList
Out [3]:  [4, 6, 5, 5, 0, 7, 4, 3, 7, 7]
```

Create an empty list using empty square brackets []

Use a counting loop for how many numbers you want

Use the command
<listname>.append(<random call>)

Program 5: Linear Search

- Write a program that searches for a user-entered item in a list

- The program should generate a list of random numbers, ask the user for a target value, search for the target value in that list, then print the list and whether or not the item was found.

- The program should return a “yes” or “no,” as well as the index of the number if found

  Sample output:
  ‘[0, 3, 6, 9, 3, 7, 4, 9, 6]’
  “Yes, item found at index 4”

- Write the program in iPython or repl.it

Program 6: Bubble Sort

The Bubble Sort compares adjacent elements in a list, and “swaps” them if they are not in order.

Each pair of adjacent elements is compared and swapped until the largest element “bubbles” to the bottom.

Repeat this process (stopping one farther from the end each time, if you want to be efficient) until the list is in order

[1, 5, 4, 3]

[**1, 5**, 4, 3]
[1, **5, 4**, 3]
[1, 4, 5, 3]
[1, 4, **5, 3**]
[1, 4, 3, 5]

[**1, 4**, 3, 5]
[1, **4, 3**, 5]
[1, 3, 4, 5]

[1, 3, 4, 5]

Program 6: Bubble Sort

- Write a program that sorts a list of numbers
- The program should generate a list of random numbers, print the original (unsorted) list, execute a bubble sort, and print the (sorted) result.
- The program should return a “yes” or “no,” as well as the index of the number if found

  Sample output:
  ‘[0, 3, 6, 9, 3, 7, 4, 9, 6]’
  “Yes, item found at index 4”

- Write the program in iPython or repl.it

What do I submit?

Submit your code for Problems 4 (5 points), 5 (10 points), and 6 (10 points).

Submit one single .zip file with the name <your computing ID>.zip

This compressed file, should only include three files named:

problem4.py, problem5.py, problem6.py

Double check your code to ensure it runs and does not throw any errors.

Comment your code for full credit consideration.