Python
WEEK 1
Introduction & Review

AI Academy

WELCOME TO AI ACADEMY !

Program Courses

1. Computer Programming with Python
2. Data Mining
3. Introduction to Artificial Intelligence
4. Machine Learning

COMPUTER PROGRAMMING WITH PYTHON

Instructor - James E. Robinson, III

Teaching Assistant - Travis Martin

ABOUT TRAVIS

Travis has several degrees including a master's degree in computer science education, a bachelors in physics education, and a bachelors in computer engineering. Additionally, he taught high school science for 15 years, mostly physics.

Travis currently lives in Texas but has lived in multiple places in the US over the course of his life.

ABOUT JAMES

James has a Masters of Science in Computer Engineering from NC State University. The latter with a focus on computer networking and software design.

While currently involved primarily in systems architecture, James still writes integration code for projects when required (Python, SQL, cloud service APIs, Amazon Web Services).

ABOUT JAMES

- Pascal
- C
- C++
- TCSH
- BASH
- Awk
- Perl
- Java
- TCL/TK
- Ruby
- Go
- JavaScript
- Python

DEVELOPMENT ENVIRONMENT

This environment will be used by subsequent AIA courses.

Recommended -

- Windows 10
- Anaconda v. 2022.05 or later
  - Interpreter: Python 3.9 - included
  - IDE: Jupyter Notebook and Spyder - included
  - Modules: numpy, pandas, matplotlib, etc. - included

LIGHTNING REVIEW

- Variables
- Input / Output
- Expressions
- Functions
- Conditional Control
- Looping
- Data Types
  - Lists
  - Tuples
  - Dictionary
  - Sets

VARIABLES

VARIABLES ARE LABELS TO VALUES

Good Examples

full_name = "Laila Ali"
StreetName = "Butterfly Street"

Note:
Good variable names define the purpose of the value they posses.

Variable names are case sensitive.

Bad Examples

1st_name = "Laila"
XÆA12 = "son"
First name = "Laila"

Note:
Variable names cannot start with numbers.

Yes, Python does understand unicode characters. No, do not use them for variable/object names.

INPUTS & OUTPUTS

Print Statement

print("This is a print statement")

OUTPUT : This is a print statement

Input Statement

input_string = input("Enter a short string here: ")

OUTPUT : Enter a short string here: *hello*

Printing Value

print("Your input string is: ", input_string)

OUTPUT : Your input string is: hello

Formatting Output

float_number = 2.676

print("Float number with 2 decimal places is ", format(float_number, '.2f'))

OUTPUT : Float number with 2 decimal places is 2.68

FORMATTED STRING LITERALS

FOR STRINGS THAT MAY INCLUDE VARIABLES OR EXPRESSIONS

Syntax

f"any text plus {expression}"

Example

```
num1 = 5
num2 = 10
print(f" {num1} times {num2} gives the result: {num1*num2}")
```

OUTPUT : 5 times 10 gives the result: 50

EXPRESSIONS

EXPRESSIONS ARE COMBINATIONS OF VALUES, VARIABLES AND OPERATORS

Assignment

```
y = 3
x = 2
x = y = 5
```

Operations

```
z = x % y
# operator can be  + , - , * , / , // , %
```

Note:
% is modulo operator
// is absolute division

Shorthand Operations

```
z += 1
# essentially z = z+1
# operator can be  + , - , * , /
```

Note:
z needs to have a preassigned value

Complex Operations

```
z = (x+y) * (x-y) / (2*x)
```

Note:
PEMDAS is followed

FUNCTIONS

A BLOCK OF CODE THAT RUNS WHEN CALLED

Simple Function

```
def my_summer(param1, param2 = 0):
    '''
    Sum of two values.
    Inputs:
        param1 - int or float
        param2 - int or float, optional
    Returns:
        sum - int or float
    '''
    return param1 + param2
```

Function Call

```
my_sum = my_summer(10,20)
```

Note :

A function may accept several or no parameters

A function may or may not return a value

A function is also considered as an expression if it returns a value

CONDITIONAL CONTROL

EXECUTE CODE BASED ON THE TRUTH VALUE OF A CONDITION

IF ELSE Structure

```
if temperature > 100.3: # per CDC guidelines
    print("Patient has a fever")
    if temperature > 102.9: # nested IF condition
        print("Patient should seek medical attention")
elif temperature > 99.5:
    print("Patient has low-grade fever")
else:
    print("Patient is either normal or has assumed room temp.")
```

Note :

If one condition in a IF/ELSE structure is true, other conditions after that are not checked by code.

CONDITIONAL CONTROL

EXECUTE CODE BASED ON THE TRUTH VALUE OF A CONDITION

Boolean Logic

```
if temperature > 100.3 and cough == True: # per CDC guidelines
    print("Patient may have COVID")
elif temperature > 99.5 or cough == False:
    print("Patient has low-grade fever but not likely COVID")
else:
    print("Patient is either normal or has assumed room temp.")
```

Note :
Boolean logic consists of AND, OR, NOT.

LOOPING

EXECUTE SAME CODE BLOCK NUMBER OF TIMES

While Loop Example

```
x = 1
y = 22
while x < y:
    x += 3
    print(f"while: x = {x}")
```

For Loop Example

```
y = 22
for x in range(1, 100, 3):
    if x < y:
        print(f"for: x = {x}")
    else:
        break
```

Nested Loop Example

```
y = 22
for x in range(1, 100, 3):
    z = x
    while z > 10:
        print(f"while: z= {z}")
        z -= 3
    if x < y:
        print(f"for: x = {x}")
    else:
        break
```

DATA STRUCTURES - LISTS

Example

a = ['foo', 'bar', 2, 2.5, True]

Note:

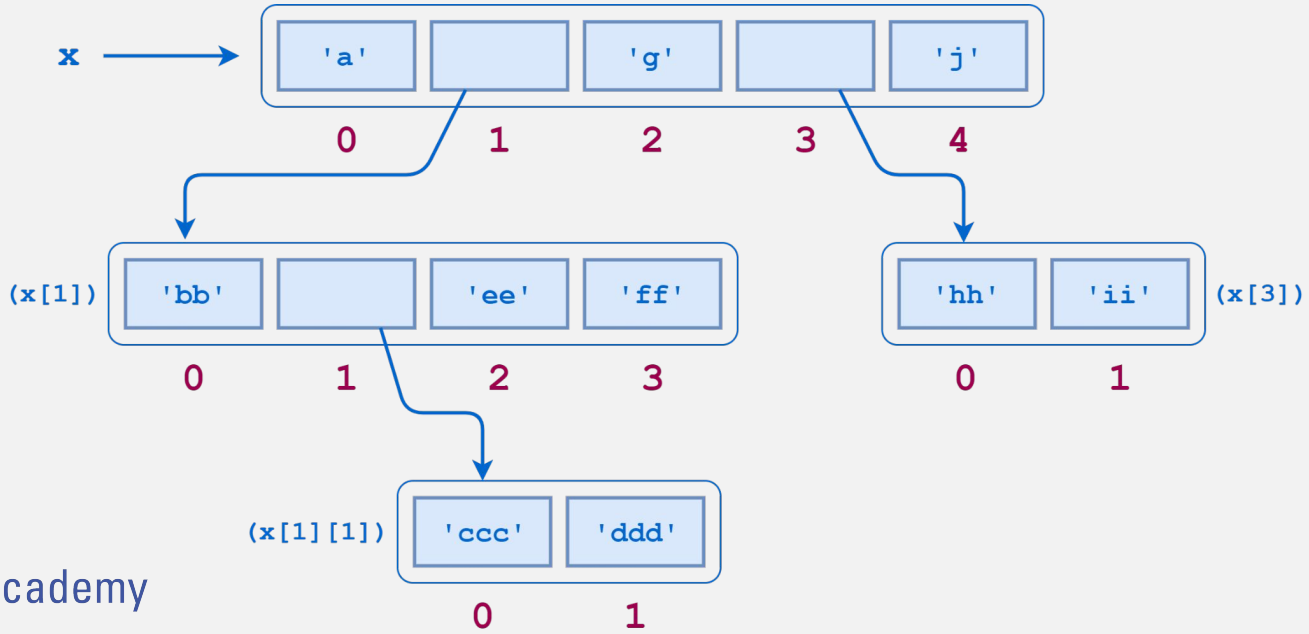
- Lists are ordered.
- Lists can contain any arbitrary objects.
- List elements can be accessed by index.
- Lists can be nested to arbitrary depth.
- Lists are mutable.
- Lists are dynamic.

DATA STRUCTURES - LISTS

NESTED LISTS

Example

x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']



DATA STRUCTURES - TUPLES

A TUPLE IS AN IMMUTABLE LIST

Example

t = ('foo', 'bar', 'baz', 'qux', 'quux')

Note:

- Tuples are ordered.
- Tuples can contain any arbitrary objects.
- Tuples elements can be accessed by index.
- Tuples can be nested to arbitrary depth.
- Tuples are immutable

Immutability

t[1] = 'boo'

>>> TypeError: 'tuple' object does not support item assignment

DATA STRUCTURES - DICTIONARY

DICTIONARIES ARE KEY VALUE PAIRS

Example

```
this_dict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
      }

print(this_dict["brand"])
```

OUTPUT : Ford

Note:

- Dictionary keys are unique
- Dictionaries are dynamic and mutable
- Dictionaries can be nested
- Dictionaries can be indexed, although it defeats the purpose.

DATA STRUCTURES - SETS

SETS ARE UNORDERED LISTS WITH UNIQUE VALUES

Example

test_set = set(['Jodi', 'Carmen', 'Aida"])

Note :

- Set values are unique - If a list with duplicates it converted to set, duplicates are removed
- Sets are mutable
- Sets are unordered and sorted
- Sets can be indexed

DEBUGGING

SYNTAX ERRORS AND SEMANTIC ERRORS

Syntax Errors

```
In [2]: x = input("Enter Values"
          File "C:\Users\NitishTalekar\AppData\Local\Temp\ipykernel_9072\1861133380.py", line 1
            x = input("Enter Values"
                                    ^
        SyntaxError: unexpected EOF while parsing
```

```
In [8]: a = 1
        x 1
          File "C:\Users\NitishTalekar\AppData\Local\Temp\ipykernel_9072\1794020002.py", line 2
            x 1
              ^
        SyntaxError: invalid syntax
```

Semantic Errors

THESE ERRORS DON'T HAVE A DISPLAY MESSAGE, BUT THE CODE DOES NOT RUN AS EXPECTED.

DEBUGGING

RUNTIME ERRORS

NameError

```
In [9]: x = y

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9072\4248681928.py in <module>
----> 1 x = y

NameError: name 'y' is not defined
```

TypeError

```
In [10]: x = "Hi" + 5

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9072\3031677582.py in <module>
----> 1 x = "Hi" + 5

TypeError: can only concatenate str (not "int") to str
```

AI Academy

KeyError

```
In [12]: x = {"a":0}
         x["b"]

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9072\221223370.py in <module>
      1 x = {"a":0}
----> 2 x["b"]

KeyError: 'b'
```

AttributeError

```
In [14]: x = "Hello"
         x.append("Hi")

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9072\4217443345.py in <module>
      1 x = "Hello"
----> 2 x.append("Hi")

AttributeError: 'str' object has no attribute 'append'
```

IndexError

```
In [15]: x = [1,2,3]
         x[3]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9072\3922793346.py in <module>
      1 x = [1,2,3]
----> 2 x[3]

IndexError: list index out of range
```

THANK YOU

FOR ADDITIONAL QUERIES OR DOUBTS
CONTACT:
jerobins@ncsu.edu

AI Academy