ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Introduction to Information Systems and Applications

Course Unit 2: Data processing with python

M. Tzagarakis, V. Daskalou

School of Business Administration

Department of Economics

Unit Scope

- Make an introduction to the features of the python language
- Introduce the basic capabilities of programming languages for data processing

Unit contents

1. Characteristics of python
2. Programming with python
    1. Basics:

       Calculations and variables, Strings, Flow control, Composite types (lists, tuples, dictionaries), Functions, Files
    2. Manipulate CSV files
    3. Figures and plots
    4. Statistics

The python language

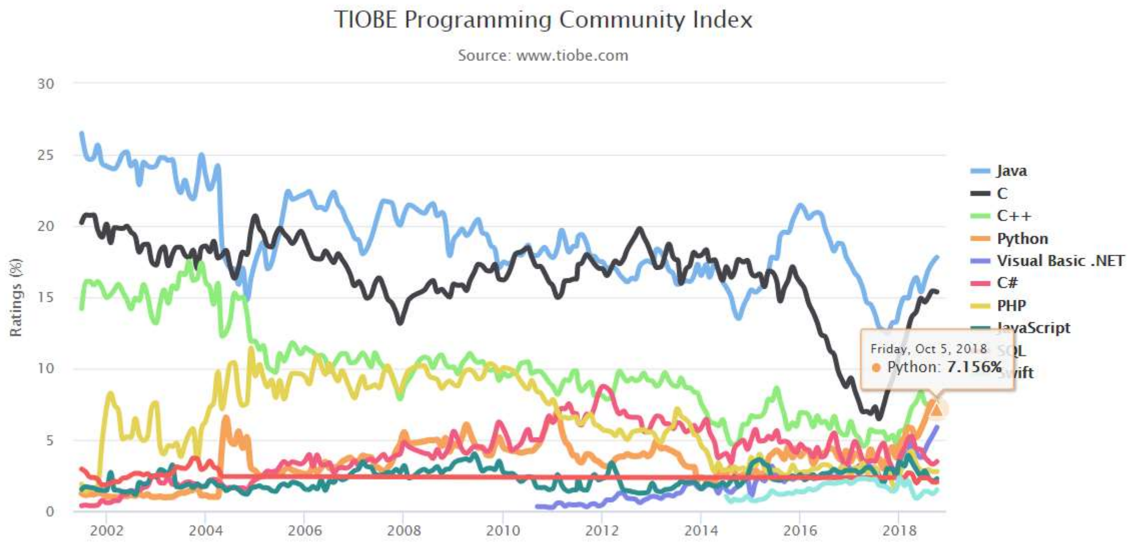
History of Python

- Created by Guido van Rossum in 1989 (name inspired by Monty Python)
- 2 versions:
  - Python 2.0 (October 2000) last versions 2.6 & 2.7
  - Python 3.0 (December 2008): not completely backward-compatible

Guido van Rossum

Πηγή : https://en.wikipedia.org/wiki/Guido_van_Rossum

python: Importance



Top-10 programming languages (Oct-2018)

By TIOBE Software B.V. [CC BY-SA 4.0 (http://creativecommons.org/licenses/by-sa/4.0)], via Wikimedia Commons
Source: https://www.tiobe.com/tiobe-index//

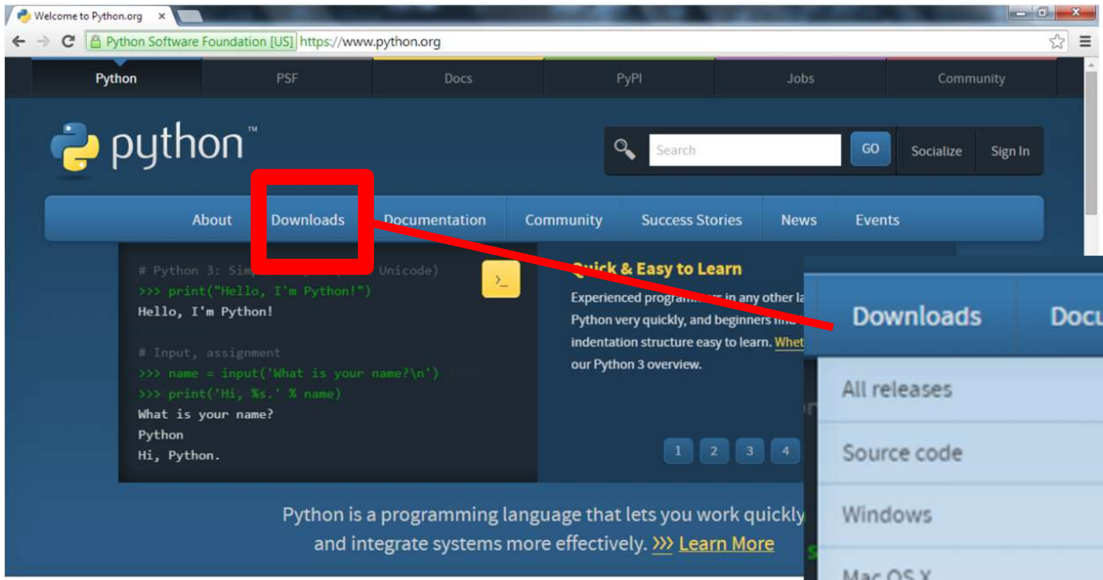
- #4 in top-10 programming languages
- Important for web, dbs & academic computing (source)
- the Most Popular Introductory Teaching Language at Top U.S. Universities (source)

Main characteristics

- Open source
- High level programming: Use natural language terms (English) ->easy to understand
- Compatible in different environments (OS)
- Philosophy: *"there should be one—and preferably only one—obvious way to do”*
  instead of "there is more than one way to do it" (Perl language)

Installation: base python (1)

1) Visit http://www.python.org/

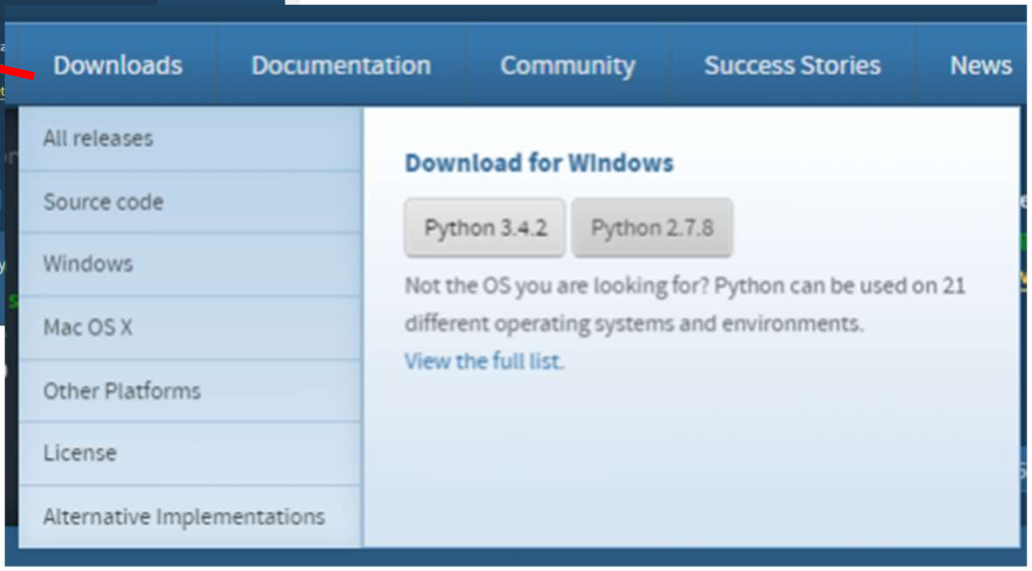


```
# Python 3: Sim                Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!

# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
```

Experienced programmers in any other la Python very quickly, and beginners fin indentation structure easy to learn. Whet our Python 3 overview.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

2) Download Python (current 3.x.x version)

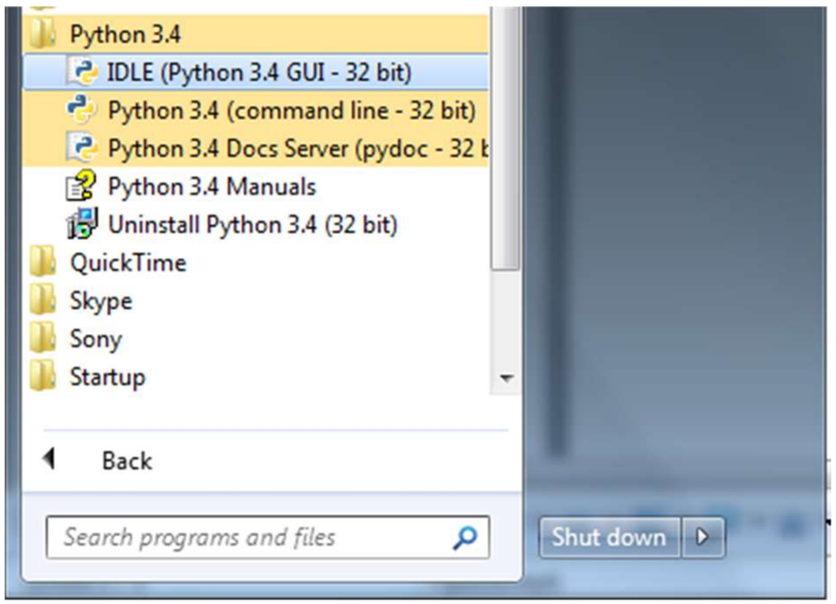


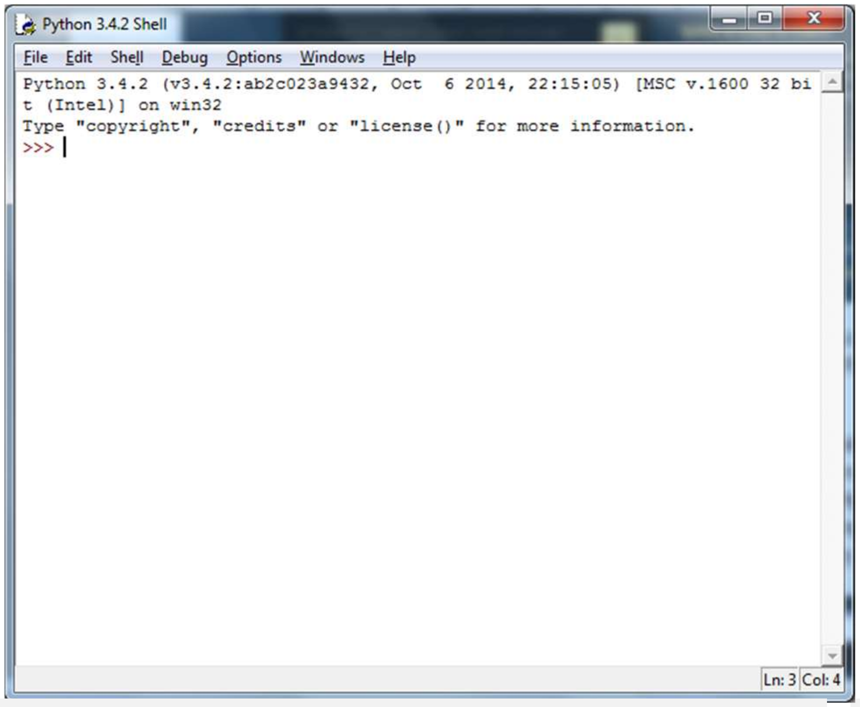
Download for Windows

Not the OS you are looking for? Python can be used on 21 different operating systems and environments. View the full list.

Installation: base python (2)

3) Download and install following the wizard. **IMPORTANT**: Click "Add python to path"
4) 2 ways to run a Python program:
   - Source file .py
   - Interactive interpreter prompt:
     - From Start Menu, open IDLE as follows: Start → All Programs → Python 3.x → IDLE (Python GUI)

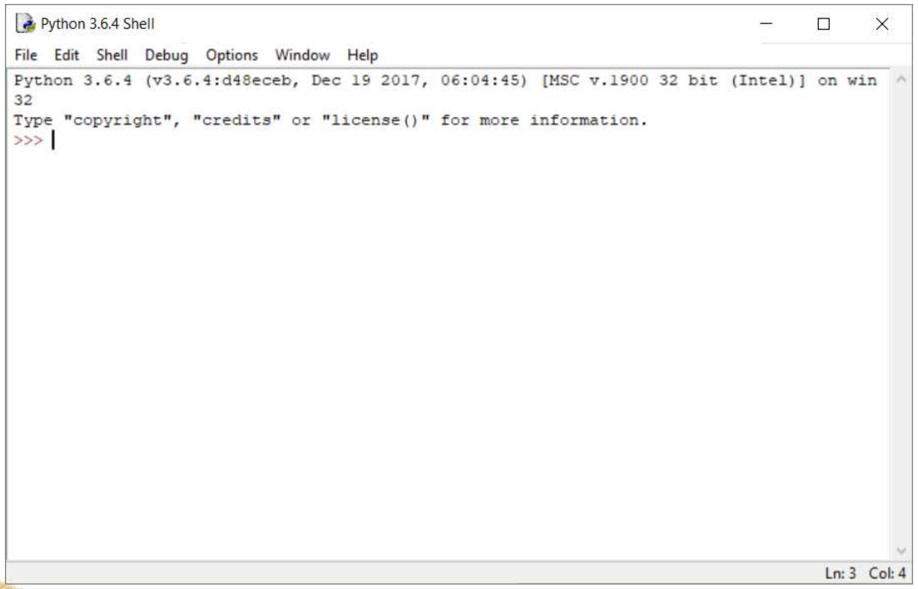

Python 3.4
IDLE (Python 3.4 GUI - 32 bit)
Python 3.4 (command line - 32 bit)
Python 3.4 Docs Server (pydoc - 32 b
Python 3.4 Manuals
Uninstall Python 3.4 (32 bit)
QuickTime
Skype
Sony
Startup
Back
Search programs and files
Shut down



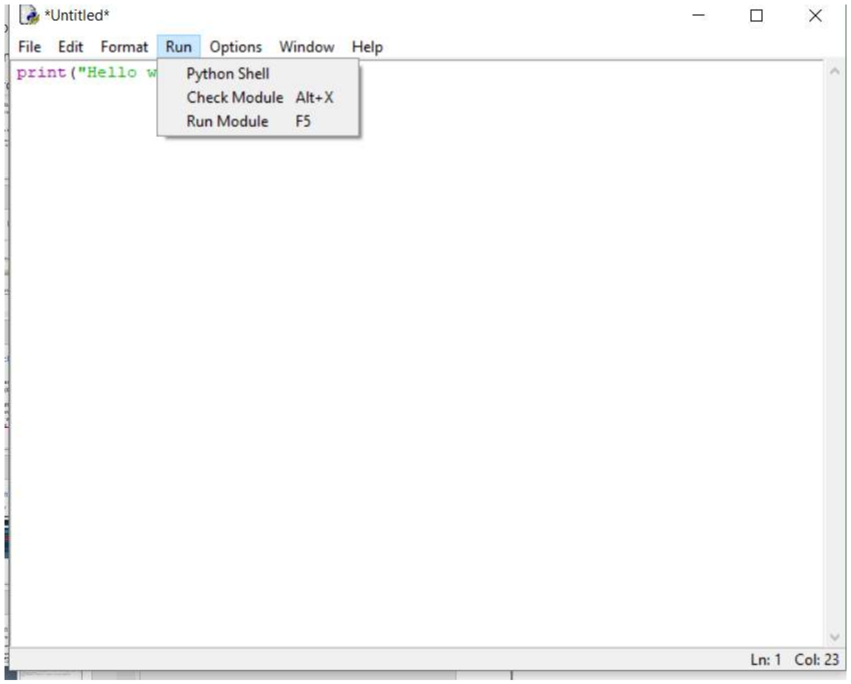
```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:15:05) [MSC v.1600 32 bi
t (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

Python IDLE

- "Integrated DeveLopment Environment" for Python.
- A software package that lets us write Python commands and edit and run Python programs.
- Python Shell:
- Helps writing python programs easily and create source files:
    - File->New->Save->Run->Run Module(F5)



```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 
```



```
print("Hello w
```

Anaconda python

- Python distribution from Continuum Analytics
- Advantages:
  - Pre-installed popular add-in packages
  - Integrated development environment: Spyder
- Download:
  - https://www.anaconda.com/download/
  - Anaconda 5.3 For Windows Installer->Python 3.7 version->Download
  - Double click the downloaded .exe and follow the instructions

Calculation and variables

Calculation and operators

```
>>>  (2+5)*5
35
>>>  2**2
4
>>>  14/4
3.5
>>>  14//4
3
>>>  14%4
2
>>>  2**1000
1071508607186267320948425049060001810561404811705533
   6074437503883703510511249361224931983788156958581 2
   7594672917553146825187145285692314043598457757469 8
   5748039345677748242309854210746050623711418779541 8
   2153046474983581941267398767559165543946077062914 5
   7119647768654216766042983165262438683720566806937 6
```

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 31
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -11
* Multiplication	Multiplies values on either side of the operator	a * b = 210
/ Division	Divides left hand operand by right hand operand	b / a = 2.1
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 1
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Variables and assignment

Variable:
- reserved memory locations to store values
- when you create a variable you reserve some space in memory
- a name to reference that space

Assignment:
- `vrbl=5`
  use = to store values in a variable, not equal
- `a=a+a`
  the use of a variable name on the right of the assignment operator = refers to the value stored in the variable

Variables

- Names:
  - letters, digits, or underscores _, always begin with a letter
- Reserved words
- Case sensitive
- Naming conventions:

https://en.wikipedia.org/wiki/Naming_convention_(programming)#Python_and_Ruby

lowercase_separated_by_underscores

Basic variable types

String (str): (in single or double quotes)

```
mystr='Hello Nikos',
grGM="Goodmorning",
yourPhone='2610459220'
```

Integer (int):

```
a=6, b=1234, c=-567
```

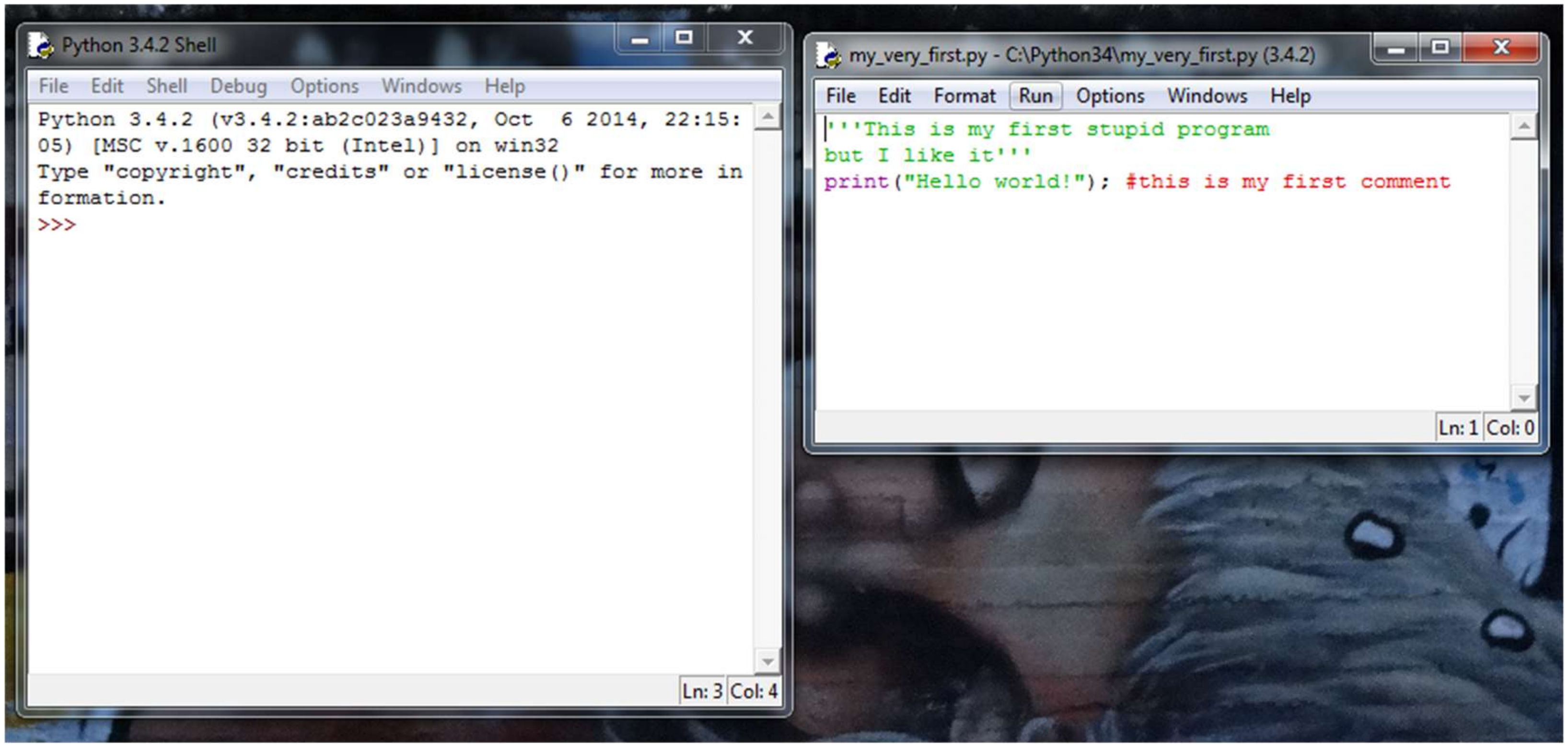
Floating point (float):

```
x=6.2, y=52.3E-4, z=-567.56789
```

Boolean: `True, False`

My first program in IDLE

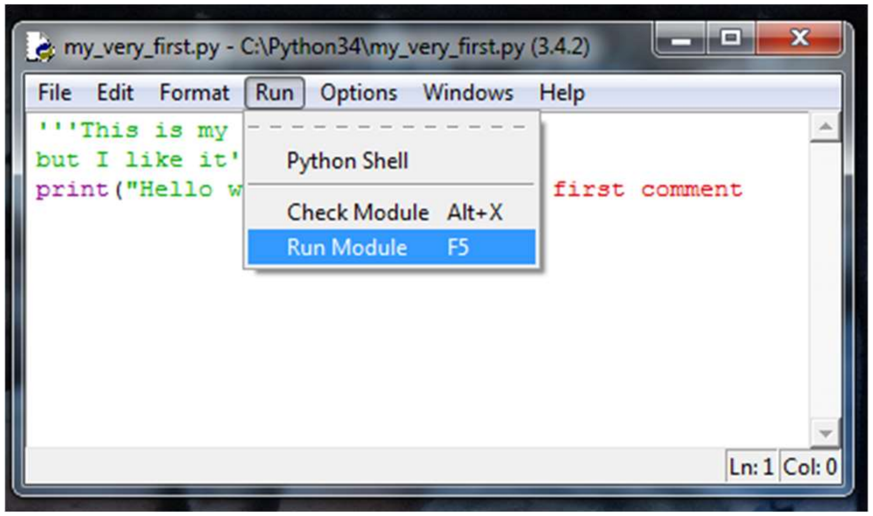
IDLE->File->New File


Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Ln: 3 Col: 4
my_very_first.py - C:\Python34\my_very_first.py (3.4.2)
File Edit Format Run Options Windows Help
Ln: 1 Col: 0

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:15:
05) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more in
formation.
>>>
```

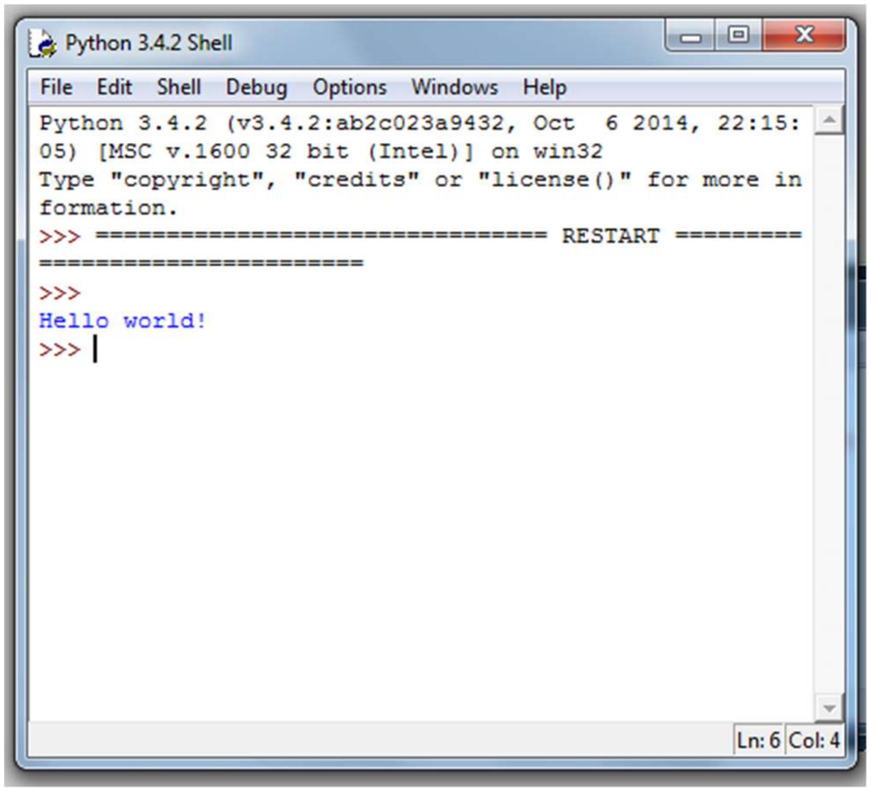
```
'''This is my first stupid program
but I like it'''
print("Hello world!"); #this is my first comment
```

My first program in IDLE: Execute



```
'''This is my
but I like it'
print("Hello w                    first comment
```

my_very_first.py - C:\Python34\my_very_first.py (3.4.2)
File Edit Format Run Options Windows Help
Python Shell
Check Module Alt+X
Run Module F5
Ln: 1 Col: 0

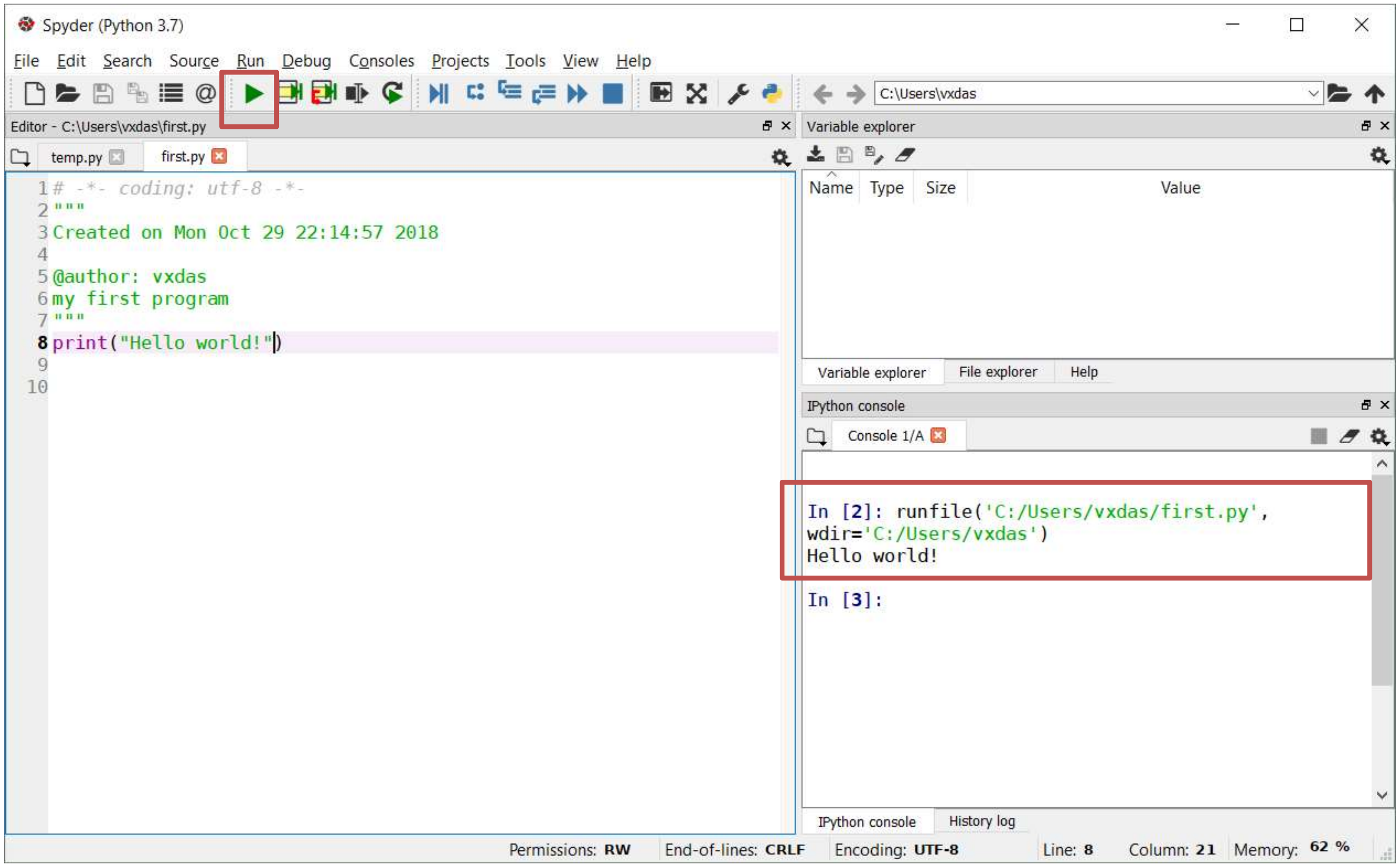


```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:15:
05) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more in
formation.
>>> ================================ RESTART =========
=======================
>>>
Hello world!
>>> 
```

My first program in IDLE

1. At IDLE Shell choose *File->New file*
2. The window of IDLE editor opens
3. We write our first program with python commands
4. We save our program with *File->Save*
5. We execute the program from *Run->Run Module*
6. The IDLE Shell window prints *RESTART* and starts the execution of our programs

My first program: Anaconda Spyder



```python
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 29 22:14:57 2018

@author: vxdas
my first program
"""
print("Hello world!")
```

```
In [2]: runfile('C:/Users/vxdas/first.py',
wdir='C:/Users/vxdas')
Hello world!

In [3]:
```

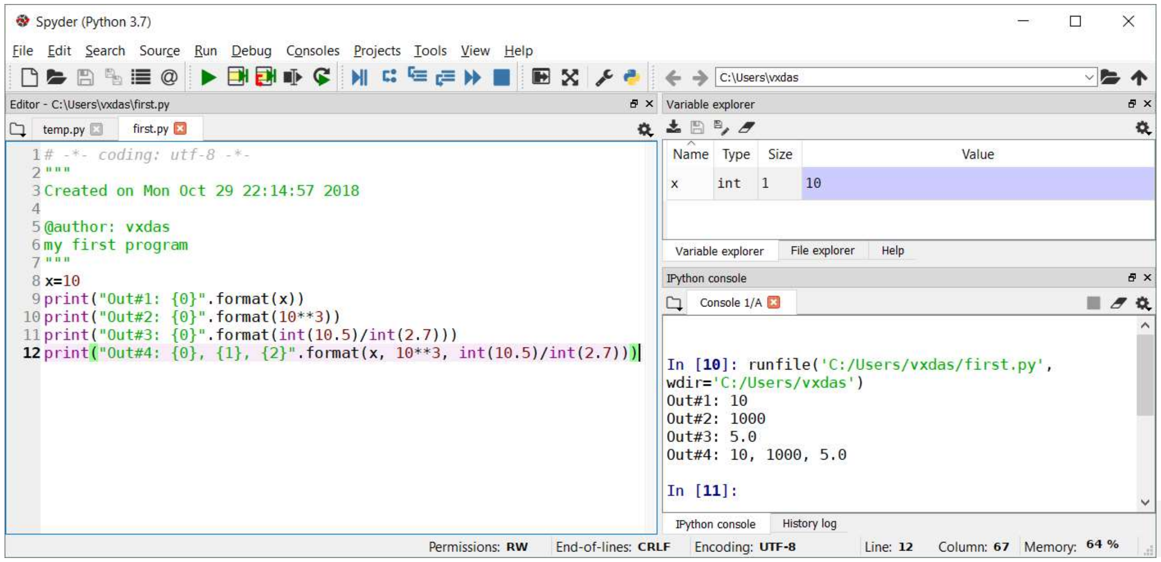
Numbers

Numbers (I)

Integers

Understand " ".format() function
https://www.geeksforgeeks.org/python-format-function

```
x=10

print("Out#1: {0}".format(x))

print("Out#2: {0}".format(10**3))

print("Out#3: {0}".format(int(10.5)/int(2.7)))

print("Out#4: {0}, {1}, {2}".format(x, 10**3, int(10.5)/int(2.7)))
```



```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 29 22:14:57 2018

@author: vxdas
my first program
"""
x=10
print("Out#1: {0}".format(x))
print("Out#2: {0}".format(10**3))
print("Out#3: {0}".format(int(10.5)/int(2.7)))
print("Out#4: {0}, {1}, {2}".format(x, 10**3, int(10.5)/int(2.7)))
```

Name	Type	Size	Value
x	int	1	10

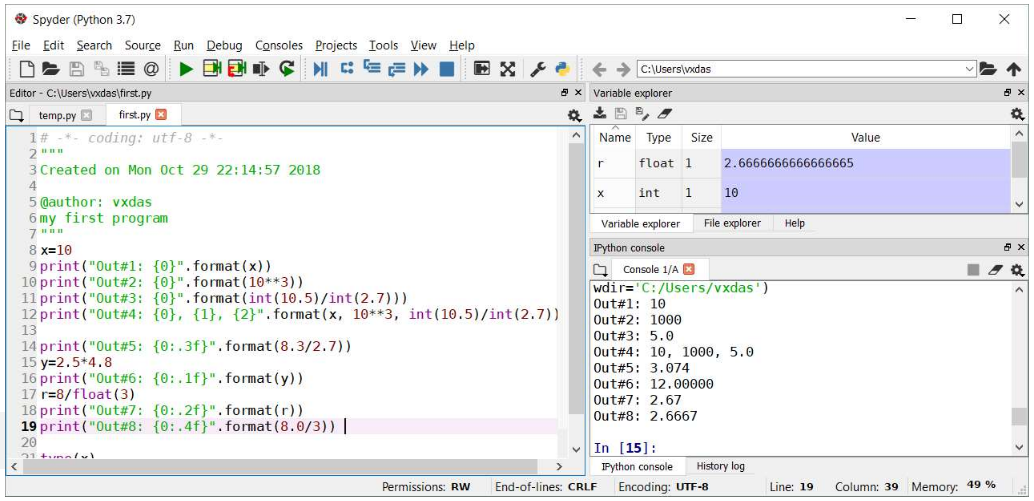
```
In [10]: runfile('C:/Users/vxdas/first.py',
wdir='C:/Users/vxdas')
Out#1: 10
Out#2: 1000
Out#3: 5.0
Out#4: 10, 1000, 5.0

In [11]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 12 Column: 67 Memory: 64 %

Numbers (II)

- Floating-point numbers

```
print("Out#5: {0:.3f}".format(8.3/2.7))

y=2.5*4.8

print("Out#6: {0:.1f}".format(y))

r=8/float(3)

print("Out#7: {0:.2f}".format(r))

print("Out#8: {0:.4f}".format(8.0/3))
```



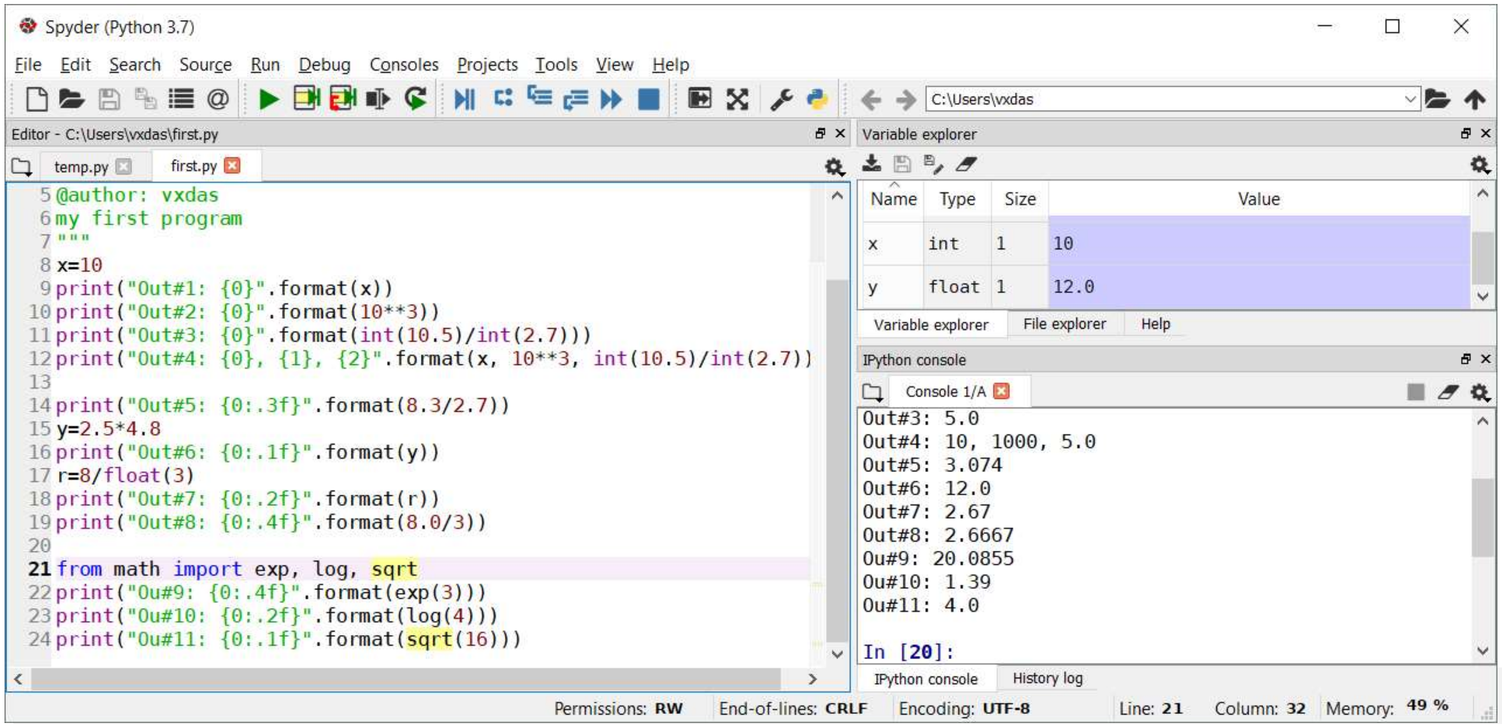
```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 29 22:14:57 2018

@author: vxdas
my first program
"""
x=10
print("Out#1: {0}".format(x))
print("Out#2: {0}".format(10**3))
print("Out#3: {0}".format(int(10.5)/int(2.7)))
print("Out#4: {0}, {1}, {2}".format(x, 10**3, int(10.5)/int(2.7))

print("Out#5: {0:.3f}".format(8.3/2.7))
y=2.5*4.8
print("Out#6: {0:.1f}".format(y))
r=8/float(3)
print("Out#7: {0:.2f}".format(r))
print("Out#8: {0:.4f}".format(8.0/3))

```

Name	Type	Size	Value
r	float	1	2.6666666666666665
x	int	1	10

```
wdir='C:/Users/vxdas')
Out#1: 10
Out#2: 1000
Out#3: 5.0
Out#4: 10, 1000, 5.0
Out#5: 3.074
Out#6: 12.00000
Out#7: 2.67
Out#8: 2.6667

In [15]:
```

`math` module

```
from math import exp, log, sqrt
print("Ou#9: {0:.4f}".format(exp(3)))
print("Ou#10: {0:.2f}".format(log(4)))
print("Ou#11: {0:.1f}".format(sqrt(16)))
```



```
@author: vxdas
my first program
"""
x=10
print("Out#1: {0}".format(x))
print("Out#2: {0}".format(10**3))
print("Out#3: {0}".format(int(10.5)/int(2.7)))
print("Out#4: {0}, {1}, {2}".format(x, 10**3, int(10.5)/int(2.7)))

print("Out#5: {0:.3f}".format(8.3/2.7))
y=2.5*4.8
print("Out#6: {0:.1f}".format(y))
r=8/float(3)
print("Out#7: {0:.2f}".format(r))
print("Out#8: {0:.4f}".format(8.0/3))

from math import exp, log, sqrt
print("Ou#9: {0:.4f}".format(exp(3)))
print("Ou#10: {0:.2f}".format(log(4)))
print("Ou#11: {0:.1f}".format(sqrt(16)))
```

Name	Type	Size	Value
x	int	1	10
y	float	1	12.0

```
Out#3: 5.0
Out#4: 10, 1000, 5.0
Out#5: 3.074
Out#6: 12.0
Out#7: 2.67
Out#8: 2.6667
Ou#9: 20.0855
Ou#10: 1.39
Ou#11: 4.0

In [20]:
```

User input with input() function

```
>>> a=input("Give me first #: ")
Give me first #: 5
>>> b=input("Give me second #: ")
Give me second #: 10
>>> a+b
'510'
>>> int(a)+int(b)
15
>>> x=float(input(Give me a real: '))
Give me a real: 567.1234
>>> print(x)
567.1234
```

`input(prompt)`

- Parameter: prompt string
- Return value: string
- Integer input:
  `int(input("Give an integer: "))`
- Real input:
  `float(input("Give a real: "))`

Strings

Strings

```
>>> 'very ' + 'hot'
'very hot'
>>> 3*'very ' + 'hot'
'very very very hot'
>>> '7'+'2'
'72'
>>> type('dog')
<class 'str'>
>>> type('7')
<class 'str'>
>>> type(7)
<class 'int'>
```

```
>>> justaTest = '''Say,
"I'm in!"
This is line 3'''
>>> print(justaTest)
Say,
"I'm in!"
This is line 3
>>>
```

- String: Alphanumeric in single or double quotes
- Use triple single or triple double quotes for writing in multi-line strings
- Use + for string concatenation

String operations(1)

- `yourString.`**`upper`**`()` – uppercase letters
- `yourString.`**`lower`**`()` – lowercase letters
- `yourString.`**`capitalize`**`()` – capitalize only the first string letter
- `yourString.`**`title`**`()` – capitalize the first letter of every word
- `yourString.`**`replace`**`(x,y)` - replace **x** with **y**
- **`len`**`(yourString)` – length of string

More string functions:

- https://docs.python.org/3.4/library/stdtypes.html#string-methods

String operations: example

```
>>> yourString='The answer to the ultimate question of life, the universe and
    everything is 42.'
>>> yourString.upper()
'THE ANSWER TO THE ULTIMATE QUESTION OF LIFE, THE UNIVERSE AND EVERYTHING IS 42.'
>>> yourString.lower()
'the answer to the ultimate question of life, the universe and everything is 42.'
>>> yourString.capitalize()
'The answer to the ultimate question of life, the universe and everything is 42.'
>>> yourString.title()
'The Answer To The Ultimate Question Of Life, The Universe And Everything Is 42.'
>>> yourString.replace('a','A')
'the Answer to the ultimAte question of Life, the Universe And Everything is 42.'
```

Strings: more operations

myStr	T	h	e		a	n	s	w	e	r		i	s		4	2
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Every string is an *array of characters* (index starts at 0)

- `myStr[x]` - The xth character from the start (e.g `myStr[0]` is the first char)
- `myStr[start:stop]` - substring from start till stop-1
- `myStr[start:]` – substring from start till the end
- `myStr[:stop]` – substring from start till stop-1
- `myStr[:]` – the whole string
- `myStr[-x]` – The xth character from the end
- `myStr[-x:]` – the last x characters
- `myStr[:-x]` – the whole string without the last x characters