## Python

1. What is Python
    a. Multi-purpose(Web, GUI, Scripting)
    b. Object oriented
    c. Interpreted
    d. Strongly typed and Dynamically typed
    e. Focus on readability and productivity
    f. Features
        i. Batteries included
        ii. Everything is an Object
        iii. Interactive Shell
        iv. Strong Introspection
        v. Cross Platform
        vi. CPython, Jython, IronPython, PyPy
    g. Who uses python
        i. Google.
        ii. NASA
        iii. Instagram
        iv. Shopify
    h. Releases
        i. Created in 1989 by Guido Van Rossum
        ii. Python 1.0 reqleased in 1994
        iii. Python 2.0 released in 2000
        iv. Python 3.0 released in 2008
        v. Current version 3.6.4
2. Syntax

```
#!/usr/bin/env python
print "Hello World!"
```

    a.
    b. Indentation
        i.

```
# Python code
if foo:
    if bar:
        baz(foo, bar)
    else:
        qux()
```

        ii.

1. Python code

3. Comments

```
# A traditional one line comment

"""
Any string not assigned to a variable is
considered a comment.
This is an example of a multi-line comment.
"""

"This is a single line comment"
```

a.

4. Data Types
    a. Strings

```
# This is a string
name = "Nowell Strite (that\"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = '''You can find me online
on sites like GitHub and Twitter.'''

# This is also a multi-line string
bio = """If you don't find me online
you can find me outside."""
```

i.

    b. Numbers

```
# Integers Numbers
year = 2010
year = int("2010")

# Floating Point Numbers
pi = 3.14159265
pi = float("3.14159265")

# Fixed Point Numbers
from decimal import Decimal
price = Decimal("0.02")
```

i.

    c. Null

i.

```
optional_data = None
```

d. Lists

```
# Lists can be heterogeneous
favorites = []

# Appending
favorites.append(42)

# Extending
favorites.extend(["Python", True])

# Equivalent to
favorites = [42, "Python", True]
```

i.

```
numbers = [1, 2, 3, 4, 5]

len(numbers)
# 5

numbers[0]
# 1

numbers[0:2]
# [1, 2]

numbers[2:]
# [3, 4, 5]
```

ii.

e. Dictionaries

```
person = {}

# Set by key / Get by key
person['name'] = 'Nowell Strite'

# Update
person.update({
    'favorites': [42, 'food'],
    'gender': 'male',
    })

# Any immutable object can be a dictionary key
person[42] = 'favorite number'
person[(44.47, -73.21)] = 'coordinates'
```

i.

f. Dictionary methods

```
person = {'name': 'Nowell', 'gender': 'Male'}

person['name']
person.get('name', 'Anonymous')
# 'Nowell Strite'

person.keys()
# ['name', 'gender']

person.values()
# ['Nowell', 'Male']

person.items()
# [['name', 'Nowell'], ['gender', 'Male']]
```

i.

g. Booleans

```
# This is a boolean
is_python = True

# Everything in Python can be cast to boolean
is_python = bool("any object")

# All of these things are equivalent to False
these_are_false = False or 0 or "" or {} or []
or None

# Most everything else is equivalent to True
these_are_true = True and 1 and "Text" and
{'a': 'b'} and ['c', 'd']
```

i.

5. Operators

a. Arithmetic

i.
```
a = 10          # 10
a += 1          # 11
a -= 1          # 10

b = a + 1       # 11
c = a - 1       # 9

d = a * 2       # 20
e = a / 2       # 5
f = a % 3       # 1
g = a ** 2      # 100
```

b. String manipulations

i.
```
animals = "Cats " + "Dogs "
animals += "Rabbits"
# Cats Dogs Rabbits

fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# Apple, Banana, Orange

date = '%s %d %d' % ('Sept', 11, 2010)
# Sept 11 2010

name = '%(first)s %(last)s' % {
    'first': 'Nowell',
    'last': 'Strite'}
# Nowell Strite
```

c. Logical comparisons

i.
```
# Logical And
a and b

# Logical Or
a or b

# Logical Negation
not a

# Compound
(a and not (b or c))
```

d. Identity comparisons

```
# Identity
1 is 1 == True

# Non Identity
1 is not '1' == True

# Example
bool(1) == True
bool(True) == True

1 and True == True
1 is True == False
```

i. True

ii. Arithmetic comparisons

```
# Ordering
a > b
a >= b
a < b
a <= b

# Equality/Difference
a == b
a != b
```

1.

6. Control flow

a. Conditionals

```
grade = 82
if grade >= 90:
    if grade == 100:
        print 'A+'
    else:
        print "A"
elif grade >= 80:
    print "B"
elif grade >= 70:
    print "C"
else:
    print "F"
```

i. `# B`

b. Loops - For

i.

```
fruits = ['Apple', 'Orange']

for fruit in fruits:
    print fruit
```

```
for x in range(10): #0-9
    print x
```

1.

ii. Expanded for loop

```
states = {
    'VT': 'Vermont',
    'ME': 'Maine',
    }

for key, value in states.items():
    print '%s: %s' % (key, value)
```

1.

iii. While loop

```
x = 0
while x < 100:
    print x
    x += 1
```

1.

7. List comprehension

i.

```
odds = [ x for x in range(50) if x % 2 ]
```

ii.

```
odds = []
for x in range(50):
    if x % 2:
        odds.append(x)
```

8. Functions

a. Basic function

i.
```
def my_function():
    """Function Documentation"""
    print "Hello World"
```

b. Function arguments

i.
```
# Positional
def add(x, y):
    return x + y

# Keyword
def shout(phrase='Yipee!'):
    print phrase

# Positional + Keyword
def echo(text, prefix=''):
    print '%s%s' % (prefix, text)
```

c. Arbitrary arguments

i.
```
def some_method(*args, **kwargs):
    for arg in args:
        print arg

    for key, value in kwargs.items():
        print key

some_method(1, 2, 3, name='Numbers')
```

d. Fibonacci

i.
```
def fib(n):
    """Return Fibonacci up to n."""
    results = []
    a, b = 0, 1
    while a < n:
        results.append(a)
        a, b = b, a + b
    return results
```

ii.

```
def fib():
    """Yield Fibonacci."""
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b
```

9. Classes
    a. Class declaration

i.

```
class User(object):
    pass
```

    b. Class attributes

i.

```
class User(object):
    name = None
    is_staff = False
```

    c. Class methods

i.

```
class User(object):
    is_staff = False

    def __init__(self, name='Anonymous'):
        self.name = name
        super(User, self).__init__()

    def is_authorized(self):
        return self.is_staff
```

    d. Class instantiation and attribute access

i.

```
anonymous = User()
print user.name
# Anonymous


print user.is_authorized()
# False
```

    e. Class inheritance

i.
```
class SuperUser(User):
    is_staff = True
```
ii.
```
nowell = SuperUser('Nowell Strite')
print user.name
# Nowell Strite
print user.is_authorized()
# True
```

f. Pythons way
   i. No real private attributes/functions
   ii. Private attributes start (but do not end) with double underscores.
   iii. Special class methods start and end with double underscores.
   iv. __init__, __doc__, __cmp__, __str__

10. Imports
    a. Allows code isolation and re-use
    b. Adds references to variables/classes/functions/etc. into current namespace
    c.
```
# Renaming imports
from datetime import date
from my_module import date as my_date

# This is usually considered a big No-No
from datetime import *
```
```
# Imports the datetime module into the
# current namespace
import datetime
datetime.date.today()
datetime.timedelta(days=1)

# Imports datetime and addes date and
# timedelta into the current namespace
from datetime import date, timedelta
date.today()
timedelta(days=1)
```

11. Error handling

```
import datetime
import random

day = random.choice(['Eleventh', 11])
try:
    date = 'September ' + day
except TypeError:
    date = datetime.date(2010, 9, day)
else:
    date += ' 2010'
finally:
    print date
```

a.

12. Documentation
    a. Docstrings

```
def foo():
    """
    Python supports documentation for all modules,
classes, functions, methods.
    """
    pass

# Access docstring in the shell
help(foo)

# Programatically access the docstring
foo.__doc__
```

i.

13. Tools
    a. Web frameworks
        i. Django
        ii. Flask
        iii. Pylons
        iv. turboGears
        v. Zope
        vi. Grok
    b. IDE's
        i. Emacs
        ii. Vim
        iii. Komodo
        iv. PyCharm
        v. Eclipse
    c. Package management
    d. Easy_install pip

e. Pip install django

14. Decorators

a.
```
def my_decorator(some_function):

    def wrapper():

        num = 10

        if num == 10:
            print("Yes!")
        else:
            print("No!")

        some_function()

        print("Something is happening after some_function() is called.")

    return wrapper
```

b.
```
from decorator07 import my_decorator


@my_decorator
def just_some_function():
    print("Wheee!")

just_some_function()
```

15. Lambda functions
    a. Anonymous function is a function that is defined without a name, in Python anonymous functions are defined using the lambda keyword.
    b. lambda arguments: expression
    c. double = lambda x: x * 2
16. Generators
    a. Python generators are a simple way of creating iterators.
    b.
```
def yrange(n):
    i = 0
    while i < n:
        yield i
        i += 1
```

c.

```
>>> y = yrange(3)
>>> y
<generator object yrange at 0x401f30>
>>> y.next()
0
>>> y.next()
1
>>> y.next()
2
>>> y.next()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

**Links**:
https://www.tutorialspoint.com/python/index.htm

**Assignments:**

**Set 1:**

1. With a given integral number n, write a program to generate a dictionary that contains (i, i*i) such that is an integral number between 1 and n (both included). and then the program should print the dictionary.
   Suppose the following input is supplied to the program:
   8
   Then, the output should be:
   {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
2. Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.
   Suppose the following input is supplied to the program:
   34,67,55,33,12,98
   Then, the output should be:
   ['34', '67', '55', '33', '12', '98']
   ('34', '67', '55', '33', '12', '98')
3. Define a class which has at least two methods:
   getString: to get a string from console input
   printString: to print the string in upper case.
   ~~Also please include simple test function to test the class methods.~~
4. Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.
   Suppose the following input is supplied to the program:
   without,hello,bag,world
   Then, the output should be:

bag,hello,without,worl

5. Write a program to generate and print another tuple whose values are even numbers in the given tuple (1,2,3,4,5,6,7,8,9,10).
6. Define a class named Circle which can be constructed by a radius. The Circle class has a method which can compute the area.
7. Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the user name of a given email address. Both user names and company names are composed of letters only.
   If the following email address is given as input to the program:
   john@google.com
   Then, the output of the program should be:
   john
8. Get a list of numbers from users and print the smallest odd number.
9. Read and print number of lines, words and characters in the given file.
10. Let 'a' be the list of users who likes a post! I want to get displayed as below.
    a. eg 1 :- a = []
    b. Output : Nobody likes This
    c. eg 2 :- a = ['Alice']
    d. Output : Alice likes This
    e. eg 3 :- a = ['Alice','Bob']
    f. Output : Alice and Bob likes This
    g. eg 4 :- a = ['Alice', 'Bob', 'Charls']
    h. Output : Alice, Bob and Charls likes This
    i. eg 5 :- a = ['Alice', 'Bob', 'Charls','Denny']
    j. Output : Alice, Bob and 2 others likes This
    k. eg 6 :- a = ['Alice', 'Bob', 'Charls','Denny','Emely']
    l. Output : Alice, Bob and 3 others likes This

**Set 2**

2. Learn below packages:
    a. os
    b. sys
    c. re
    d. math
    e. random
    f. urllib2
    g. smtplib
    h. datetime
    i. zlib
    j. unittest
    k. requests
    l. scrapy
    m. BeautifulSoup
    n. nltk
    o. nose

**Set 3**

1. Write a Python program to print yesterday, today, tomorrow.
2. Get email input from user and check it is valid or not(Output: valid/ Invalid)
3. Write a program to find all mobile number inside a string.

4. Fetch data from the URL https://www.w3schools.com/xml/simple.xml and print name, price of each item.
5. Create a class with functions to add, subtract, multiply and division. Also write unit test for each functions.
6. Write a function to send email to given email address.

**Set 4**

1. Fetch image url, country name from the URL http://example.webscraping.com/ using beautifulsoup

**Set 5:**

1. Learn pep8 - https://www.python.org/dev/peps/pep-0008/
2. Create a Python SDK for few API's mentioned here https://jsonplaceholder.typicode.com/
    a. Features:
        i. Post
            1. Get posts
            2. Get posts based on the user
            3. Get post detail
        ii. Comment
            1. Get comments
            2. Get comments based on the post
        iii. Write unittest for minimum two functions
    b. **Note**: Use facebook-SDK for reference https://github.com/mobolic/facebook-sdk/