UDACITY COURSE II INTRO TO COMPUTER PROGRAMMING

How to build a search engine:

1. Find data
2. Build the index
3. Rank the pages

Web crawler is a bot or spider that would collect data from web for the search engine.

We extract the first link on the web page. Web crawler finds the web page for our search engine from a “seed page” and following links on that page to find other pages. Each of the links could have a web page that is associated with it, which could lead to other web pages. This builds a collection of data for our web page.

A web page is a chunk of text that comes from internet into your browser.

The programs that we write would be an input to the python program which runs on the computer. Python is an interpreter, which runs our program, it interprets them execute the program that we wrote in the python language by running a program in a language that can be understood by the computer.

We need new language for computer because of ambiguity (same word can mean different things in different place) and verbosity (to describe a task, normal language would take up a huge amount of text).

Backus-Naur Form – produce sentence from non-terminal to terminal words (derivation), this is used to break down the sentences that we give to computer via python into useful statements like assignments, variables etc.

Expression = expression operator expression (expression can be number or expression operator expression) (operator +,-,\* etc)

Expression = (expression)

We cannot have spaces between a single number and we need to have decimals in order to get more accurate results. Any one digit in the calculation should be in decimal to make the answer accurate.

Speed of light = 299 792 458 m/sec ~ 30 cm/ns

Nanostick = a piece of wire with length equal to speed of light in nanosceonds = 30 cm

Admiral Grace Hopper wrote one of the pioneer language COBOL. Compiler is a program that produces other programs like python. Python is an interpreter. The difference between a compiler and interpreter is that compiler translates the whole code into a new program and then runs it, while interpreter does translation and execution at the same time.

Variables: stores value, the name refers to the value, value is assigned using the assignment statement.

If a variable that is not assigned with any value is used in computation in python, the answer is error.

For a string we can use both single and double quote in python. We can have a string assigned to a variable to print it. It is always better not to start a variable name with capital letters.

+ Along with strings result in concatenation. Eg

name = “Soumya Vikraman”

Print “hello “ + name

Python doesn’t understand if you try to add a string and a number, but it totally makes sense when you try to multiply a string with numbers

Eg: print “My name is” + 9 ---🡪 error

Print ”!” \* 38 --🡪 prints ! 38 times.

We can extract subsequences from string. – ‘string’<expn>

‘udacity’[0] -> ‘u’

‘udacity’[1+1] -> ‘a’

Name = ‘Dave’

Name[0] = ‘D’

Name[4] -> results in error

Name[-1] -> ‘e’ (last character in string) name[-2] -> ‘v’

Start stop

<string> [<expression> : <expression>] ---🡪 produces string [start:stop-1]

Number : number

Word = ‘assume’

Word[3] – u

Word[3:4] – u

Word[3:3] – no string output (this doesn’t result in error)

Word[4:6] – me

Word[4:] – me

Word[:2] – as

Word[:] – assume

Many individual values that are extracted from the word can go wrong but not with a bunch of character. Eg: s[0] csn result in error but s[0:2] might not.

>>> hello=''

>>> print(hello[0])

Traceback (most recent call last):

File "<pyshell#17>", line 1, in <module>

print(hello[0])

IndexError: string index out of range

>>> print(hello[0:2])

>>>

For any string s, s + s[0: -1 + 1] = s and s[:3] + s[3:] = s

FINDing strings in strings:

<search\_String>.find(<target\_string>) – if found, the function returns the first position in search string where the target string occurred. If not found the function returns -1. Uppercase and lowercase matters.

If s = ‘<any string>’, s.find(‘’) [empty string will always result in value 0 as there is nothing to look for an empty string in a string, doesn’t result in -1]

<search\_string>.find(<target\_string> , number) – number indicate the position from where the target string needs to be searched in the search string.

Danton = “De l’audace, encore de l’audace, toujours de l’audace”

Print danton.find(’audace’) – 5

Print danton.find(’audace’,0) – 5

Print danton.find(’audace’,5) – 5

Print danton.find(’audace’,6) – 25

Print danton[6:] - udace, encore de l’audace, toujours de l’audace

Print danton[25:] - audace, toujours de l’audace

Print danton.find(’audace’,25) – 25

Print danton.find(’audace’,26) – 47

Print danton[47:] – audace

Print danton.find(’audace’,48) - -1

<a href : <anchor tag hyperlink reference specified by the URL.

Clickable links are something that is encapsulated by <a href.

The syntax a, x = x, a is an assignment statement where all the values on the right side are assigned to the corresponding variables on the left side. For example, after a, b = 3, 4 the value of a is 3 and the value of b is 4.

What's really going on here is that the values on the right side are all packaged into a tuple, and expanded into the variables on the left side.

For an empty string s, accessing s[0] could cause an error. This is called the edge case. But at the same time we can access s[0:], which would not give an error.

# Given the variables s and t defined as:

s = 'udacity'

t = 'bodacious'

# write Python code that prints out udacious without using any quote characters in your code.

print s[:1] + t[2:]

print text.find(“zip”, text.find(“zip”)+1)

str function :

str(<number>) - > string

eg: str(89) = ‘89’

Given a variable, x, that stores the value of any decimal number, write Python code that prints out the nearest whole number to x. If x is exactly half way between two whole numbers, round up, so 3.5 rounds to 4 and 2.5 rounds to 3. You may assume x is not negative. Hint: The str function can convert any number into a string. eg str(89) converts the number 89 to the string '89'

Along with the str function, this problem can be solved using just the information introduced in unit 1.

x = 3.14159

>>> 3 (not 3.0)

x = 27.63

>>> 28 (not 28.0)

x = 3.5

>>> 4 (not 4.0)

x = 3.14159

x = x + 0.5

y = str(x)

z = y.find(".")

print y[0: z]

len(string) - > gives you the length of string.

Str[::-1] - > to print a string in reverse

To see if a given word is palindrome or not:

reverse\_word = word[::-1]

offset = word.find(reverse\_word)

is\_palindrome = offset

Indentation is very important in python as it indicates the beginning and ending of blocks and functions. MARK THEM CAREFULLY.

BNF is used to construct expressions in python. Grammar is also used by the python interpreter to break the strings that we put into their parts. The interpreter has to find out if the instruction provided is an assignment or a variable etc. when people implement compilers or interpreters, they start by writing the grammar and there is a tool that takes the grammar and turns into a program.

Procedures are a way to package the code so that we can reuse them more easily. Control is something that help us to make the decision and to repeat a set of action.

Procedural abstraction: to turn a set of repeating code into a procedure where the input is something that changes every time, while the output is something that is required as an output from the procedure.

+ is an example for procedural abstraction as it works on different inputs to generate different outputs. But it is a built in procedure.

def <procedure\_name> (parameters):

<block>

Parameters: are just inputs that are separated by commas. They can have no inputs. Then denoted - ()

Block is just a collection of statements. The code in the blocks are intended. Usually 4 spaces away, which makes it different from the other unintended code indicating that it is the part of block.

Return statement is used to return values from the procedure, it is a list of variables. The keyword is return.

Side effects are something that we use in the procedure, but not as an output. An example of that would be a print statement in the procedure. We can see the results happening but we wont get it as the output.

How to use a procedure: <procedure\_name> (input1, input2). Inputs are also called operands or arguments. Built in procedure is <search\_string>.find(target\_string, offset), where search\_string, offset and target\_string are the inputs to the procedure. [search\_string is a also called an object]

Python uses the value “None” to mean that there is no actual output or value. When you have a procedure that has no return statement after any calculation, python returns the statement None when you try to print the value of the procedure evaluated. You can also have print statements in the procedure, but still if there is no return statement in the procedure, it would show the output as none.

Def sum (a,b)

A = a + b

Print sum(5,3) // will give output None

Def sum (a,b)

A = a + b

Return a

Print sum (5,3) // will give output 8

Def sum(a,b):

A=a+b

A=2

B=123

Print sum(a,b)

Print a // would print a = 2.

This Is because the variables a and b in the procedure are local to themselves, In which the variable a gets modified. As soon as the control comes out of the procedure the variables a and b in the procedure no longer exists and if we try to print a outside of the procedure, you get the value of a to be 2.

Procedure composition: using procedure within the same procedure to obtain the result. Eg: square(square(x))

Github username: pythonlove143 email id: [sreejavikraman86@gmail.compassword](mailto:sreejavikraman86@gmail.compassword): unnipomi143!

Decision making:

For numbers we have the same operations as that of found in arithmetic expression like:

< , > , <= , >=, !=, == -> number <operand> number – output is a Boolean value (true/false)

We normally compare the values of the same type : eg: int to int and string to string. But comparing int to string won’t result in error, but produces consistent irrelevant results.

If statements:

If<test expression>:

<block>

Rest of the program

The block inside the if expression is evaluated to true or false. If true, then the block is executed, else the rest of the program is executed. Here indentation is very important as that determines the block. Now we can also use the construct else along with if.

If<test expression>:

<block>

Else:

<block>

Define a procedure, is\_friend, that takes a string as its input, and returns a Boolean indicating if the input string is the name of a friend. Assume I am friends with everyone whose name starts with D and no one else. You do not need to check for the lower case 'd'.

def is\_friend(name):

if(name[0] == 'D'):

return "True" // we can return strings from function.

else:

return "False"

print is\_friend('Diane')

To return Boolean values like True or False, we do not have to use quotes in the statements: return True / return False.

We can write the program without the if else block, which is:

return name[0] == ‘D’ -> this would evaluate the check and return either true or false.

To see for multiple comparison, we can also use a new variable called as or operator – directly use or.

The or operator will always look for the Boolean expression True. Even if one expression evaluates to True, the output will be true.

Print True or is\_this\_an\_error -> will print output as True, even though is\_this\_an\_error is an undefined variable.

But, print is\_this\_an\_error or True -> would result in error as is\_this\_an\_error is the first expression evaluated and it is undefined.

To find the biggest number between three numbers given as the inputs, we can use the function bigger which compares the two inputs a, b and the again use the same function bigger to compare the big out of c and output of a and b.

Print bigger(bigger(a,b),c)

We also have a built in function called max, that compares three numbers and produces the biggest of them.

Return Max(a,b,c)

Loops:

While loops: very similar to the if loops.

While <test expression>:

<block>

The difference between the if and while is if loop will be executed 0 or 1 time, but while loops will be executed as long as the expression evaluates to true.

A program that runs forever is an infinite loop, happens when the program/software hangs.

Define a procedure, factorial, that takes one number as its input and returns the factorial of that number.

def factorial(n):

i = n

if n == 0:

return 1

else:

while (n>1):

i = i \* (n-1)

n = n-1

return i

or

def factorial(n):

result = 1

while (n>=1)

result = result \* n

n = n-1

return result

Break statement: is used to break out from a while loop, even when the while condition is true. This part of code is present in the <block> portion of the while code. When the if statement in the while code is true, the execution breaks off from the while completely and starts executing the code after the while.

While<test expression>:

<code in while>

If(break test):

Break

<more code in while>

<code after while>

def print\_numbers(n):

I = 1

While i<=n:

Print i

I = I + 1

Can be rewritten as :

Def print\_numbers(n):

While True:

If(i>n)

Break

Print i

I = i+1

‘None’ can be returned from a function, which is not a string, but can be interpreted as false. So we can also write ‘None’ in an if condition as:

url = None

If None or if URL.

This means that the test condition for a Boolean doesn’t have to be true or false, can be different values. Any value other than an empty string is treated as True.

To use the web page to obtain the different URLs in the page, we use the function get\_page(url link)

 def get\_page(url): try: import urllib return urllib.urlopen(url).read() except: return '' Include this code above your get\_next\_target() procedure in your answer.

This prints the text from web page.

GET THE COMPLETE CODE FOR THUS FUNCTIONALITY OF THE SEARCH ENGINE.

Just like if, we have another construct called if not in python.

We cannot modify strings in python, can’t replace any character as it is immutable. For example, since I cannot modify an existing element in a list, I created a new temporary list called abacus\_temp.

abacus\_temp[i] = abacus[i][0:(length-k)] + "\*" + abacus[i][(length-k+1):]

to insert an element into the list you can either use insert function or range function.

If you write an if statement, you should always include a return statement as otherwise the else would be executed.

If you do not want anything to be returned from if(but you have to), to print from if, you can print the statement and return true.

Do not panic when you get a problem. First step in solving a problem is understanding the possible inputs. Then the next step would be to calculate the output of a program. Finally we have to solve the problem. Returning a number from a function is much different than printing out the numbers as return values can be used for further computation purpose, while that is not possible with a value just printed. To understand the relation between the inputs and outputs, we have to work out some examples.

If a problem is really complex and challenging, you can consider a systematic approach in solving them before starting to write the code. Always find a simple mechanical solution to fix the problem.

Def nextday(year,month,day):

“”” warning: this version incorrectly assumes that all months have 30 days””” – is treated by python as a docstring. When you use the function name nextday in idle, we can see the docstring appears in the idle screen.