NUS AI SUMMER EXPERIENCE 2019

BASIC VARIABLES IN PYTHON

PAN BINBIN

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

WHY PYTHON

- EASY TO LEARN
- EVERYONE IS USING IT
- EXTENSIVE HELP AND LIBRARIES

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

OUTLINE

01 VARIABLES AND DATATYPES

02 STRINGS

03 BOOLEANS

04 OPERATORS

01 VARIABLES & DATATYPES

VARIABLE

- Add Stuff
- Manipulate: Mix/Stir/Heat
- Get what you want

Variable is a container

VARIABLE

- Program executes exactly what you tell it to do
- Strict rules (syntax)

```
print('Hello World')
```

```
Hello World
```

- But you don't want to have to type the names of everyone you want to say hello to
- Use a variable: named memory location (container) that holds a value

- name is a variable
- Assigning value 'John' to name.
- Takes any value assigned to it

```
name = 'John'
print ('Hello ',name)
```

```
Hello  John
```

The program prints out whatever is contained in the variable

ASSIGNING VALUES TO VARIABLES

- Assign value inside program (as in previous slide)
- Assign value by getting inputs from console (interaction with user)

```
name = input('What is your name?  ')
print ('Hello ',name)
```

```
What is your name?  Thomas
 Hello  Thomas
```

- Assign value by reading from database (most common)

```
name =open('Names.txt').read().split("\n")
for i in name:
    print('Hello ', i)
```

```
Hello  Amy
Hello  Beth
Hello  Cate
```

The file containing the names

DATATYPES

- Every value has a datatype
- Attributes
- Three types of data
  - Numeric
    - Integers
      - No. of people in room, Team score in game
    - Floats
      - CAP, Price of something
  - Strings
    - Name, Address
  - Booleans
    - True (T) or False(F)

DATATYPES DETERMINE HOW OPERATORS WORK

```
stringA = '3'
stringB = '4'
numberA = 3
numberB = 4
print('Data type stringA: ', type(stringA),'\n', 'Data type numberA: ', type(numberA))
```

```
Data type stringA:  <class 'str'>
 Data type numberA:  <class 'int'>
```

```
numberA+numberB
```

```
7
```

```
stringA+stringB
```

```
'34'
```

Same operator "+"
Different results for numbers and strings

DATATYPES DETERMINE APPLICABILITY OF OPERATORS

```
stringA/stringB
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-d270b6e01b10> in <module>()
----> 1 stringA/stringB

TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

Some operator/method not applicable for certain datatypes

```
stringA+numberA
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-682b894ec95b> in <module>()
----> 1 stringA+numberA

TypeError: Can't convert 'int' object to str implicitly
```

Cannot mix datatypes for some operators

EXAMPLES OF ASSIGNMENTS OF VALUES WITH DIFFERENT DATATYPES

- age = 51 (integer: a number without any decimal)
- CAP = 3.5 (float: a number with decimal points)
- statement = 'I am a No. 1' (string: enclosed with either single quotes ' ' or double quotes " ")
- alive = True (Boolean: takes values of True or False. No quotes)

NAMING CONVENTION FOR VARIABLES

- Use as many words a necessary for understanding and recall
- No space between words
  - number of students
- Use camel case or underscore
  - numberOfStudents
  - number_of_students
- Variable names are case-sensitive
  - number_of_students is not same as Number_of_students
- Not start with a digit
  - 2_name not allowed as variable name.

```
2_name = 'good'
  File "<ipython-input-7-150e05115436>", line 1
    2_name = 'good'
     ^
SyntaxError: invalid token
```

FORBIDDEN WORDS FOR VARIABLE NAMES

Special keywords used by Python

and	continue	except	is	print	yield
as	def	exec	lambda	raise	True
assert	del	if	not	return	False
break	elif	import	or	try	None
class	else	in	pass	while	

02 STRINGS

BASIC FEATURES OF STRINGS

- String is a sequence of characters.
- Includes characters like ‘abc123’, blank spaces, and other symbols ‘\n’ (for newline)
- Delimited by single quotes (‘..’), double quotes (“...”)

```
string = ' 123456789 '
len(string)
```

```
11
```

- count each item as separate
- can be accessed separately unlike a number

WHAT CAN TRIP YOU UP ON A STRING?

- Special characters
  - Use of special character must be preceded by escape character \
  - Example,
    - you want to print ‘I love apples’ (with quotation marks)
    - not I love apples (without quotation marks)

```
stringApp = ''I love apple''
print(stringApp)
```

```
  File "<ipython-input-50-43916704d95f>", line 1
    stringApp = ''I love apple''
                 ^
SyntaxError: invalid syntax
```

Typing the ‘ ‘ marks directly does not work

```
stringApp = '\'I love apple\''
print(stringApp)
```

```
'I love apple'
```

The \' treats the ‘ marks after the escape character \ as a normal quotation mark, not a special character

Alternative code: stringApp = "'I love apple'" (enclose the string with “ ...”. The single ‘ ..’ within the double “ “ will be treated as part of string not special characters. This is NOT a special feature of “ ..”. It works because the “ .. “ used to define the string is different from the ‘..’)

WHAT CAN TRIP YOU UP ON A STRING?

- Printing on new line (\n)
- \ followed by some alphabets has special meaning within the string

```
stringAppNewLine = 'I \nlove \napple'
print(stringAppNewLine)
```

```
I
love
apple
```

The \n gets the code to go a newline after “I” and after “love”

```
stringAppNewLine = 'I \\nlove \napple'
print(stringAppNewLine)
```

```
I \nlove
apple
```

See the difference in this code?

Reference for escape characters
http://python-reference.readthedocs.io/en/latest/docs/str/escapes.html

Manipulating Strings

s = 'Singapore is a very hot place with temperature going up to 36 degrees Celcius'

Index 0 9 76

Each element in the string has an index or position. Starts from 0.

```
s = 'Singapore is a very hot place with temperature going up to 36 degrees Celcius'
s[0:9]
```

```
'Singapore'
```

Accessing substring by specifying its positon, s[i:j]

- extract elements from index i to index (j-1)

s[i:] – extract elements starting from index i to end of string

s[:j] – extract elements starting from index 0 to index (j-1)

Negative index: count backwards from the last element towards the first element

s[-1] -last element on the list

s[-3:-1] – start from the third element from the back and stop at the second element from back

FINDING & REPLACING ELEMENTS

s = 'Singapore is a very hot place with temperature going up to 36 degrees Celcius'

```
s.find('hot')
```

```
20
```

The find method outputs the index of the occurrence of the first letter of the word searched for

```
s.replace('Singapore', 'Pulau Ubin')
```

```
'Pulau Ubin is a very hot place with temperature going up to 36 degrees Celcius'
```

s.replace(phrase1,phrase2) – replace phrase 1 by phrase 2 in the string s

STRING SPLITTING

s = 'Singapore is a very hot place with temperature going up to 36 degrees Celcius'

```
s1 = s.split()
s1
```

```
['Singapore',
 'is',
 'a',
 'very',
 'hot',
 'place',
 'with',
 'temperature',
 'going',
 'up',
 'to',
 '36',
 'degrees',
 'Celcius']
```

How do we want to split up the string s.
By space in between the words

s.split() creates a list (we talk more about list later), with each word an element

```
print(s[0],'\n',s1[0])
```

```
S
 Singapore
```

Before splitting, each letter is one element in a string, s
After splitting, each word is one element in a list s1

STRING JOINING


```
s11 = 'Singapore'; s12 = 'is'; s13 = 'hot'
sjoin = ' '.join([s11,s12,s13])
sjoin
```

```
'Singapore is hot'
```

join 3 strings, s11, s12, s13 into one string sjoin. The 3 strings are separated by a space.

'delimiter'.join() – in this case, the delimiter (or separator) between the strings to be joined is a space. You can put anything as delimiter

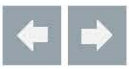
```
s11 = 'Singapore'; s12 = 'is'; s13 = 'hot'
sjoin = '&'.join([s11,s12,s13])
sjoin
```

```
'Singapore&is&hot'
```

' '.join() takes only 1 argument. We have three strings, s11, s12, s13, that we want to join so we put them into ONE list [ ].

In this case, we use '&' as the delimiter so the final string sjoin has words separated by &

03 BOOLEAN



BINARY NATURE OF BOOLEAN


```
x=0; y=1.2
result = x>0 and y <1
print(result, '\n', type(result))
```

```
False
 <class 'bool'>
```

```
result = x==0 and y >1
print(result, '\n', type(result))
```

```
True
 <class 'bool'>
```

Outcome of an evaluation with two possible results – True of False

Boolean is a data type

BOOLEAN EVALUATION OF OBJECTS


```
bool(0)
```

```
False
```

```
bool(1.3)
```

```
True
```

```
sfill = 'some string'
bool(sfill)
```

```
True
```

```
sempty = ''
bool(sempty)
```

```
False
```

0 (False), and numbers different from 0 (True)
Empty objects (False), nonempty objects (True)

04 OPERATORS

MATHEMATICAL OPERATORS

- + Add
- - Subtract
- / Divide
- // Integer Divide
- * Multiply
- ** Raise to the power of
- % Modulo (also known as remainder)
- () Grouping

PRIORITY OF MATHEMATICAL OPERATORS

- Brackets ( )
- Orders (power) **
- Division /
- Multiplication *
- Addition +
- Subtraction -
- x = 5 + 4 * 9/6
- x =(5 + 4) * (9/6)

Bitwise operators

- Python Bitwise Operators take one to two operands, and operates on it/them ***bit by bit***, instead of whole.
- Bitwise AND (&),
- Bitwise OR (|),
- Bitwise XOR (^),
- Bitwise 1's Complement (~),
- Left-Shift (<<),
- Right-Shift (>>).

Bitwise AND (&)

0 & 0	0
0 & 1	0
1 & 0	0
1 & 1	1

```
bin(5)
'0b101'
bin(7)
'0b111'
5 & 7
5
```

Number	Binary
5	101
7	111
5 & 7	101
5|7	111

Bitwise OR (|)

0|0	0
0|1	1
1|0	1
1|1	1

```
5|7
7
```

Bitwise XOR (^)

0^0	0
0^1	1
1^0	1
1^1	0

```
6^6
```
```
0
```
```
6^0
```
```
6
```
```
6^3
```
```
5
```

Number	Binary
0	0
3	011
6	110
6^3	101
45	101101
-46	-101110

Bitwise 1's Complement (~)

This operator takes a number's binary, and returns its one's complement. For this, it flips the bits until it reaches the first 0 from right. ~x is the same as -x-1.

```
~45
```
```
-46
```
```
print(bin(45))
print(bin(-46))
```
```
0b101101
-0b101110
```


Left-Shift (<<),


```
2<<1
```
```
4
```
```
3<<2
```
```
12
```
```
print(bin(2))
print(bin(4))
print(bin(3))
print(bin(12))
```
```
0b10
0b100
0b11
0b1100
```

- The left-shift operator shifts the bits of the number by the specified number of places. This means it adds 0s to the empty least-significant places now. Let's begin with an unusual example.

Right-Shift (>>).

```
3>>1
```
```
1
```
```
31>>3
```
```
3
```
```
print(bin(3))
print(bin(1))
print(bin(31))
```
```
0b11
0b1
0b11111
```

- On the other hand, Right-Shift shifts the bits to the right by the specified number of places.

SIMPLE CALCULATOR

Area and Circumference of a Circle

1. Read in the radius of the circle
2. Store radius in a variable
3. Compute circumference = $2\pi r$
4. Compute area = πr^2
5. Print out circumference and area

Code

```
radius = input('What is the radius of circle (in meters)? ')
circumference = 2*(22/7)*radius
area = (22/7)*(radius)**2
print('Circumference: ', circumference, 'Area:', area)
```

Code does not work.
What is the error?
How to fix the error?

PAN BINBIN

BINBIN.PAN@NUS.EDU.SG

NUS National University of Singapore

NUS AI SUMMER EXPERIENCE 2019

BASIC VARIABLES IN PYTHON