General Questions

Q1. What is Python? What are the benefits of using Python?

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling and automatic memory management which help in modelling real-world problems and building applications to solve these problems.

Q2. What are the key features of Python?

- Python is an interpreted language. That means that, unlike languages like *C* and its variants, Python does not need to be compiled before it is run.
- Python is dynamically typed, this means that you don't need to state the types of variables when you declare them or anything like that.
- Python is well suited to object orientated programming in that it allows the definition of classes along with composition and inheritance.
- In Python, functions are first-class objects. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects
- Writing Python code is quick but running it is often slower than compiled languages. Fortunately, Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The numpy package is a good example of this, it's really quite quick because a lot of the number crunching it does isn't actually done by Python
- Python finds use in many spheres – web applications, automation, scientific modeling, big data applications and many more. It's also often used as "glue" code to get other languages and components to play nice.
- Python is a case sensitive language.

Q3. What is pep 8?

PEP stands for Python Enhancement Proposal. It is a set of rules that specify how to format Python code for maximum readability. Examples:

- Indentation: 4 spaces per indentation level (do not use tabs)
- Maximum line length: 79 characters (using **backslash** "\" to connect multiple lines)
- Imports: import statement should usually be on separate lines

Q4. Is indentation required in python?

Indentation is necessary for Python. It specifies a block of code. All code within loops, classes, functions, etc is specified within an indented block. It is usually done using four space characters. If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

Q5. How is memory managed in Python?

- Memory management in python is managed by Python private heap space. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.
- The allocation of heap space for Python objects is done by Python's memory manager.
- Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

Q6. How to comment code in Python?

Consecutive Single-line Comments (pound character) to comment single line

''' triple quotation marks to comment multiple lines.

Q7. What is PYTHONPATH?

It is an environment variable which is used when a module is imported. Whenever a module is imported, PYTHONPATH is also looked up to check for the presence of the imported modules in various directories. The interpreter uses it to determine which module to load.

Q8. What are python modules? Name some commonly used built-in modules in Python?

Python modules are files containing Python code. This code can either be functions, classes or variables. A Python module is a .py file containing executable code. Some of the commonly used built-in modules are: os, sys, math, random, datetime

Q9. How to import modules in python?

```
import array          #importing using the original module name
import array as arr    # importing using an alias name
from array import *    #imports everything present in the array module
```

Q10. Magic Methods

Use the **dir()** function to see the number of magic methods inherited by a class:

__new__()

the __new__() magic method is implicitly called before the __init__() method. The __new__() method returns a new object, which is then initialized by __init__().

__init__()

__init__ is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. (super().__init__() or Parent.__init__())

__str__()

It is overridden to return a printable string representation of any user defined class. default returns an object instance with address.

Magic Method	When it gets invoked (example)	Explanation
`__new__(cls [,...])`	`instance = MyClass(arg1, arg2)`	`__new__` is called on instance creation
`__init__(self [,...])`	`instance = MyClass(arg1, arg2)`	`__init__` is called on instance creation
`__cmp__(self, other)`	`self == other, self > other, etc.`	Called for any comparison
`__pos__(self)`	`+self`	Unary plus sign
`__neg__(self)`	`-self`	Unary minus sign
`__invert__(self)`	`~self`	Bitwise inversion
`__index__(self)`	`x[self]`	Conversion when object is used as index
`__nonzero__(self)`	`bool(self)`	Boolean value of the object
`__getattr__(self, name)`	`self.name # name doesn't exist`	Accessing nonexistent attribute
`__setattr__(self, name, val)`	`self.name = val`	Assigning to an attribute
`__delattr__(self, name)`	`del self.name`	Deleting an attribute
`__getattribute__(self, name)`	`self.name`	Accessing any attribute
`__getitem__(self, key)`	`self[key]`	Accessing an item using an index
`__setitem__(self, key, val)`	`self[key] = val`	Assigning to an item using an index
`__delitem__(self, key)`	`del self[key]`	Deleting an item using an index
`__iter__(self)`	`for x in self`	Iteration
`__contains__(self, value)`	`value in self, value not in self`	Membership tests using `in`
`__call__(self [,...])`	`self(args)`	"Calling" an instance
`__enter__(self)`	`with self as x:`	`with` statement context managers
`__exit__(self, exc, val, trace)`	`with self as x:`	`with` statement context managers
`__getstate__(self)`	`pickle.dump(pkl_file, self)`	Pickling
`__setstate__(self)`	`data = pickle.load(pkl_file)`	Pickling

Q11. What are docstrings in Python? __doc__?

Python docstrings are the string literals that appear right after the definition of a function, method, class, or module. It is accessible from the doc attribute __doc__ or built-in function: help(). dir() function returns all properties and methods of the specified object. help() function just returns its docstring.

Q12. What is self in Python?

Self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. It helps to differentiate between the methods and attributes of a class with local variables. The self variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

Q13. What are deep copy (copy.deepcopy)? shallow copy (copy.copy)?

A shallow copy constructs a new object and then (to the extent possible) inserts **references** into it to the objects found in the original. A deep copy constructs a new object and then, recursively, inserts **copies** into it of the objects found in the original.

Q14. What are local variables and global variables in Python? How to access a global variable in a function?

Global Variables: Variables declared outside a function or in global space are called global variables. These variables can be accessed by any function in the program. **GLOBAL** Keyword is used to access/modify the variables out of the current scope.

Local Variables: Any variable declared inside a function is known as a local variable. This variable is present in the local space and not in the global space.

Q15. What is type conversion in Python?

Type conversion refers to the conversion of one data type into another.

int() – converts any data type into integer type

float() – converts any data type into float type

ord() – converts characters into integer

hex() – converts integers to hexadecimal

oct() – converts integer to octal

tuple() – This function is used to convert to a tuple.

set() – This function returns the type after converting to set.

list() – This function is used to convert any data type to a list type.

dict() – This function is used to convert a tuple of order (key,value) into a dictionary.

str() – Used to convert integer into a string.

complex(real,imag) – converts real numbers to complex(real,imag) numbers.

Q16. What are the built-in types of python?

Integers

Floating-point

Complex numbers

Strings

Boolean

Built-in functions

Q17. All built-in functions of Python

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

map(function, iter)

In general, map function has two arguments. The first one is a function, The second is an iterable object. map function applies the given function to each item of a given iterable and return a map object which is an iterator.

help()

help() function is to display the documentation string (__doc__) and also facilitates you to see the help related to modules, keywords, attributes, etc.

dir()

The dir() function returns all properties and methods of the specified object, without the values. This function will return all the properties and methods, even built-in properties which are default for all objects.

Q18. All String functions of Python

refer to file “00_04_StringFunctions.py”

myString.capitalize() # capitalize the first letter of string

myString.upper() # translate to all uppercase letters

myString.lower() # translate to all lower case letters

myString.title() # capitalize the first character of each words

myString.swapcase() # uppercase to lowercase; lowercase to uppercase

myString.find(“i”) #return the index of first finding substring, -1 if not found.

myString.index(“i”) # same with find() but return error if not found.

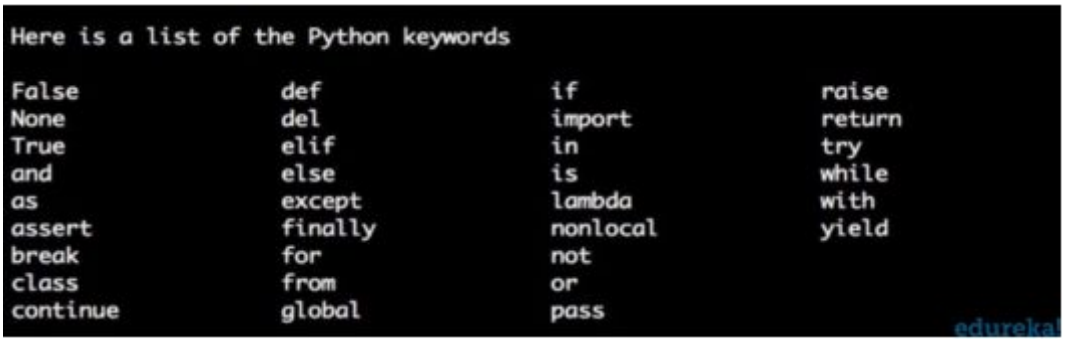
myString.replace(“s”, “d”, 2) #src, dest, times

myString.split(“i”, 2) # delimiter, times myString.rsplit(“i”,2) # starting at the right side.

string connection: **a + b** , **(“%s %s”)%(a, b)** , **“ “.join([a, b])**

string to ASCII: print(ord(“a”)) print(chr(ord(“a”))) # **<=122**; **%122 + 96**

Q19. All keywords of Python




```
import keyword

print(keyword.kwlist)

#this will get you the list of all keywords in python.

print(keyword.iskeyword('try'))

#this will return true, if the mentioned name is a keyword.
```

Q20. What does this mean: *args, **kwargs? Why would we use it?

We use *args when we aren't sure how many arguments are going to be passed to a function. **kwargs is used when we don't know how many key-word arguments will be passed to a function.

Q21. What is the difference between list and tuples in Python?

Lists are mutable i.e they can be edited.	Tuples are immutable (tuples are lists which can't be edited).
Lists are slower than tuples.	Tuples are faster than list.
Syntax: list_1 = [10, 'Chelsea', 20]	Syntax: tup_1 = (10, 'Chelsea' , 20)

Q22. Why is List slower than Tuple?

List is slower than Tuple because Tuple has a smaller memory. List is stored in two blocks of memory (one is fixed size and the other is variable size for storing data) and Tuple is stored in one memory block. So creating a List is slower than creating Tuple.

- Tuple created with () **round brackets**
- List created with [] **square brackets**
- Dictionary created with {} **curly brackets**

Q23. What is the difference between Python Arrays and lists?

lists can hold any data type elements	arrays hold only a single data type element
Lists is the basic syntax of Python	Array need to be imported firstly
Lists consume more memory as they are allocated a few extra elements to allow for quicker appending of items.	Since arrays stay the size that they were when they were first initialized, they are compact.

```
import array as arr

My_Array=arr.array('i',[1,2,3,4])
```

```
My_list=[1,'abc',1.20]
```

Q24. What advantages do NumPy arrays offer over Python lists?

We use python NumPy array instead of a list because of the below three reasons:

1. Less Memory
2. Fast
3. Convenient

The very first reason to choose python NumPy array is that **it occupies less memory** as compared to list. Then, **it is pretty fast** in terms of execution and at the same time, it is very convenient to work with NumPy (Numpy array has the various functions, methods, and variables, to **ease** our task of matrix computation.). So these are the major advantages that Python NumPy array has over list.

Q25. What is a lambda function? When is it useful and when not?

A lambda function is a small **anonymous** function. **It can take any number of arguments but can have just one statement**. Example:

```
a = lambda x,y : x + y

print(a(5, 6))
```

A lambda is much more readable than a full function since it can be written in-line. Hence, it is a good practice to use lambdas when the function expression is small.

The beauty of lambda functions lies in the fact that they return function objects. This makes them helpful when used with functions like map or filter which require function objects as arguments.

Lambdas aren't useful when the expression exceeds a single line.

Q26. How can the ternary operators be used in python?

The Ternary operator is the operator that is used to show the conditional statements. This consists of the true or false values with a statement that has to be evaluated for it. Example: `big = x if x < y else y`

Q27. How does break, continue and pass work?

Break	Allows loop termination when some condition is met and the control is transferred to the next statement.
Continue	Allows skipping some part of a loop when some specific condition is met and the control is transferred to the beginning of the loop
Pass	Used when you need some block of code syntactically, but you want to skip its execution. This is basically a null operation. Nothing happens when this is executed.

Q28. What are negative indexes and why are they used?

The sequences in Python are indexed and it consists of the positive as well as negative numbers. The index for the negative number starts from '-1' that represents the last index in the sequence and '-2' as the penultimate index and the sequence carries forward like the positive number.

The negative index is used to remove any new-line spaces from the string and allow the string to except the last character that is given as S[:-1]. The negative index is also used to show the index to represent the string in correct order.

Q29. What does [::-1] do?

[::-1] is used to reverse the order of an array or a sequence.

[::-1] reprints a reversed copy of ordered data structures such as an array or a list. the original array or list remains unchanged.

Q30. How can you randomize the items of a list in place in Python?

```
from random import shuffle

x = ['Keep', 'The', 'Blue', 'Flag', 'Flying', 'High']

shuffle(x)

print(x)
```

The output of the following code is as below.

```
['Flying', 'Keep', 'Blue', 'High', 'The', 'Flag']
```

Q31. How can you generate random numbers in Python?

Random module is the standard module that is used to generate a random number. The method is defined as:

```
import random

random.random
```

Q32. What is the difference between range & xrange?

For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that **range returns a Python list object and xrange returns an xrange object.**

This means that xrange doesn't actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really **gigantic** range you'd like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really **memory sensitive system** such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It's a memory hungry beast.

Q33. What is pickling and unpickling?

Python pickle module is used for serializing and de-serializing a Python object structure. Any objects in Python can be pickled so that it can be saved on disk.

```
import pickle
data = {"key" : "value"}
file = open('data.txt', 'wb') # must be binary mode
pickle.dump(data, file)
```

```
file.close()
readfile = open('data.txt', 'rb') # must be binary mode
dataloaded = pickle.load(readfile)
readfile.close()
print(dataloaded)
```

Q34. Iterator

An iterator is an object that contains a countable number of values. In Python, an iterator is an object which implements the iterator protocol, which consists of the method __iter__() and __next__(). Refer to **00_02_03_IteratorAndGenerator.py**

Q35. Keyword 'yield', Generator and Generator Expression

Generator is an easy method to build an iterator with the keyword 'yield'. When the function contains a 'yield' statement, it becomes a generator function and returns an iterator object. 'yield' statement pauses the function saving all its states and later continues from there on successive calls. Generator Expression is to create anonymous generator functions, like: **myGenerator = (x**2 for x in range(5))**

Q36. Decorator

Decorators is a wrapper of function in order to extend the behavior of wrapped function, without permanently modifying it. **Decorators use the sign of "@"**. Since decorators are a callable object so class, function could be decorators. Some real cases: **recording the run time of function**, **log information of function**.

Q37. monkey patching

The term of monkey patch refers to dynamic modifications of a class or module at run-time.

Q38. What is the difference between eval() and exec()?

eval() returns the run result but exec() not.

Q39. @property

@property is a built-in decorator for the property() function in Python. It is used to give “special” functionality to certain methods to make them act as getters, setters, deleters when we defined properties in a class (For encapsulation)

Q40. Access class attribute

blu.__class__.attribute
Bird.attribute

Q41. Code practise

一行代码实现1--100之和： sum(range(1,101))

字典如何删除键和合并两个字典

del dict[“name”]

dict.update(dict2)

列表去重的方法

list = [‘a’, ‘c’, ‘a’, ‘c’]

s = set(list)

list = [x for x in s]

限定类的属性：

```
__slots__ = ('name', 'age') # 用tuple定义允许绑定的属性名称
```

with方法打开处理文件

```
with open("1.txt") as file:
    data = file.read()
```

如果按照常规的f.open写法，我们需要try,except,finally，做异常判断，并且文件最终不管遇到什么情况，都要执行finally f.close()关闭文件，with方法帮我们实现了finally中f.close()

列表[1,2,3,4,5],请使用map()函数输出[1,4,9,16,25]，并使用列表推导式提取出大于10的数，最终输出[16,25]

list = [1,2,3,4,5]

def fn(x):

    return x*x

r = map(fn, list)

f = [x for x in r if x > 10]

生成随机整数、随机小数、0--1之间小数方法

随机整数：random.randint(a,b),生成区间内的整数. random.randint(10, 20)

随机小数：np.random.randn(5) 5个随机小数

0-1随机小数：random.random(),括号中不传参

assert（）方法，断言成功，则程序继续执行，断言失败，则程序报错

字符串去重并从小到大排序输出"adfjl"

s = "ajldjlajfdljfddd"

s = set(s) #用字符串构建集合去重

s = list(s) # 集合转为列表然后进行排序

s.sort(reverse=false) # 从小到大进行排序

res = “”.join(s)

lambda函数实现两个数相乘

s = lambda a,b: a*b

字典根据键从小到大排序: sorted(dict.key())

filter方法求出列表所有奇数并构造新列表，a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
def fn(a):
    return a%2 == 1
res = filter(fn, a)
```

列表推导式求列表所有奇数并构造新列表，a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
res = [x for x in a if x%2 == 1]
```

正则re.complie作用

re.compile是将正则表达式编译成一个对象，加快速度，并重复使用

a=（1，）b=(1)，c=("1") 分别是什么类型的数据？tuple, int, str

两个**列表**[1,5,7,9]和[2,2,6,8]**合并**为[1,2,2,3,6,7,8,9]

```
l1 = [1,5,7,9]
l2 = [2,2,6,8]
l1.extend(l2)
l1.sort(reverse=False)
```

时间戳

```
import datetime
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
```

统计图（条形图、折线图等）绘制的开源库 matplotlib

[[1,2],[3,4],[5,6]]一行代码展开该列表，得出[1,2,3,4,5,6]

[y for x in a for y in x]

举例说明异常模块中try except else finally的相关意义

try..except..else没有捕获到异常，执行else语句

try..except..finally不管是否捕获到异常，都执行finally语句

zip()函数在运算时，会以一个或多个序列（可迭代对象）做为参数，返回一个元组的列表。同时将这些序列中并排的元素配对。

a="张明 98分"，用re.sub，将98替换为100

re.sub(r"\d+", "100", a)

提高python运行效率的方法

1、使用生成器，因为可以节约大量内存

2、循环代码优化，避免过多重复代码的执行

3、核心模块用Cython PyPy等，提高效率

4、多进程、多线程、协程

5、多个if elif条件判断，可以把最有可能先发生的条件放到前面写，这样可以减少程序判断的次数，提高效率

排序算法：冒泡，

```
    for i in range(len(array)):
        for j in range(len(array)-i-1):
            if array[j] > array[j+1]:
                array[j], array[j+1] = array[j+1], array[j]
```

插入，

```
    for i in range(1,len(array)):
        j = i
        while j > 0 and array[j] < array[j-1]:
            array[j-1], array[j] = array[j], array[j-1]
            j -=1
```

选择

```
  for i in range(len(array)):
    min_idx = i
    for j in range(i+1, len(array)):
      if array[min_idx] > array[j]:
        min_idx = j
```

```
        array[i], array[min_idx] = array[min_idx], array[i]
```

字符偏移计算：

```
def caesarCipherEncryptor(string, key):
            key %= 26
            myNewStr = []
    for i in string:
        j = ord(i) + key
                    c = chr(j) if j <= 122 else chr(96+j%122)
        myNewStr.append(c)
    return ''.join(myNewStr)
```

保留两位小数: **round**(float(x), 2)

列出几种魔法方法并简要介绍用途

__init__:对象初始化方法

__new__:创建对象时候执行的方法，单列模式会用到

__str__:当使用print输出对象的时候，只要自己定义了__str__(self)方法，那么就会打印从在这个方法中return的数据

__del__:删除对象执行的方法

举例sort和sorted对列表排序，list=[0,-1,3,-10,5,9]

list.sort(reverse=False) and sorted(list, reverse=False)

s="info:xiaoZhang 33 shandong",用正则切分字符串输出['info', 'xiaoZhang', '33', 'shandong']

|表示或，根据冒号或者空格切分：　　**re.split(r'\s|:', s)**

列举3条以上PEP8编码规范:

1、顶级定义之间空两行，比如函数或者类定义。

2、方法定义、类定义与第一个方法之间，都应该空一行

3、三引号进行注释

4、使用Pycharm、Eclipse一般使用4个空格来缩进代码

OOPS Interview Questions

Q1. Inheritance

Inheritance allows One class to gain all the members(say attributes and methods) of another class. Inheritance provides code reusability, makes it easier to create and maintain an application. The class from which we are inheriting is called superclass and the class that is inherited is called a derived / child class.
They are different types of inheritance supported by Python:
Single Inheritance – where a derived class acquires the members of a single super class.
Multiple inheritance – a derived class is inherited from more than one base class.
Multilevel inheritance – a derived class d1 is inherited from base class base1, and class base1 is inherited from class base2.

Q2. Encapsulation

Encapsulation means binding the code and the data together. A Python class in an example of encapsulation

Q3. Polymorphism

Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

Q4. Namespace

A namespace is basically a system to make sure that all the names in a program are unique and can be used without any conflict.

在python中，如果要访问某一个对象（包括变量，模块，方法等）都是会去

namespace中根据对象名称去检索，这里涉及到一个检索顺序，称为：LEGB

namespace type	namespace description
locals	函数的namespace，只记录当前函数内的对象。
enclosing function	这个namespace针对的对象比较特殊，像闭包函数会有这样一个namespace，记录的是闭包函数所在函数内的对象。
globals	python模块的namespace，每个模块都有一个自己的namespace，记录模块内的class、function等
__builtins__	python内置的namespace，在python解释器启动的时候创建，有很多内置的函数

locals -->> enclosing function -->> globals -->> __builtins__

Q5. Private and Protected Modifiers

Python uses the '_' symbol to determine the access control for a specific data member or a member function of a class.

Protected Access Modifier:The members of a class that are declared protected are only accessible to a class derived from it. Data members of a class are declared **protected by adding a single underscore '_'** symbol before the data member of that class. It can be accessed but not do it.

Private Access Modifier: The members of a class that are declared private are accessible within the class only, private access modifier is the most secure access modifier. **Data members of a class are declared private by adding a double underscore '__' symbol** before the data member of that class. (_Class__Variable)

Q6. How to create an empty class in Python?

An empty class is a class that does not have any code defined within its block. It can be created using the ***pass*** keyword. However, you can create objects of this class outside the class itself. IN PYTHON THE PASS command does nothing when it's executed. it's a null statement.

Q7. What does an object() do?

The object() function returns an empty object. We cannot add any new properties or methods to this object. It is the base for all classes (built-in properties and methods).

Q8. What is the difference between Class attribute and Instance attribute?

re module

meta characters

[ ] open square brackets/close square brackets: a set of characters

**** backslash : a special sequence

. period, dot : any character

^ caret : starts with $ dollar : end with

* asterisk : zero or more occurrences

+ plus : one or more occurrences

{} curly brackets : exactly the specified number of occurrences.

| bar : either or

() round brackets : capture and group

special sequence

\A at the beginning of the string, r"\Aaaa"

\b at the beginning or end of a word, r"\baaa" or r"aaa\b"

\B matched but not at the beginning or end.

\d number 0~9 \D not 0~9

\s space \S not space

\w a~Z and 0~9, _ (underscore) \W not \w

\Z at the end. r"cc\Z"

sets

A set is a set of characters inside a pair of square brackets [ ]

[^ard] not a, r, d

[+] character +, any meta characters will miss their meaning just characters.

functions

findall()

x = re.findall("\w\d", string) # x is a list returned with all matched strings or empty list.

search()

x = re.search(“\s”, string) # x is a **matched object** or None

split()

uses a regex pattern to “split” a given string into a list.

x = re.split(“\s”, string) # returns a list

sub()

finds all substrings where the regex pattern matches and then replace them with a different string.

x = re.sub(“\s”, “|”, string) # replace string with what you input

subn()

similar with sub() but returned the replaced string and the number of replacements.

matched object

groups()

returns all matched, captured group

group(1)

returns the first captured group

start(1)

returns the start index of first captured group

end(1)

returns the end index of first captured group

span(1)

returns the start and end index of first captured group as a list

string: returns the raw string

lastIndex: returns the index of the last matched group, 2 means 2 matched groups.