Lab -4

Programing in Python

Instructor : AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

python

What you will learn:

Debug in IDLE
Dependencies
PIP
Strings,
String and List Methods,

PYTHON INSTITUTE
Open Education & Development Group

CISCO™

Use your brain to find the answers

```
my_tup = (1, 2, 3)
print(my_tup[2])
```

```
tup = 1, 2, 3
a, b, c = tup

print(a * b * c)
```

```
my_dictionary = {"A": 1, "B": 2}
copy_my_dictionary = my_dictionary.copy()
my_dictionary.clear()
print(copy_my_dictionary)
```

```
colors = {
    "white": (255, 255, 255),
    "grey": (128, 128, 128),
    "red": (255, 0, 0),
    "green": (0, 128, 0)
    }

for col, rgb in colors.items():
    print(col, ":", rgb)
```

```
try:
value = int(input("Enter a value: "))
print(value/value)
except ValueError:
print("Bad input...")
except ZeroDivisionError:
print("Very bad input...")
except:
print("Booo!")
```

Debug in IDLE

```
## volume.py  volume calculations with torus
# Author:
# Date:
# There are libraries of common functions, for example a math library
# that has functions such as sine, cosine, logs, etc.
# There ARE syntax and semantic errors in this program as given!
from math import pi   # use the value of pi defined in the math library
def main():
    radius = input("Enter the radius: ")
    height = input("Enter the height: ")
    conevol = 1/3*pi*radius*2*height   # cone volume
    cylvol = pi*radius**2*height       # cylinder volume
    spherevol = 3/4*Pi*r**3        # sphere volume
    print ('The volume of a cone: ',conevol)
    print ("The volume of a cylinder: ",cylvol)
    print ("The volume of a sphere: ",spherevol)

main()
```

How to debug in IDLE
https://www.cs.uky.edu/~keen/help/debug-tutorial/debug.html

Dependencies

To make a long story short, we can say that dependency is a phenomenon that appears every time you're going to use a piece of software that relies on other software. Note that dependency may include (and generally does include) more than one level of software development.

How to use pip

pip help

If you want to know more about any of the listed operations, you can use the following form of pip invocation:

pip help operation -> (pip help install)

If you want to know what Python packages have been installed so far, you can use the list

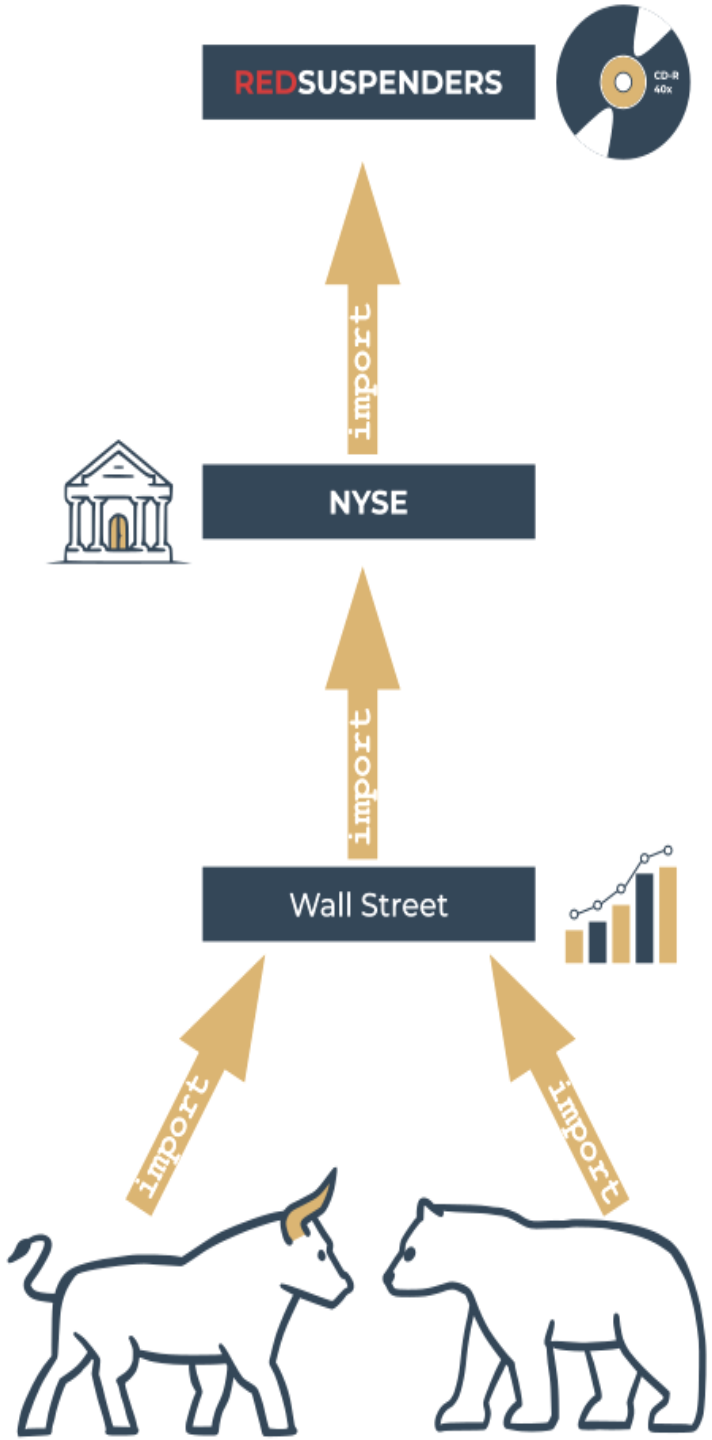
pip list

there's a command that can tell you more about any of the installed packages (note the word installed)

pip show package_name -> pip show pip

For searching about a package in pypi

pip search anystring



How to use *pip*

Before you install a package:

1. you want to install a new package for you only – it won't be available for any other user (account) existing on your computer; this procedure is the only one available if you can't elevate your permissions and act as a system administrator;
2. you've decided to install a new package system-wide – you have administrative rights and you're not afraid to use them.

https://www.youtube.com/watch?v=gmGh5aTsins&list=PLqhYARgo2Xidv82kSvJg49Kn7r1kw8hA_&index=5

pip show pygame

pip list

Command Prompt

```
C:\Users\user>pip install --user pygame
Collecting pygame
  Downloading pygame-1.9.6-cp38-cp38-win32.whl (4.4 MB)
     |                                        | 4.4 MB 88 kB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.6

C:\Users\user>
```

pip is able to update a locally installed package – e.g **pip install <package_name> --upgrade**. If you want to make sure that you're using the latest version of a particular package, you can run the following command **pip install -U package_name**

pip is also able to install a user-selected version of a package (pip installs the newest available version by default); to achieve this goal you should use the following syntax: pip install package_name==package_version (pip install pygame==1.9.2)

a simple test program

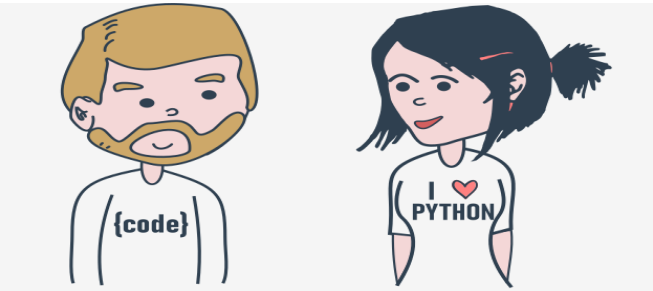
```
import pygame

run = True
width = 400
height = 100
pygame.init()
screen = pygame.display.set_mode((width, height))
font = pygame.font.SysFont(None, 48)
text = font.render("Welcome to pygame", True, (255, 255, 255))
screen.blit(text, ((width - text.get_width()) // 2, (height -
text.get_height()) // 2))
pygame.display.flip()
while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT\
        or event.type == pygame.MOUSEBUTTONUP\
        or event.type == pygame.KEYUP:
            run = False
```

How to uninstall a package?
pip unistall package_name
pip uninstall pygame

A lazy programmer is a programmer who looks for existing solutions and analyzes the available code before they start to develop their own software from scratch.

This is why PyPI and pip exist – use them!



How computers understand single characters

Computers store characters as numbers. Every character used by a computer corresponds to a unique number, and vice versa.

ASCII (short for American Standard Code for Information Interchange)

I18N

INTERNATIONALIZATION

Code points and code pages

UNICODE

UCS-4

32 bits to store each character

UTF-8

One of the most commonly used is UTF-8.

The name is derived from Unicode Transformation Format.

Python 3 fully supports Unicode and UTF-8:

1. you can use Unicode/UTF-8 encoded characters to name variables and other entities;
2. you can use them during all input and output.

This means that Python3 is completely I18Ned.

Character	Code	Character	Code	Character	Code	Character	Code
(NUL)	0	(space)	32	@	64	`	96
(SOH)	1	!	33	A	65	a	97
(STX)	2	"	34	B	66	b	98
(ETX)	3	#	35	C	67	c	99
(EOT)	4	$	36	D	68	d	100
(ENQ)	5	%	37	E	69	e	101
(ACK)	6	&	38	F	70	f	102
(BEL)	7	'	39	G	71	g	103
(BS)	8	(	40	H	72	h	104
(HT)	9	)	41	I	73	i	105
(LF)	10	*	42	J	74	j	106
(VT)	11	+	43	K	75	k	107
(FF)	12	,	44	L	76	l	108
(CR)	13	-	45	M	77	m	109
(SO)	14	.	46	N	78	n	110
(SI)	15	/	47	O	79	o	111
(DLE)	16	0	48	P	80	p	112
(DC1)	17	1	49	Q	81	q	113
(DC2)	18	2	50	R	82	r	114
(DC3)	19	3	51	S	83	s	115
(DC4)	20	4	52	T	84	t	116
(NAK)	21	5	53	U	85	u	117
(SYN)	22	6	54	V	86	v	118
(ETB)	23	7	55	W	87	w	119
(CAN)	24	8	56	X	88	x	120
(EM)	25	9	57	Y	89	y	121
(SUB)	26	:	58	Z	90	z	122
(ESC)	27	;	59	[	91	{	123
(FS)	28	<	60	\	92	|	124
(GS)	29	=	61	]	93	}	125
(RS)	30	>	62	^	94	~	126
(US)	31	?	63	_	95		127

Strings

```
# Example 1

word = 'by'
print(len(word))


# Example 2

empty = ''
print(len(empty))


# Example 3

i_am = 'I\'m'
print(len(i_am))
```

Multiline strings

```
multiline = 'Line #1
Line #2'

print(len(multiline))
```

Operations on strings

concatenated (joined)

replicated.

```
str1 = 'a'
str2 = 'b'

print(str1 + str2)
print(str2 + str1)
print(5 * 'a')
print('b' * 4)
```

More operations on strings

```
# Demonstrating the ord() function.
char_1 = 'a'
char_2 = ' '  # space
print(ord(char_1))
print(ord(char_2))
```

```
# Demonstrating the chr() function.
print(chr(97))
print(chr(945))
```

```
# Demonstrating the ord() function.
char_1 = 'a'
char_2 = ' '  # space
print(ord(char_1))
print(ord(char_2))
```

```
# Indexing strings.
the_string = 'silly walks'
for ix in range(len(the_string)):
    print(the_string[ix], end=' ')
print()
```

```
# Iterating through a string.
the_string = 'silly walks'
for character in the_string:
    print(character, end=' ')
print()
```

```
# Slices

alpha = "abdefg"
print(alpha[1:3])
print(alpha[3:])
print(alpha[:3])
print(alpha[3:-2])
print(alpha[-3:4])
print(alpha[::2])
print(alpha[1::2])
```

```
alphabet = "abcdefghijklmnopqrstuvwxyz"

print("f" in alphabet)
print("F" in alphabet)
print("1" in alphabet)
print("ghi" in alphabet)
print("Xyz" in alphabet)
```

Python strings are immutable

```
alphabet = "abcdefghijklmnopqrstuvwxyz"
del alphabet[0]
alphabet.append("A")
alphabet.insert(0, "A")
```

```
# Demonstrating min() - Example 1:
print(min("aAbByYzZ"))

# Demonstrating min() - Examples 2 & 3:
t = 'The Knights Who Say "Ni!"'
print('[' + min(t) + ']')

t = [0, 1, 2]
print(min(t))
```

```
# Demonstrating max() - Example 1:
print(max("aAbByYzZ"))

# Demonstrating max() - Examples 2 & 3:
t = 'The Knights Who Say "Ni!"'
print('[' + max(t) + ']')

t = [0, 1, 2]
print(max(t))
```

```
# Demonstrating the list() function:
print(list("abcabc"))
```

Methods on strings

```
# Demonstrating the index() method:
print("aAbByYzZaA".index("b"))
print("aAbByYzZaA".index("Z"))
print("aAbByYzZaA".index("A"))
```

```
# Demonstrating the count() method:
print("abcabc".count("b"))
print('abcabc'.count("d"))
```

```
# Demonstrating the capitalize() method:
print('aBcD'.capitalize())
```

```
# Demonstrating the center() method:
print('[' + 'alpha'.center(10) + ']')
```

```
# Demonstrating the endswith() method:
if "epsilon".endswith("on"):
    print("yes")
else:
    print("no")
```

Methods on strings

```
# Demonstrating the find() method:
print("Eta".find("ta"))
print("Eta".find("mma"))
print('kappa'.find('a', 2))
```

```
1
-1
4
```

```
the_text = """A variation of the ordinary lorem ipsum
text has been used in typesetting since the 1960s
or earlier, when it was popularized by advertisements
for Letraset transfer sheets. It was introduced to
the Information Age in the mid-1980s by the Aldus Corporation,
which employed it in graphics and word-processing templates
for its desktop publishing program PageMaker (from Wikipedia)"""

fnd = the_text.find('the')
while fnd != -1:
    print(fnd)
    fnd = the_text.find('the', fnd + 1)
```

```
15
80
198
221
238
```

```
# Demonstrating the isalnum() method:
print('lambda30'.isalnum())
print('lambda'.isalnum())
print('30'.isalnum())
print('@'.isalnum())
print('lambda_30'.isalnum())
print(''.isalnum())
```

```
True
True
True
False
False
False
```

```
# Example 1: Demonstrating the isapha() method:
print("Moooo".isalpha())
print('Mu40'.isalpha())

# Example 2: Demonstrating the isdigit() method:
print('2018'.isdigit())
print("Year2019".isdigit())
```

```
True
False
True
False
```

```
# Example 1: Demonstrating the islower() method:
print("Moooo".islower())
print('moooo'.islower())

# Example 2: Demonstrating the isspace() method:
print(' \n '.isspace())
print(" ".isspace())
print("mooo mooo mooo".isspace())

# Example 3: Demonstrating the isupper() method:
print("Moooo".isupper())
print('moooo'.isupper())
print('MOOOO'.isupper())
```

```
# Demonstrating the join() method:
print(",".join(["omicron", "pi", "rho"]))
```

Methods on strings

```
# Demonstrating the lower() method:
print("SiGmA=60".lower())
```

```
sigma=60
```

```
# Demonstrating the rstrip() method:
print("[" + " upsilon ".rstrip() + "]")
print("cisco.com".rstrip(".com"))
```

```
[ upsilon]
cis
```

```
# Demonstrating the lstrip() method:
print("[" + " tau ".lstrip() + "]")
```

```
[tau ]
```

```
# Demonstrating the split() method:
print("phi      chi\npsi".split())
```

```
['phi', 'chi', 'psi']
```

```
# Demonstrating the replace() method:
print("www.netacad.com".replace("netacad.com", "pythoninstitute.org"))
print("This is it!".replace("is", "are"))
print("Apple juice".replace("juice", ""))
```

```
www.pythoninstitute.org
Thare are it!
Apple
```

```
# Demonstrating the startswith() method:
print("omega".startswith("meg"))
print("omega".startswith("om"))
print()
# Demonstrating the strip() method:
print("[" + "   aleph   ".strip() + "]")
```

```
False
True
[aleph]
```

```
# Demonstrating the rfind() method:
print("tau tau tau".rfind("ta"))
print("tau tau tau".rfind("ta", 9))
print("tau tau tau".rfind("ta", 3, 9))
```

```
8
-1
4
```

```
# Demonstrating the swapcase() method:
print("I know that I know nothing.".swapcase())
print()
# Demonstrating the title() method:
print("I know that I know nothing. Part 1.".title())
print()
# Demonstrating the upper() method:
print("I know that I know nothing. Part 2.".upper())
```

```
i KNOW THAT i KNOW NOTHING.

I Know That I Know Nothing. Part 1.

I KNOW THAT I KNOW NOTHING. PART 2.
```

Comparing strings

Python's strings can be compared using the same set of operators which are in use in relation to numbers.

==	'alpha' == 'alpha' 'alpha' != 'Alpha'
!=	
>	'alpha' < 'alphabet'
>=	
<	
<=	'beta' > 'Beta'

'10' == '010' '10' > '010' '10' > '8' '20' < '8' '20' < '80'	False True False True True

'10' == 10 '10' != 10 '10' == 1 '10' != 1 '10' > 10	False True False True TypeError exception

```
# Demonstrating the sorted() function:
first_greek = ['omega', 'alpha', 'pi', 'gamma']
first_greek_2 = sorted(first_greek)

print(first_greek)
print(first_greek_2)

print()

# Demonstrating the sort() method:
second_greek = ['omega', 'alpha', 'pi', 'gamma']
print(second_greek)

second_greek.sort()
print(second_greek)
```

Strings vs. numbers

```
itg = 13
flt = 1.3
si = str(itg)
sf = str(flt)

print(si + ' ' + sf)
```

13 1.3

```
si = '13'
sf = '1.3'
itg = int(si)
flt = float(sf)

print(itg + flt)
```

14.3

Example: The IBAN Validator

```
# IBAN Validator.

iban = input("Enter IBAN, please: ")
iban = iban.replace(' ','')

if not iban.isalnum():
   print("You have entered invalid characters.")
elif len(iban) < 15:
   print("IBAN entered is too short.")
elif len(iban) > 31:
   print("IBAN entered is too long.")
else:
   iban = (iban[4:] + iban[0:4]).upper()
   iban2 = ''
   for ch in iban:
      if ch.isdigit():
         iban2 += ch
      else:
         iban2 += str(10 + ord(ch) - ord('A'))
   iban = int(iban2)
   if iban % 97 == 1:
      print("IBAN entered is valid.")
   else:
      print("IBAN entered is invalid.")
```

Example: The Caesar

```
# Caesar cipher.
text = input("Enter your
message: ")
cipher = ''
for char in text:
   if not char.isalpha():
      continue
   char = char.upper()
   code = ord(char) + 1
   if code > ord('Z'):
      code = ord('A')
   cipher += chr(code)

print(cipher)
```

```
# Caesar cipher - decrypting a message.
cipher = input('Enter your cryptogram: ')
text = ''
for char in cipher:
   if not char.isalpha():
      continue
   char = char.upper()
   code = ord(char) - 1
   if code < ord('A'):
      code = ord('Z')
   text += chr(code)

print(text)
```

Lab 4.1

Scenario

As you probably know, Sudoku is a number-placing puzzle played on a 9x9 board. The player has to fill the board in a very specific way:

- each row of the board must contain all digits from 0 to 9 (the order doesn't matter)
- each column of the board must contain all digits from 0 to 9 (again, the order doesn't matter)
- each of the nine 3x3 "tiles" (we will name them "sub-squares") of the table must contain all digits from 0 to 9.

If you need more details, you can find them here.

Your task is to write a program which:

- reads 9 rows of the Sudoku, each containing 9 digits (check carefully if the data entered are valid)
- outputs Yes if the Sudoku is valid, and No otherwise.

https://www.youtube.com/watch?v=tvP_FZ-D9Ng

https://github.com/kying18/sudoku/blob/main/sudoku.py

Input	output-> yes
295743861	
431865927	
876192543	
387459216	
612387495	
549216738	
763524189	
928671354	
154938672	

See you Next week ☺