



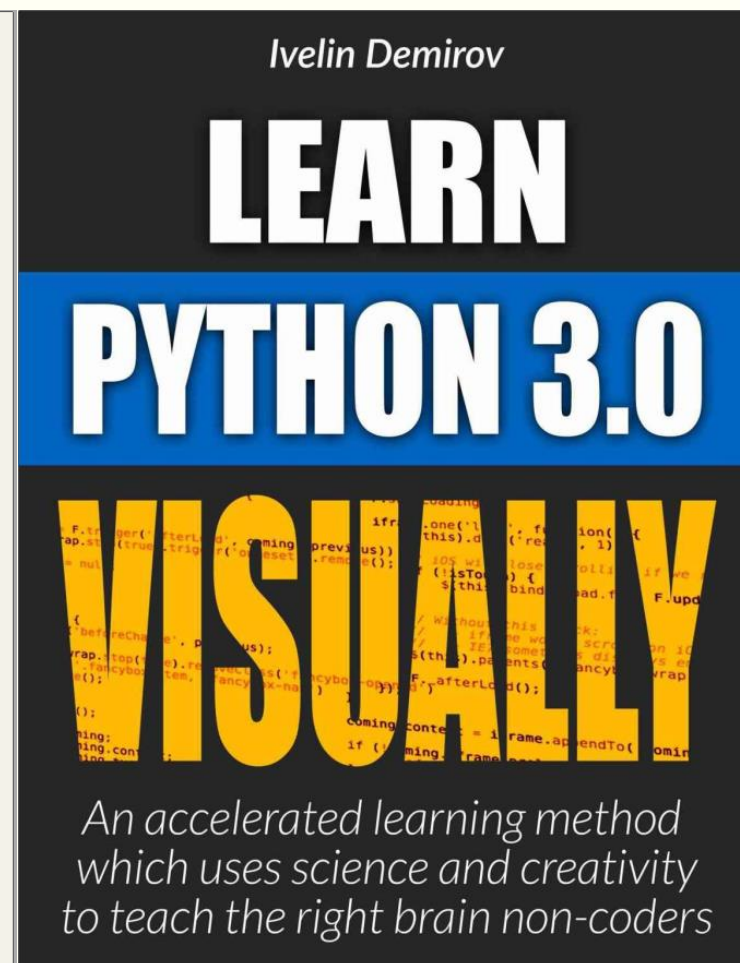
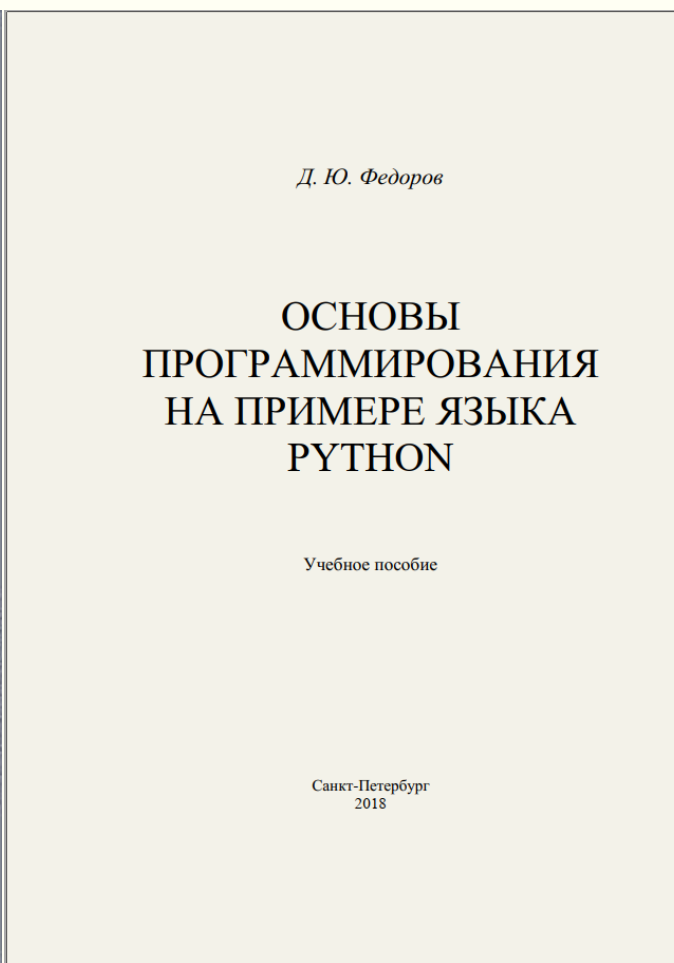
ВСТУП ДО РОЗРОБКИ МОВОЮ PYTHON

Тема 05

План лекції

- Організація розробки Python-додатків
- Структура програми мовою Python
- Управляючі оператори мови програмування Python

Рекомендована література





ОРГАНІЗАЦІЯ РОЗРОБКИ PYTHON- ДОДАТКІВ

Питання 5.1.

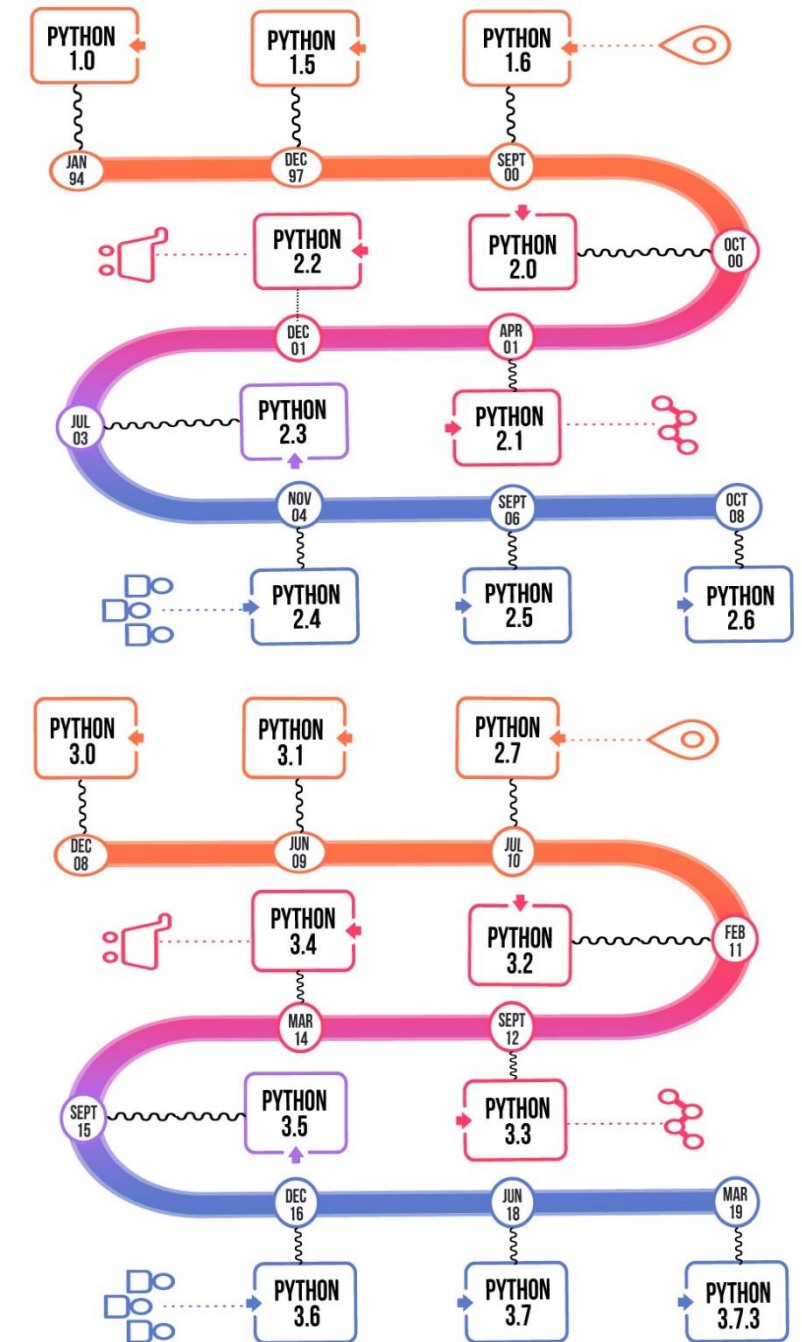
Мова програмування Python

■ *Має відкритий код*

- Еталонна реалізація – [CPython](#)
- Має реалізації різними мовами: [Jython](#), [IronPython](#), [CLPython](#), [PyPy](#), [Cython](#)
- Має хорошу розширюваність, дозволяє писати Python-код всередині програм, написаних іншими мовами програмування

■ *Високорівнева мова програмування*

- Має простий та компактний синтаксис
- Динамічно типізована, інтерпретована мова програмування
- Портативний код
- Імплементує різні парадигми програмування (ООП, функціональне програмування, реактивне програмування)
- Має широкий набір стандартних бібліотек



Початок роботи

- Завантажте інсталяційний пакет Python
 - Включає консоль Python IDLE для розробки простих додатків



- Більш потужні інструменти розробки:

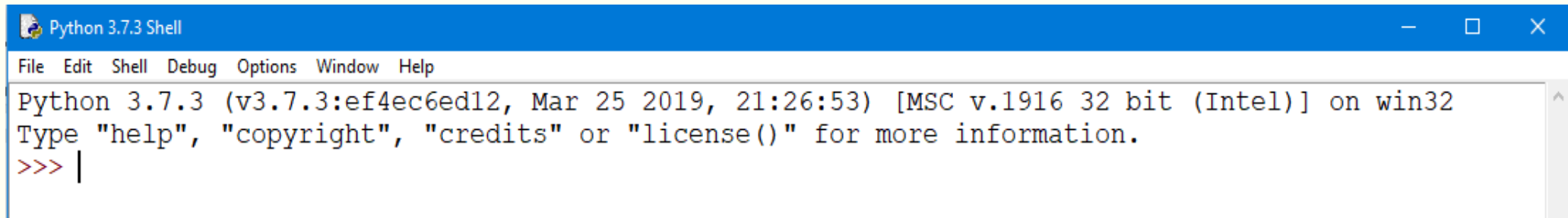
- PyCharm



- Anaconda (редактор коду Spyder)

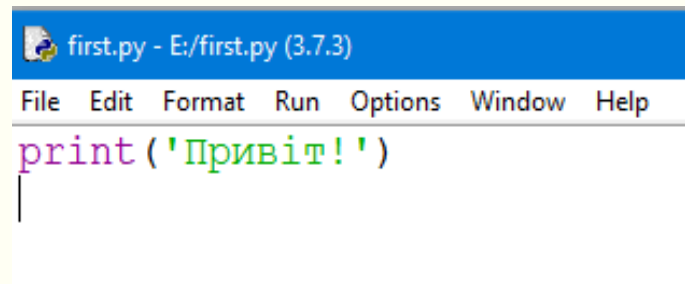


Python IDLE



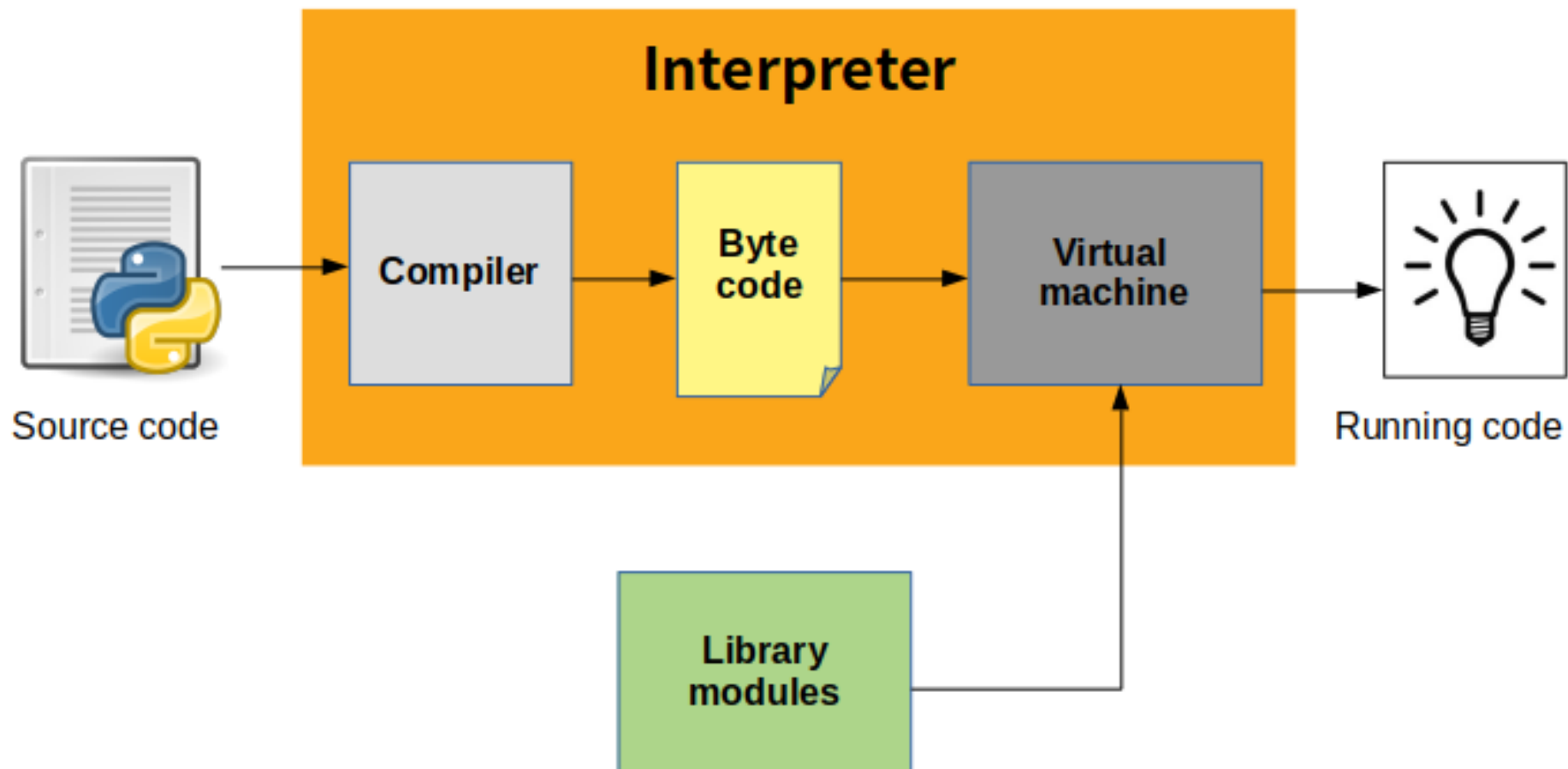
```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

- Для створення Python-скрипту перейдіть у меню File – New File або натисніть Ctrl+N
 - Для запуску на компіляцію натисніть F5

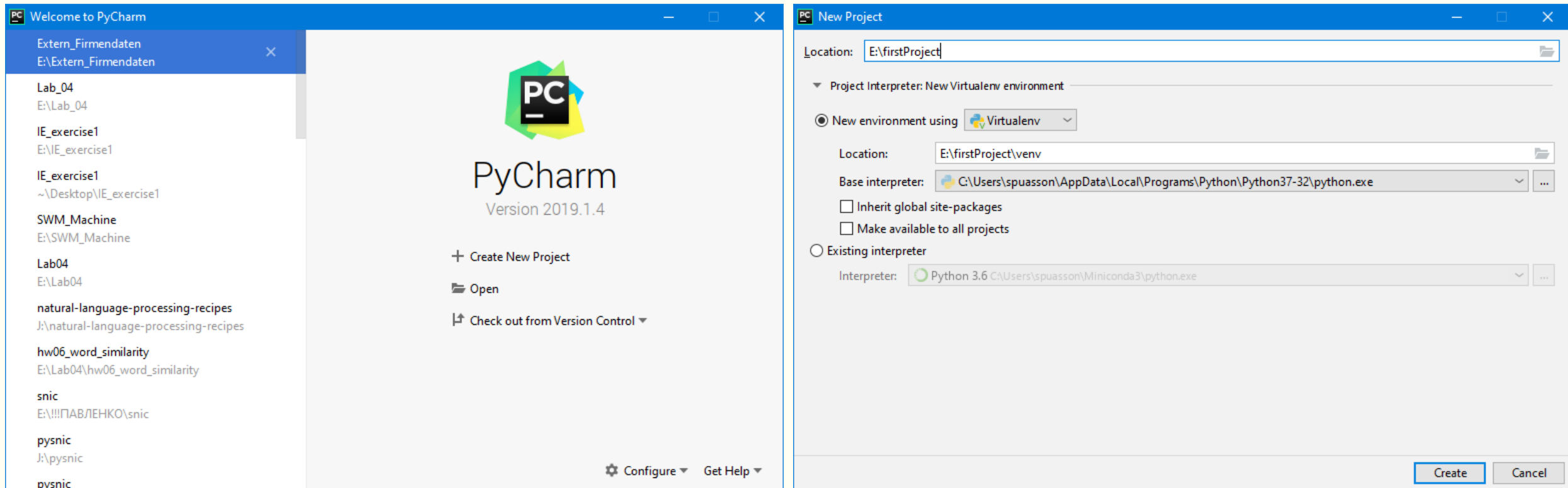


```
first.py - E:/first.py (3.7.3)
File Edit Format Run Options Window Help
print('Привіт!')
```

Схема виконання коду мовою Python

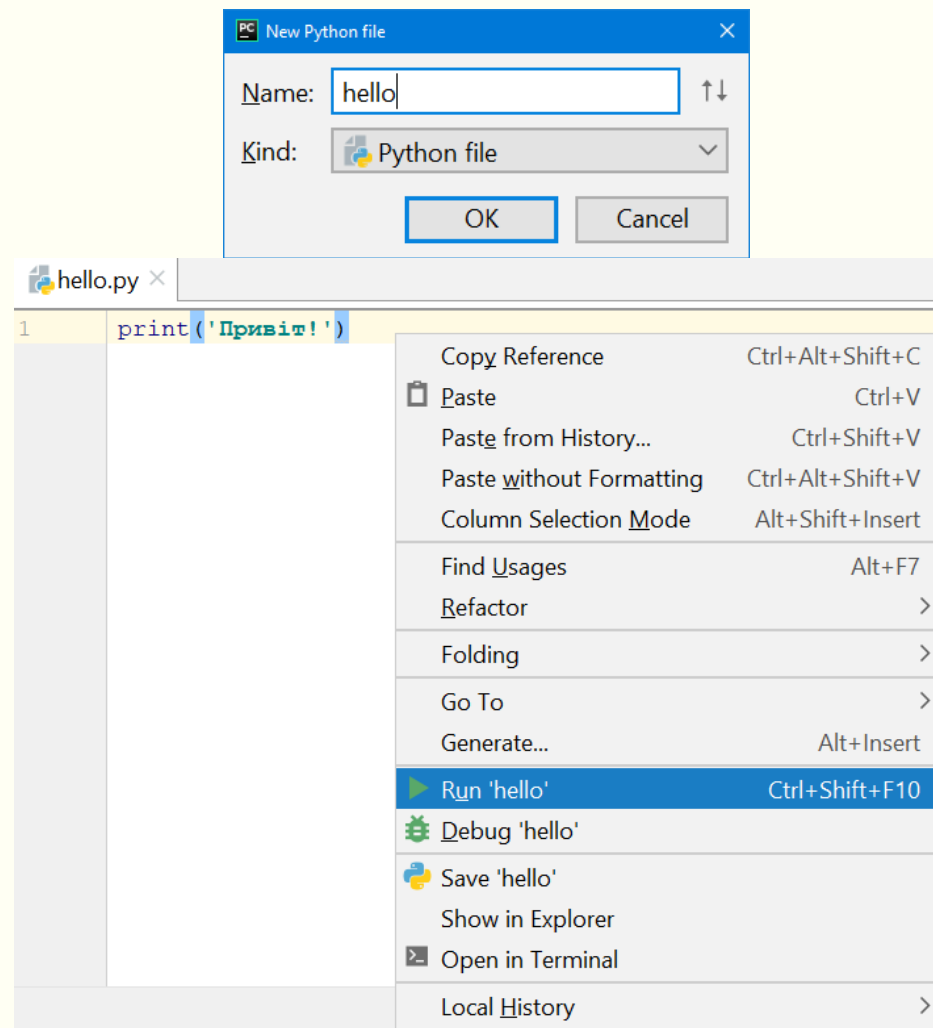
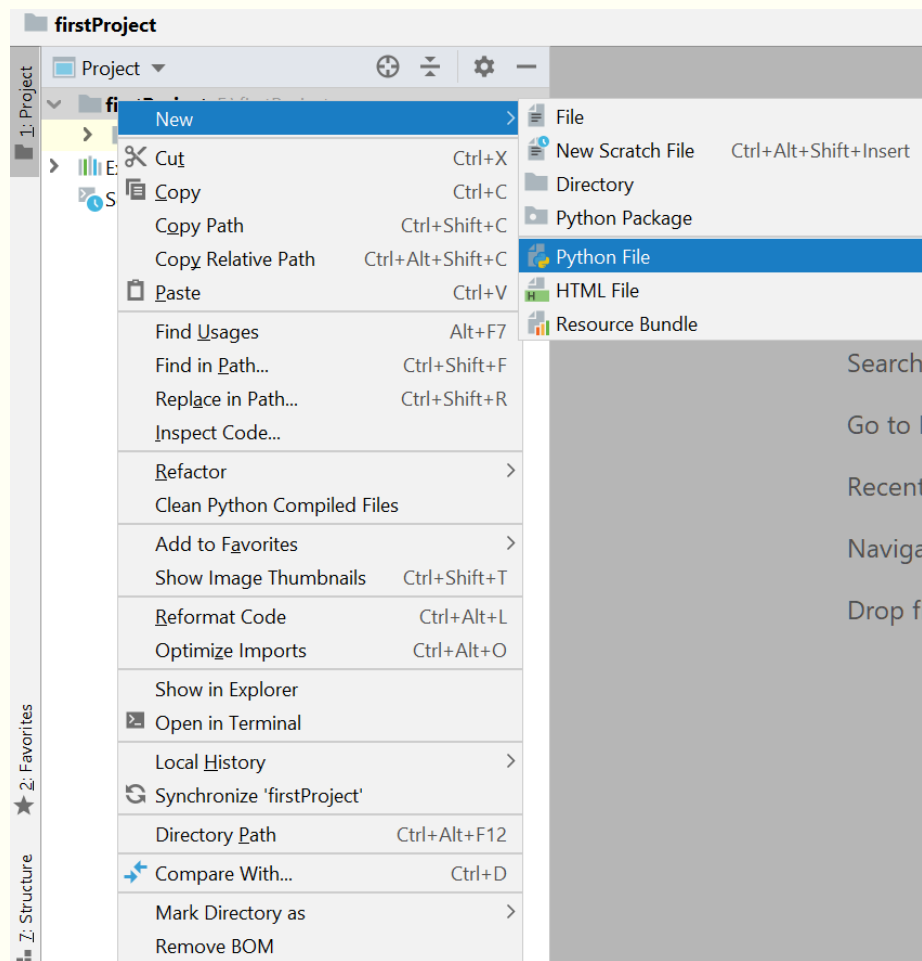


PyCharm – створення проекту

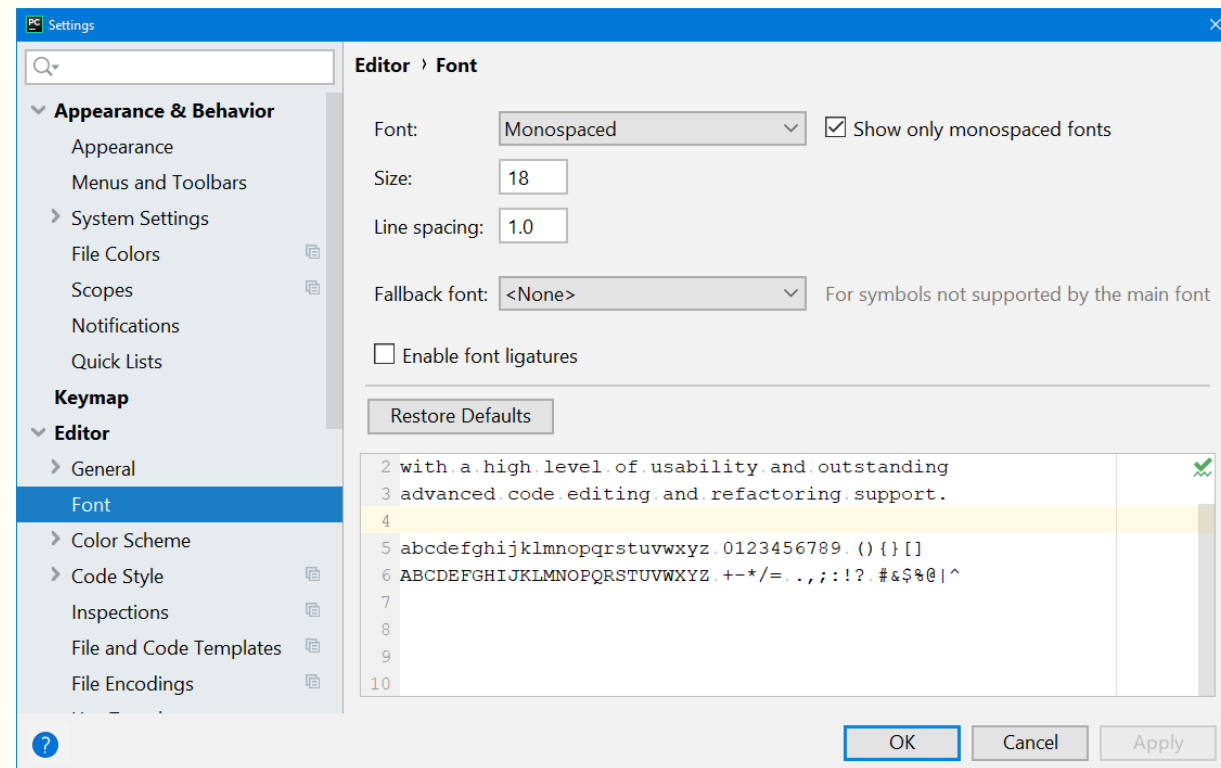
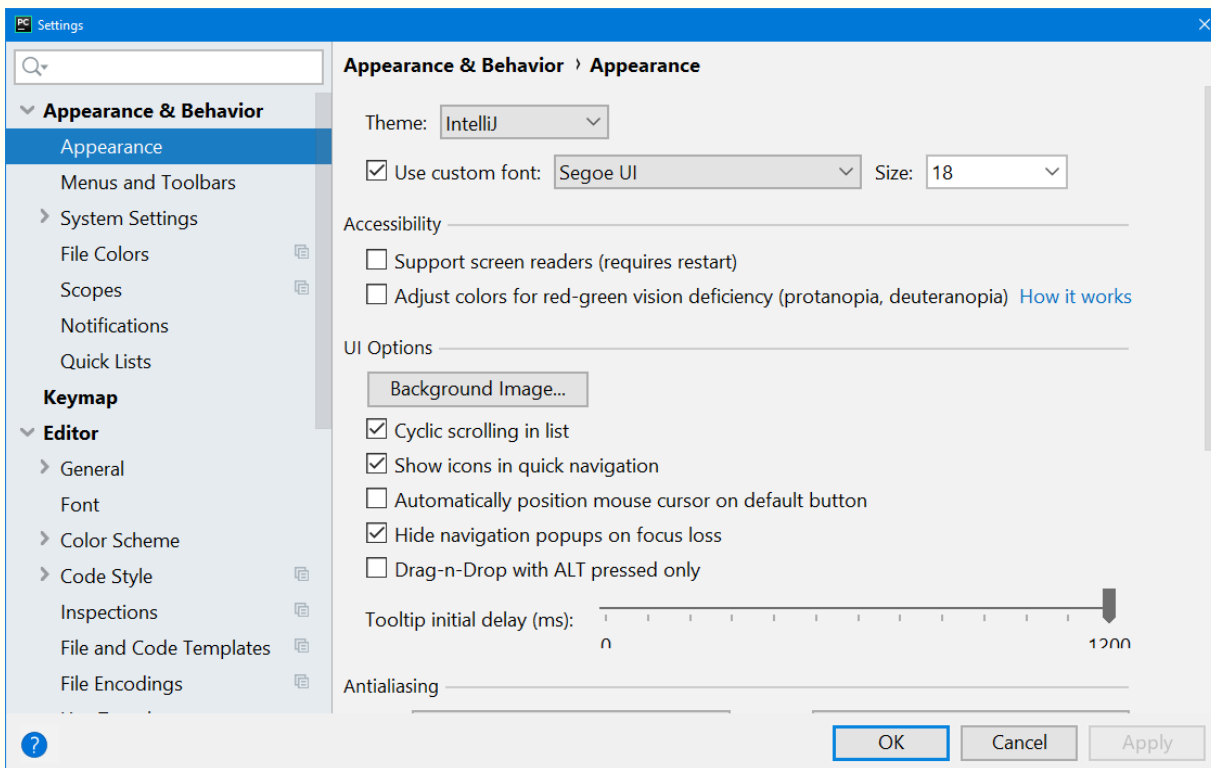


- Натисніть Create New Project та налаштуйте початковий проект.
 - Virtualenv – ізольоване віртуальне середовище для проектів Python

Доповнення проекту новим скриптом



Налаштування для тексту та підписів



Типи даних у мові Python

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey" : "value" , "name" : "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

Числові типи

- три числових типи: integers, floating point number і complex number.
 - Логічний тип даних є підтипом цілих чисел.
 - Цілі числа мають необмежену точність
 - Дробові числа зазвичай реалізовано за допомогою double в C;
 - У комплексних числах – реальна та уявна частина (дробові числа).
 - Для їх отримання з комплексного числа z , використовуйте $z.real$ і $z.imag$.
 - Стандартна бібліотека включає додаткові числові типи:
 - [fractions](#) – зберігає раціональні числа,
 - [decimal](#) – зберігає числа з плаваючою крапкою з визначеною користувачем точністю.

```
a = 2
print("Тип змінної a: ", type(a))

b = 2.0
print("\nТип змінної b: ", type(b))

c = 2 + 4j
print("\nТип змінної c: ", type(c))
```

```
Тип змінної a:  <class 'int'>

Тип змінної b:  <class 'float'>

Тип змінної c:  <class 'complex'>
```

Рядки в мові Python

- Масиви байтів, які представляють Unicode-символи.
 - Символьного типу в Python немає – є рядок з довжиною 1
 - Доступ до символів відбувається за індексом

G	E	E	K	S	F	O	R	G	E	E	K	S
0	1	2	3	4	5	6	7	8	9	10	11	12
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	

```
str1 = "GeeksForGeeks"  
print(str1[-1])  
print(str1[3:-2])
```

```
s  
ksForGee
```

```
str1 = 'Вітаю в світі Python-рядків!'  
print(str1)
```

```
str1 = "Це рядок у подвійних лапках"  
print(str1)
```

```
str1 = '''"Потрійні" лапки теж згодяться!'''  
print(str1)
```

```
str1 = '''Якщо  
        Хочете,  
        Можна й так!'''  
print(str1)
```

```
E:\firstProject\venv\Scripts\python.exe E:/firstProject/hello.py  
Вітаю в світі Python-рядків!  
Це рядок у подвійних лапках  
"Потрійні" лапки теж згодяться!  
Якщо  
        Хочете,  
        Можна й так!
```

Маніпулювання рядками

▪ Оновлення рядка

- Присвоєння символів чи підрядків не допускається

```
str1 = "GeeksForGeeks"  
str1[2] = 'p'
```



```
Traceback (most recent call last):  
  File "E:/firstProject/hello.py", line 2, in <module>  
    str1[2] = 'p'  
TypeError: 'str' object does not support item assignment
```

- Можна повністю оновити рядок

```
str1 = "Це старий рядок"  
print(str1)  
str1 = "Це новий рядок"  
print(str1)
```



```
Це старий рядок  
Це новий рядок
```

Маніпулювання рядками

- **Видалення рядка**

- *Видалення символів чи підрядків не допускається*

```
str1 = "Це рядок"  
del str1[2]  
print(str1)
```



```
Traceback (most recent call last):  
  File "E:/firstProject/hello.py", line 2, in <module>  
    del str1[2]  
TypeError: 'str' object doesn't support item deletion
```

- *Видалення рядка призводить до його повної втрати*

```
str1 = "Це рядок"  
del str1  
print(str1)
```



```
Traceback (most recent call last):  
  File "E:/firstProject/hello.py", line 3, in <module>  
    print(str1)  
NameError: name 'str1' is not defined
```


Форматування рядків

```
str1 = "{} {} {}".format('Це', 'новий', 'рядок')
print(str1)

# позиційне форматування
str1 = "{1} {0} {2}".format('це', 'новий', 'рядок')
print(str1)

# іменоване (keyword) форматування
str1 = "{l} {f} {g}".format(g='це', f='новий',
l='рядок')
print(str1)
```

Це новий рядок
новий це рядок
рядок новий це

Форматування рядків

▪ Представлення чисел:

```
# двійкове представлення  
str1 = "{0:b}".format(16)  
print(str1)  
  
# експоненційне представлення  
str1 = "{0:e}".format(165.6458)  
print(str1)  
  
# округлення чисел  
str1 = "{0:.2f}".format(1 / 6)  
print(str1)
```

```
10000  
1.656458e+02  
0.17
```

Форматування рядків

```
# вирівнювання рядків
str1 = "|{:<10}|{: ^10}|{:>10}|".format('Це', 'новий', 'рядок')
print(str1)
str1 = "\n{0: ^16} затонув у {1: <4} році!".format("Титанік", 1912)
print(str1)
```

```
|Це          |   новий   |   рядок|
\n          затонув у 1912 році!
```

Ідентифікатори

- Ідентифікатори - це імена, що задаються в програмі для змінних, типів, функцій і міток.
 - Для утворення ідентифікатору використовуйте латинські літери, цифри та символ `_`.
 - Наприклад, `shapeClass`, `shape_1`, `upload_shape_to_db` – коректні ідентифікатори.
 - З цифр ідентифікатор не може починатись – буде синтаксична помилка.
 - Ім'я `0Shape` некоректне, проте `shape1` – нормальне.
 - Ключові слова (Keywords) зарезервовані.
 - Не використовуйте їх для іменування.
- Ідентифікатори в Python не можуть містити спеціальних символів [`.'`, `!`, `@`, `#`, `$`, `%`] у назві.

```
>>> for=1
SyntaxError: invalid syntax
>>> True=1
SyntaxError: can't assign to keyword
```

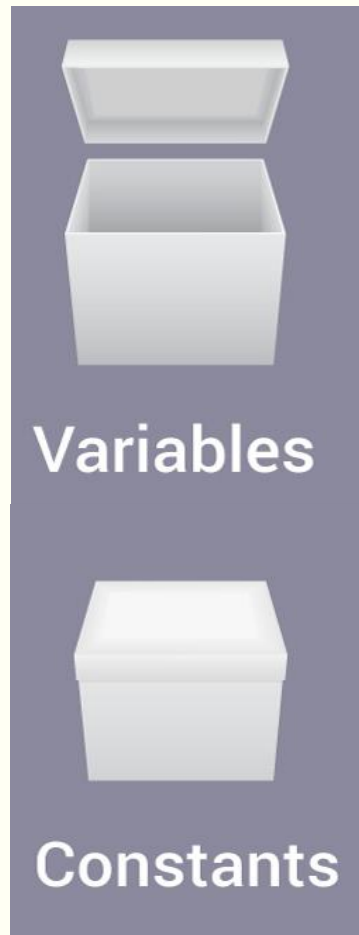
```
>>> @index=0
SyntaxError: invalid syntax
>>> isPython?=True
SyntaxError: invalid syntax
```

Рекомендації зі створення ідентифікаторів

- Класи рекомендують називати з великої літери. Решту ідентифікаторів – з малої.
- Приватні ідентифікатори іменують з символу ‘_’.
- Не використовуйте ‘_’ у першому *та* останньому символах імені. Така нотація вже використовується в Python для вбудованих типів.
- Уникайте імен з одного символу. Краще створювати змістовні назви.
 - Наприклад, `i = 1` – коректне ім’я, проте краще писати `iter = 1` або `index = 1`.
- Використовуйте символ ‘_’, щоб комбінувати кілька слів в одному імені.
 - Наприклад, `count_no_of_letters`.

- | Year | 2010 | 2011 | 2012 | 2013 | 2014 |
|--------------|------|------|------|------|------|
| Share of GDP | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 |

Змінні та константи



- Окремого синтаксису для констант у Python немає!
 - Є змінювані (mutable) та незмінювані (immutable) об'єкти.
 - Хоча присутні вбудовані константи: False, True, None, NotImplemented, __debug__, Ellipsis та ін.



ДЯКУЮ ЗА УВАГУ!

Наступне питання: структура програми мовою Python