Python For Data Science

--Sopan Shewale

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Session - 2

Agenda

- Variables
- Making Decisions
- Error Handling
- Strings
- Lists
- Dictionaries
- Sets
- Tuples
- Variables Expressions

- Mathematical Expressions
- Recording and processing Input from user (input)
- Iterators
- Collections
- Inbuilt functions
  - –In, with, range

variables

```
#!/usr/bin/python3

counter = 1000                  # An integer assignment
miles   = 1050.0                # A floating point
name    = "Hari Sadu"           # A string

print (counter)
print (miles)
print (name)

a = b = c = 1

print (a)
print (b)
print (c)

a, b, c = 1, 2, "john"

print (a)
print (b)
print (c)

#BMI - Body Mass Index = weight/(height)^2
weight = 61.0
height = 1.79
bmi = weight / height ** 2

print (bmi)
print (type(bmi))
```

```
1000
1050.0
Hari Sadu
1
1
1
1
2
john
19.038107424861895
<class 'float'>
```

variables.py

Task - 3

Develop a script called “my_variable earnings.py” to handle questions below

1. Create a variable *savings* equal to 1000
2. Create a variable *interest* equal to 1.10.
3. Use *savings* and *interest* variables to calculate the amount of money you end up earning after 8 years.
4. Store the result in a new variable, *earnings*
5. print out the value of *earnings*

Interact with Script

Python has many built-in functions - input

```
>>> input("Enter Your Name:")
Enter Your Name: Hari Sadu
' Hari Sadu'
>>>
```

To deal with numbers - you need to apply method called “int”

```
>>> input('Enter Your Age:')
Enter Your Age:40
'40'
>>> int(input('Enter Your Age:'))
Enter Your Age:40
40
>>> age = int(input('Enter Your Age:'))
Enter Your Age:40
>>> print(age)
40
>>> type(age)
<class 'int'>
```

Interact with Script (Cont ...)

```
#!/usr/bin/python3
'''Illustrate input and print.'''

applicant = input("Enter the applicant's name: ")
interviewer = input("Enter the interviewer's name: ")
time = input("Enter the appointment time: ")


print(interviewer, "will interview", applicant, "at", time)
```

```
Enter the applicant's name: John
Enter the interviewer's name: Hari
Enter the appointment time: 12:00
Hari will interview John at 12:00
```

interview.py

```
#!/usr/bin/python3

'''Conversion of strings to int before addition'''

xString = input("Enter a number: ")
x = int(xString)
yString = input("Enter a second number: ")
y = int(yString)
print('The sum of ', x, ' and ', y, ' is ', x+y, '.', sep='')
```

```
Enter a number: 23
Enter a second number: 78
The sum of 23 and 78 is 101.
```

addition.py

Task - 4

Develop a script “mean.py” which takes two numbers as input from user.

The script creates two variables (number_one & number_two).

It calculates mean, let us store the value in number_mean variable.

At the end, print number_mean.

```
$./mean.py
Enter First Number: 10
Enter Second Number: 20
The mean of both numbers is:  15.0
$_
```

Quiz

What's the output of following code snippets?

1.
```
>>> 100 * 3
```

2.
```
>>> 100 // 3.0
```

3.
```
>>> x = 25.0
>>> y = 75.0
>>> m = (x + y)/2
>>> print (m)
```

4.
```
>>> x =int(input())
100
>>> float(100)
```

5.
```
>>> 2 * 3 + 2 - 4
```

Built-in Constants

- **False**
  - The false value of the bool type
- **True**
  - The true value of the bool type
- **None**
  - The sole value of the type NoneType

```
>>> type(None)
<class 'NoneType'>
>>> type(False)
<class 'bool'>
>>> type(True)
<class 'bool'>
```

Making Decisions

```
#!/usr/bin/python3
# If the number is positive, we print an appropriate message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

```
3 is a positive number.
This is always printed.
This is also always printed.
```

if_demo.py

Making Decisions (Cont...)

```
#!/usr/bin/python3
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3

# Try these two variations as well.
# num = -5
# num = 0

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

```
Positive or Zero
```

if_else_demo.py

Making Decisions (Cont...)

```
#!/usr/bin/python3
# In this program,
# we check if the number is positive or
# negative or zero and
# display an appropriate message

num = 3.4

# Try these two variations as well:
# num = 0
# num = -4.5

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

```
Positive number
```

if_elif_demo.py

Task - 5

Develop a script “divide_numbers.py” which takes two numbers as input from user. The script creates two variables (number_one & number_two). The script prints “You tried to divide number_one by Zero” if number_two is Zero. If number_two is not zero then it prints integer part of quotient.

```
$ ./divide_numbers.py
Enter First Number:10
Enter Second Number:0
You tried to divide number_one by Zero
$
```

```
$ ./divide_numbers.py
Enter First Number:20
Enter Second Number:10
The division is: 2
```

Quiz

What's the output of following code snippets?

1.
```
>>> number = 100
>>> number > 100
```

2.
```
>>> number = 100
>>> number == 100.00
```

3.
```
#!/usr/bin/python3
number = 100
if number >= 100.00:
    if number < 1000:
        if number == 100.00
            print ("I am hundred")
else:
   print ("Don't dare to compare me with float")
```

4. Is Python Scripting Language?

5. Is Python High Level Language?

Quiz

What is the output of following code?

1.
```
>>> number = 10
>>> another_number = '20'
>>> number + another_number
```

2
```
>>> number = 10
>>> zero = 0
>>> zero/number
```

3
```
>>> number = 10
>>> zero = 0
>>> zero/division
```

Basic Operations

"and" and "or" are very frequently used operators

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

```
>>> 2 or 5
2
>>> 2 and 5
5
>>>
```

```
>>> if a < 11 and b < 16:
...     print ("I am true")
...
I am true
```

Quiz

What is the output of the following code?

1.
```
>>> if m == 100 and n == 50 and x == 25 and y == 10:
...     print ("i am true")
...
```

2
```
>>> m,n,x,y = 100, 50, 25, 10
>>> m == 100 and n == 50 and x == 25 and y == 10
```

3
```
>>> 100 and 50 and 25 and 10
```

Handling Errors

Two Types of Errors - [1] Syntax Error [2] Exceptions

Syntax Errors - Correct the Syntax!

Exceptions - Handle Exceptions, depends on the Logic of Application. Depends on Data of Application

IOError : If the file cannot be opened.

ImportError : If python cannot find the module

ValueError Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value

KeyboardInterrupt Raised when the user hits the interrupt key (normally Control-C or Delete)

EOFError Raised when one of the built-in functions (input() or raw_input()) hits an end-of-file condition (EOF) without reading any data

Handling Errors(Cont..)

```
>>> try:
...     d = 1/0
... except:
...     print ("I could not divide 1 by 0")
...
I could not divide 1 by 0
>>>
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> try:
...     d = 1/0
... except ZeroDivisionError:
...    print("Don't try to divide by 0")
...
Don't try to divide by 0
>>> try:
...     d = 10/2
... except ZeroDivisionError:
...    print("Don't try to divide by 0")
... else:
...    print("I am in else")
...
I am in else
>>> print(d)
5.0
>>>
```

Task - 6

Develop a script to take two numbers as input. Divide first number by second number. It should handle division by zero. The interaction of the script should look something similar to following

$divide.py

Enter First Number:10

Enter Second Number:0

You tried to divide 10 by zero

$*divide.py*

Enter First Number:10

Enter Second Number:2

The division is: 5.0

Collections

- A collection is similar to a basket that you can
  - add items
  - remove items
- The items can be same types or different types - it depends on collecton

In some sense - Collection is storage construct that allows you to collect things

Python offers several built-in types that fall under a vague category called collections. We will talk about following:

Strings	**Lists**
Dictionaries	**Sets**

Strings

- Strings are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:])
    - with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

OPERATIONS

- Concatenation +
- Repetition with the help of asterisk *

Strings are immutable

Strings

A string is a sequence of characters. A character is simply a symbol. For example, the English language has 26 characters. The same symbols - computer stores in numbers internally (0's and 1's) (encoding/decoding process - ASCII, Unicode)

```
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)
```

strings_one.py

```
Hello
Hello
Hello
Hello, welcome to
           the world of Python
```

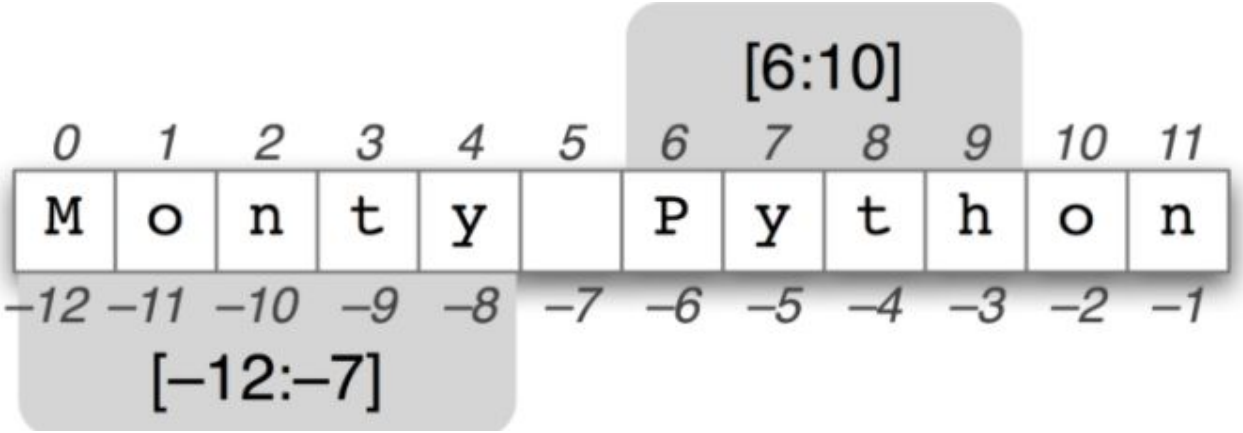
Strings (Cont ...)

```
#!/usr/bin/python3

str = 'Hello World!'

print (str)          # Prints complete string
print (str[0])       # Prints first character of the string
print (str[2:5])     # Prints characters starting from 3rd to 5th
print (str[2:])      # Prints string starting from 3rd character
print (str * 2)      # Prints string two times
print (str + "TEST") # Prints concatenated string
```

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

strings_demo.py

Strings (Cont...)




```
#!/usr/bin/python3

s = 'Hello World'
print (len(s))
print (type(s))

s = 'Monty Python'

print(s[6:10])
print(s[-12:-7])
```

```
11
<class 'str'>
Pyth
Monty
```

strings_len.py

Strings - Important Methods

- s.lower(), s.upper() -- returns the lowercase or uppercase version of the string
- s.strip() -- returns a string with whitespace removed from the start and end
- s.isalpha()/s.isdigit()/s.isspace()... -- tests if all the string chars are in the various character classes
- s.startswith('other'), s.endswith('other') -- tests if the string starts or ends with the given other string
- s.find('other') -- searches for the given other string (not a regular expression) within s, and returns the first index where it begins or -1 if not found
- s.replace('old', 'new') -- returns a string where all occurrences of 'old' have been replaced by 'new'
- s.split('delim') -- returns a list of substrings separated by the given delimiter. T
- s.join(list) -- opposite of split()

Quiz

What is the output of following code?

1
```
>>> "a" + "bc"
```

2
```
>>> "abcd"[2]
```

3
```
>>> print(r"\nhello")
```

4
```
>>> str1 = 'helloworld'
>>> str1[::-1]
```

5
```
>>> jio = '4G Liyo To Jio!'
>>> jio[3:7]
```

6
```
>>> jio = '4G Liyo To Jio!'
>>> jio[7:]
```

Quiz - Answers

What is the output of following code?

1
```
>>> "a" + "bc"
'abc'
```

2
```
>>> "abcd"[2]
'c'
```

3
```
>>> print(r"\nhello")
\nhello
```

4
```
>>> str1 = 'helloworld'
>>> str1[::-1]
'dlrowolleh'
```

5
```
>>> jio = '4G Liyo To Jio!'
>>> jio[3:7]
'Liyo'
```

6
```
>>> jio = '4G Liyo To Jio!'
>>> jio[7:]
' To Jio!'
```

String Formatting - Text Displaying

```
# implicit order (default one)
default_order = "{}, {} and {}".format('Hari','Sadu','Naukari')
print('\n--- Default Order ---')
print(default_order)

# order using positional argument
positional_order = "{1}, {0} and {2}".format('Hari','Sadu','Naukari')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {n} and {h}".format(h='Hari',s='Sadu',n='Naukari')
print('\n--- Keyword Order ---')
print(keyword_order)
```

```
--- Default Order ---
Hari, Sadu and Naukari

--- Positional Order ---
Sadu, Hari and Naukari

--- Keyword Order ---
Sadu, Naukari and Hari
```

string_format.py

String Format (Cont ...)

```
# formatting integers
#'Binary representation of 12 is 1100'
print("Binary representation of {0} is {0:b}".format(12))

 # formatting floats
#'Exponent representation: 1.566345e+03'
print("Exponent representation: {0:e}".format(1566.345))

# round off
#'One third is: 0.333'
print( "One third is: {0:.3f}".format(1/3))

# string alignment
#'|butter    |  bread   |       ham|'
print( "|{:<10}|{:^10}|{:>10}|".format('butter','bread','ham'))
```

```
Binary representation of 12 is 1100
Exponent representation: 1.566345e+03
One third is: 0.000
|butter    |  bread   |       ham|
```

Python Lists

A list contains items separated by commas and enclosed within square brackets ([])

```
#!/usr/bin/python3

list      = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']


print (list )               # Prints complete list
print (list[0]   )          # Prints first element of the list
print (list[1:3] )          # Prints elements starting from 2nd till 3rd
print (list[2:] )           # Prints elements starting from 3rd element
print (tinylist * 2 )       # Prints list two times
print (list + tinylist )    # Prints concatenated lists

print (type(list))          # type
print(len(list))            # length
```

List is sequence

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
<class 'list'>
5
```

lists_demo.py

for - Loop

```
#!/usr/bin/python3

languages = ["C", "Java", "Perl", "Python", "Node.js", "JavaScript"]

for x in languages:
     print (x)
```

```
C
Java
Perl
Python
Node.js
JavaScript
```

A few important key Words:

- break
- continue

for_demo.py

for - loop (Cont...)

```
#!/usr/bin/python3

edibles = ["ham", "spam","eggs","nuts"]
for food in edibles:
    if food == "spam":
        print("No more spam please!")
        break
    print("Great, delicious " + food)
else:
    print("I am so glad: No spam!")
print("Finally, I finished stuffing myself")
```

```
Great, delicious ham
No more spam please!
Finally, I finished stuffing myself
```

for_demo_adv.py

for_demo_adv_nospam.py

while - loop

```
#!/usr/bin/python3

count = 0
while (count < 9):
   print ('The count is:', count)
   count = count + 1

print ("Good bye!")
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

while_demo.py

Quiz

What is the output of the following code?

```
#!/usr/bin/python3
presidents = ['Clinton', 'Barack', 'Trump']
for president in presidents:
   if president == 'Trump':
        print ("No more Trump - please")
        break
   print ("Great - President: ", president)
else:
   print ("I am so glad - even Trump was covered")

print ("I am done")
```

Option - A

```
Great - President:  Clinton
Great - President:  Barack
No more Trump - please
I am done
```

Option - B

```
Great - President:  Clinton
Great - President:  Barack
Great - President:  Trump
I am so glad - even Trump was covered
I am done
```

Quiz

Study the following Script. The script stops, makes exit as soon as User enters Char "Q". Is that True?

```
#!/usr/bin/python3

ch = True
while (ch):
    key = input("Enter Key: ")
    print ("You entered: ", key)
    if key == 'Q':
        ch = False
        print ("Oh - you entered Q, you are making exit")
```

Built-in Method - “range”

```
>>> for i in range(5):
...     print (i)
...
0
1
2
3
4
```

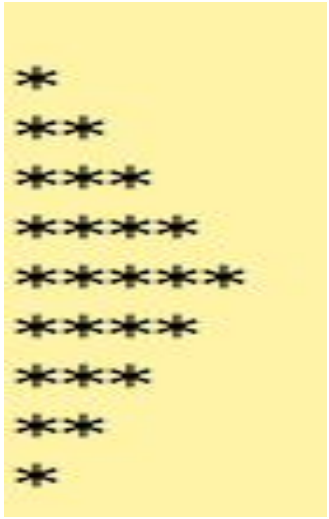
```
>>> for i in range(2, 5):
...     print (i)
...
2
3
4
```

```
>>> for i  in range(3, 10, 2):
...     print (i)
...
3
5
7
9
>>>
```

```
>>> L  = list(range(10))
>>> print (L)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(10)
range(0, 10)
>>> type(range(10))
<class 'range'>
```

Task - 7

Write a script to print pattern like below.



Thank you