Python Commands
Without Commentary
(for use as reference on exams and quizzes)

Hello, World!

```python
"""
A classic hello-world program.

:author: T.Sergeant
"""

print("Hello, World!") # we greet the world with optimism!
```

Expected output (on screen):

```
Hello, World!
```

Variables and Simple Types

```python
y = 0.1
z = 7.5 + y
name = "Fred" # quotes indicate a string
a = int(z)    # treat z as an int
a = a + 1
is_fun = True
```

y	0.1
z	7.6
a	8

name	"Fred"
is_fun	True

Displaying Values

```python
a = 23
name = "Mary"
print("Hello")
print("Name is: ", end='')
print(name)
print("Answer is: " + str(a))
```

Output of statements assuming variables have values from previous examples:

```
Hello
Name is: Mary
Answer is: 23
```

Getting Input

```python
word = input("Enter word: ")
age = input("Enter age: ")
next = int(age) + 1
num = float(input("Enter a number: "))
num = num + 2
```

```
Enter word: fun
Enter age: 10
Enter a number: 7.5
```

word	fun
age	10
next	11
num	9.5

Integer Arithmetic and Random Numbers

```python
import random
random.seed()  # do this once per program

a = 7
b = -1
c = a + b * 3 # 4
d = b - a // 3 # -3
e = b - a / 3 # -3.33333
f = a % 3     # 1
g = random.randint(1,100) # random val in range 1 to 100
```

a	7
b	-1
c	4
d	-3
e	-3.333333
f	1
g	57

Floating Point Arithmetic

```python
import math
a = 4.0
b = -2.2
c = a + b * 3           # -2.6
d = math.sqrt(a) + b # -0.2
```

a	4.0
b	-2.2
c	-2.6
d	-0.2

If Statements

```python
if a < 0:
   print("a is negative")
   a = 0
else:
   print("a is non-negative")

if b == 1:
    print("hi")
elif b == 2:
   print("there")
elif b == 3:
   print("how")
else:
   print("are")
```

Boolean Expressions

```python
is_cool = False
a = 7
b = -1
c = True
d = a < b
e = a < b or a > 0
f = a <= b and a != 0
g = not is_cool and a == 7
h = 0 <= b <= 100
```

a	7
b	-1
c	True
d	False
e	True
f	False
g	True
h	False

Strings

```
one = "Fun"
two = 'ny'
num = len(one)
three = one + two
print(three[1])
print(three[1:4])
if one < two:
   print("Fun is less than ny")
print(three.upper())
print(three.lower())
four = "   " + three + " guy"
print(four.strip())  # remove whitespace from beg and end
```

num	3
one	”Fun”
two	”ny”
three	”Funny”

Expected output:

```
u
unn
Fun is less than ny
FUNNY
funny
Funny guy
```

While Loops

```
a = 1
while a <= 8:
    print(a)
    a += 2   # shorthand for a = a + 2
```

Expected output:

```
1
3
5
7
```

Lists

```
a = ["a", 12, "fun", "hi", True, 17.3]
print(len(a))
print(a[3])
#a.sort()
print(a)

b = []           # b is an empty list
b = b + ["fun"]  # stick two lists together
b.append("neat") # add to end of list
b.insert(0, "two") # add to front of list
print(b)

c = a[1:4]
print(c)

val = a.pop()   # remove and return last element
a.remove("fun") # remove the value "fun" from list
del a[0]        # remove value a position 0
print(a)
print(val)

num = 12
if num in a:
   print("Found it!")
else:
   print("Not found.")
```

Expected output:

```
6
hi
['a', 12, 'fun', 'hi', True, 17.3]
['two', 'fun', 'neat']
[12, 'fun', 'hi']
[12, 'hi', True]
17.3
Found it!
```

For Loops

```python
a = ["hi", 12, "fun"]

for val in a:
    print(val)

for i,val in enumerate(a):
    print(str(i)+" "+str(val))

for num in range(4):
    print(num)
```

Expected output:

```
hi
12
fun
0 hi
1 12
2 fun
0
1
2
3
```

Functions

```python
def show_sum(a, b):
    c = a + b
    print("Sum is: "+str(c))

def calc_sum(a, b):
    c = a + b
    return c

# To call these functions ...
x = 7.5
show_sum(x, 1.0)
show_sum(x * 2.0, x - 2.0)       # positional arguments
show_sum(b = x - 2.0, a = x * 2.0) # keyword arguments
x = calc_sum(1.0, 2.2)
print(x)
```

Expected output:

```
Sum is: 8.5
Sum is: 20.5
Sum is: 20.5
3.2
```

DocString / reST Notation

```python
def calc_sum(a, b):
    """
    Calculate and return the sum of two numbers.

    :param a: first number to be used in sum
    :param a: second number to be used in sum
    :return: sum of a and b
    """
    c = a + b
    return c
```

Formatted Output

```python
x = 7.5
str = "Fred"
a = 2
print("%s is %d" % (str,a))
print("A%5d %4.2fB" % (a,x))
print("A%-5d %4.2fB" % (a,x))
```

Expected output:

```
Fred is 2
A    2 7.50B
A2     7.50B
```

Read Data File

```python
with open("fun.txt") as first:
    for line in first:
        line = line.rstrip() # remove \n from end of line
        print(line)

with open("fun.txt") as again:
   list_of_lines = again.read().rstrip().split("\n")
print(list_of_lines)

third = open("fun.txt")
for line in third:
   print(line.rstrip())
third.close()
```

Expected output of first and third examples is contents of the file `fun.txt` displayed to the screen. The second example will display each line of the file as values in a Python list.

Write to Data File

```python
with open("cool.txt", "w") as my_file:
   my_file.write("Hello\n")
   my_file.write("Goodbye\n")

with open("cool.txt", "a") as second:
   for i in range(3):
      second.write(str(i)+"\n")
```

Expected output (in file "cool.txt"):

```
Hello
Goodbye
0
1
2
```

Timing Code

```python
import time
start = time.time()
# put code to be timed here
stop = time.time()
time_in_seconds = stop - start
print(time_in_seconds)
```

Expected output: the time elapsed in seconds

Dictionaries

```python
letter_counts = { "a": 67, "b": 12, "c": 16 }
print(letter_counts.keys())
print(letter_counts.values())
print(letter_counts["c"])
letter_counts["d"] = 9
del letter_counts["a"]
val = letter_counts.pop("b")
print(val)
if "b" in letter_counts:
    print("b is there")
else:
    print("b is gone")
for key, value in letter_counts.items():
    print(key+" has value "+str(value))
```

Expected output:

```
['a', 'b', 'c']
[67, 12, 16]
16
12
b is gone
c has value 16
d has value 9
```

Nested Structures

```python
nested = { "a": 67, "b": [12, 15, 20], "c": { "hi": "greeting", "goodbye": "farewell" } }
print(nested["a"])
print(nested["b"])
sum = 0
for num in nested["b"]:
    sum += num
print(sum)
print(nested["c"]["hi"])
print(nested["c"]["hi"][4])
```

Expected output:

```
67
[12, 15, 20]
47
greeting
t
```

Copying Lists/Dictionaries

```python
import copy
nested = { "a": 67, "b": [12, 15, 20], "c": { "hi": "greeting", "goodbye": "farewell" } }
other = nested # other and nested are the same dictionary
other2 = nested.copy() # other2 is copy of first level of value
other3 = copy.deepcopy(nested) # other3 is completely separate copy
```

Read Data File from URL

```python
from urllib.request import urlopen # once at top

with urlopen("https://example.com/data.txt") as data:
   lines = data.read().decode('utf-8').rstrip().split("\n")
print(lines)
```

Expected output: a list whose elements are the lines of the file designated by the URL.

Jupyter Markdown

```
# Big Idea
Here are things to know:
 1. It’s big
 1. It’s fun
 1. It’s __[amazing](https://josephus.hsutx.edu)__
Other thoughts:
 - It’s *cool*
 - It’s **sleek**
Some code:
‘
for k in lst:
    print(k)
‘
```

Expected output: Big Idea
Here are things to know:

1. It’s big
2. It’s fun
3. It’s amazing

- It’s *cool*
- It’s **sleek**

Some code:

```
for k in lst:
    print(k)
```

Pandas Commands

```python
import pandas as pd

# load csv as dataframe
df = pd.read_csv("https://example.com/fun.csv")
df.head(10) # show first 10 rows of dataframe
df.tail()   # show last 5 rows of dataframe

from vega_datasets import data # use vega_datasets
movies = data.movies()   # import movies data set
movies.shape             # see # of rows and columns
movies.describe()        # see summary stats
movies.corr()            # see pairwise correlations

# select one or more columns from dataframe
movies['Major_Genre']
movies[['MPAA_Rating','Major_Genre']]

# Select rows 0 to 9 skipping by 2's and columns 0, 1, 7,
    and 8
movies.iloc[0:10:2,[0,1,7,8]]

# select rows with PG rating
movies[movies["MPAA_Rating"]=="PG"]]

# treat all titles as strings
movies['Title'].asType(str)

# Set/reset index
movies.set_index(['Title'], inplace=True)
movies.reset_index(inplace=True)

# Drop rows or columns
movies.drop(movies.index[0:6]) # delete 1st six rows
movies.drop(['Director'], axis=1) # delete column

# group by MPAA_Rating
group_by_rating = movies.groupby("MPAA_Rating")

# show average US_Gross when grouped by rating
group_by_rating["US_Gross"].agg("mean")

# enable viewing plots in Jupyter
%matplotlib inline

# Create line chart and bar chart
group_by_rating.plot("index='Year', columns='MPAA_Rating',
    values='US_Gross'")
group_by_rating["US_Gross"].agg("mean").plot(kind="bar")
```

Output too complex to show.

API Commands

```
import requests
import json
import pandas as pd

result = requests.get("https://api.example.com/v1/all")
print(result.status_code)
print(result.text)
all_json = result.json()
print(json.dumps(all_json, indent=4))
df = pd.read_json(result.text)
df.describe()
```

Expected output: an indented JSON string returned by the API endpoint. Followed by the result of the pandas describe() function for the same data.