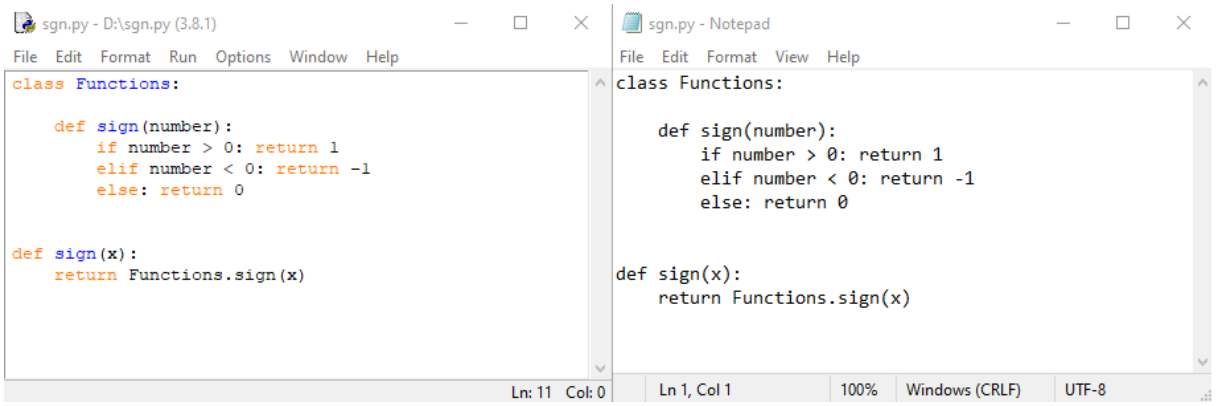
Wstęp do ćwiczeń z Pythona 3

Na zajęciach będziemy korzystali z **Pythona 3**+. Właściwa wersja jest bardzo istotna, ponieważ Python 2, zainstalowany domyślnie w wielu dystrybucjach Linuxa oferuje inną funkcjonalność niż jego młodszy brat.

Na potrzeby ćwiczeń możemy przyjąć, że język Python jest językiem **interpretowalnym** w przeciwieństwie do języków takich jak C/Cpp/C#, Java, czy Pascal, które są językami wyłącznie kompilowalnymi. Czym to się różni? Odsyłam do strony:

https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/

Na potrzeby kursu musicie wiedzieć, że do programowania w Pythonie wystarczyłby zainstalowany **interpreter** (https://www.python.org/downloads/) oraz systemowy notatnik - podczas programowania w żadnym języku **absolutnie NIE używamy edytorów tekstu typu MS Word**! Tak, jak napisałem – da się pracować w notatniku, ale zdecydowanie wygodniej jest użyć zintegrowanego środowiska programistycznego (w skrócie IDE – Integrated Developement Enviroment).



Rys. 1. Na lewo edytor w IDLE – widać podświetloną składnię. Na prawo goły notatnik.

Wśród najważniejszych zalet oferowanych przez większość IDE można wymienić:

- kolorowanie składni
- automatyczne wcięcia – co nie oznacza, że prawidłowe!
- kolorowanie błędów
- 'wbudowany' interpreter
- proponowanie zmiennych i funkcji po rozpoczęciu wpisywania – co nie oznacza, że jest to żądana funkcja lub zmienna

Rys. 1 pokazuje okno edytora kodu IDLE – najprostszego dostępnego IDE pythona, zwykle zainstalowanego domyślnie w dystrybucjach Ubuntu. Dalej pokażę jak uruchomić i używać IDLE na Windowsie. Ponadto pokażę alternatywne, bardziej rozbudowane ID – Microsoft Visual Code (VSCode). VSCode jest dostępny zarówno na Windowsa (z powyższej strony), jak i na Ubuntu – w 'sklepie' Ubuntu.

Notatnik + interpreter w terminalu

Jeżeli ktoś bardzo się upiera żeby nie korzystać z IDE to polecamy przynajmniej zainstalować notatnik z podświetlaniem składni. Propozycje:

Windows:

Notepad++ - zalety: szybki, dużo funkcjonalności; wady: żeby dobrać się do wielu funkcji często trzeba zainstalować dużo dodatków

Jedit - zalety: ma sporo fajnych funkcjonalności, działa pod Windowsem i linuxem; wady: powoli się uruchamia.

~~Notatnik wbudowany~~ - nie, po prostu nie

Linux:

Gedit - zalety: wbudowany w Ubuntu; wady: nic w nim nie ma poza składnią, nawet kolumny nie da się całej na raz przesunąć.

Jedit - jak wcześniej

Emacs - jeżeli pierwsze słyszysz to gorąco.. odradzam

vi/vim - tak, jak z Emacsem, tylko w terminalu

nano - kompletnie nie nadaje się do pisania dużej ilości kodu, ale obsługa prosta jak budowa cepa, a można na szybko poprawić jakąś literkę z poziomu terminala

Interpreter w powershellu (czy jak to tam się nazywa na Windowsie):

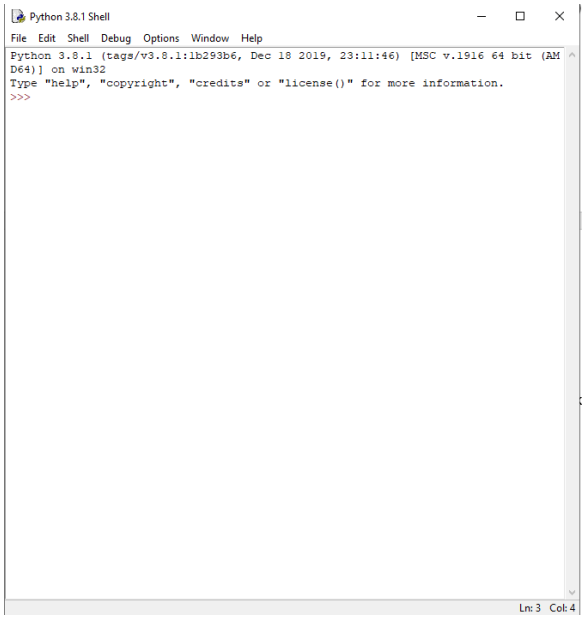
1. Wchodzimy na: https://www.python.org/downloads/ i szukamy odpowiedniej wersji (3.7+). Instalujemy, powinno samo dodać się do odpowiednich ścieżek.
2. Żeby uruchomić windowsowy terminal wpisujemy w pasku wyszukiwania cmd
3. Uruchamiamy wpisując w powershella: **python**. Jak przy uruchamianiu pokaże, że mamy taką wersję, jak pobraliśmy to fajnie. Jak pokaże jakiegoś pythona 2, to coś było wcześniej instalowane. Można spróbować poleceniem **python3**.
4. Powinniśmy cieszyć się ładnym, interaktywnym terminalem w pythonie. Żeby z niego wyjść wpisujemy **quit()** – nie, nie zamykamy i nie otwieramy okna terminala za każdym razem.
5. Co można zrobić z takiego interpretera? W sumie to niewiele, dlatego zwykle będziemy interpretować jakiś kod napisany w rzeczonym notatniku. Do tego celu użyjemy zwykle polecenia: **python sciezka_dostepowa_do_pliku**
6. Czasami chcemy sprawdzić tylko niektóre funkcje napisane w pliku. Żeby uzyskać do nich dostęp musimy dodać argument -i: **python -i sciezka_do_pliku**. Plik zostanie zinterpretowany, a następnie uruchomi się interpeter, który pozwoli uruchamiać poszczególne funkcje. W ten sposób działamy też z poziomu opisanego później VSCode.

Interpreter w Ubuntu:

1. Najpierw wpisujemy **python** i sprawdzamy, czy już nie mamy zainstalowanego pythona
2. Nie ma, albo chcemy nowsza wersję? W zaktualizowanym Ubuntu powinno wystarczyć: **sudo apt-get install python3**
3. Uruchamianie zwykle przez **python** lub **python3**, reszta jak w windzie.

IDLE

1. Instalacja identycznie jak przy pythonie w terminalu.
2. Uruchamiamy przez wyszukanie IDLE (nie w terminalu. Pojawi się terminal IDLE (Rys. 2), który w zasadzie wygląda jak interpreter z powershella/terminala.



Rys. 2. Terminal IDLE

3. Żeby zacząć pisanie klikamy File > New file. Pojawi się nowe okienko z edytorem tekstu.

Rys. 3. Edytor tekstu IDLE

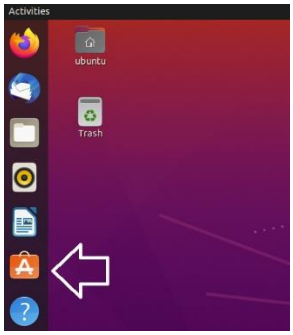
4. Żeby zinterpretować to, co napisaliśmy trzeba plik najpierw zapisać. Potem wciskamy F5 albo Run > Run module i program próbuje zinterpretować nasz kod.

VSCode

1. Instalujemy Visual Studio Code

Pobieramy ze strony https://code.visualstudio.com/ i po prostu instalujemy. Na Windowsie polecam nie kombinować ze ścieżką instalacji, bo może się okazać, że zainstalowany program będzie miał problemy z uprawnieniami.

Na Ubuntu wchodzimy w ‘sklep’ Ubuntu (Rys. 4).

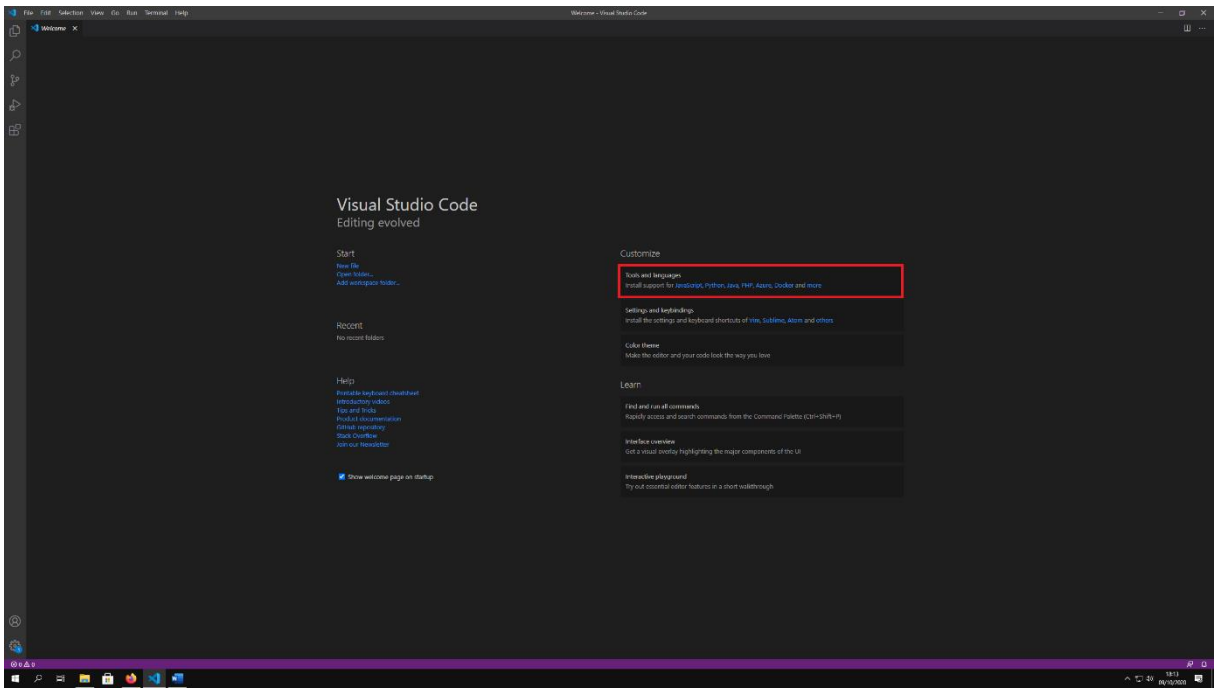


Rys. 4 ‘Sklep’ z aplikacjami pod Ubuntu

Zainstalowane VSCode jest tylko środowiskiem okienkowym dla różnych języków programowania, bez zdolności interpretacji kodu.

2. Po uruchomieniu, VSCode wygląda mniej więcej jak na Rys. 5.

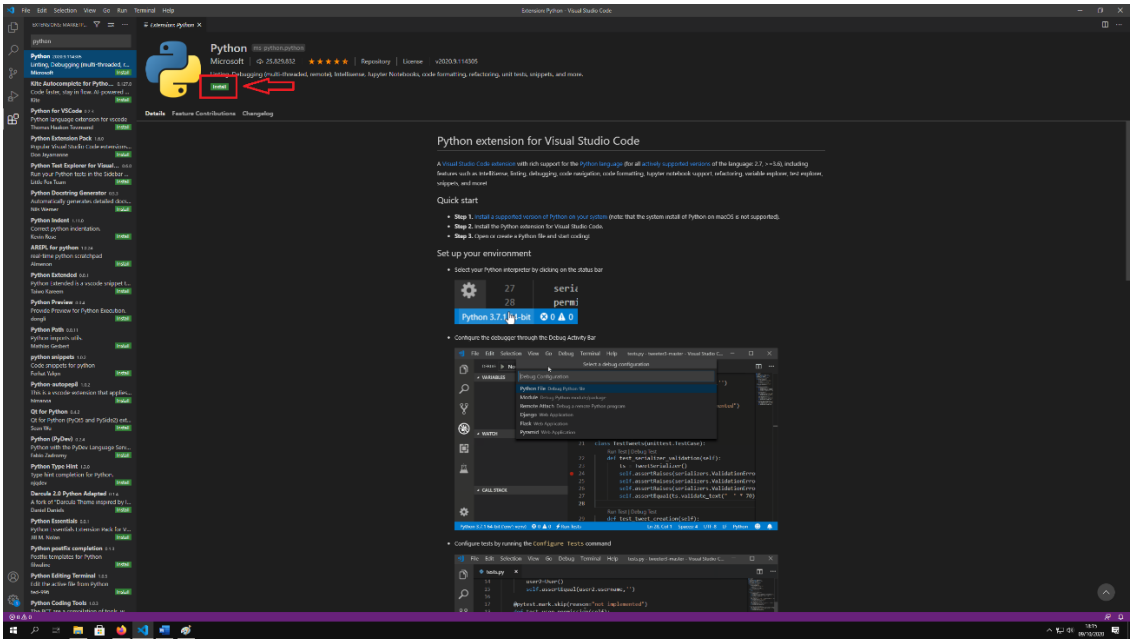
Klikamy ‘Tools and Languages’ i w otwartym panelu bocznym wpisujemy ‘python’



Rys. 5. Okienko ‘powitalne’ VSCode

3. Instalujemy podstawową paczkę Pythona

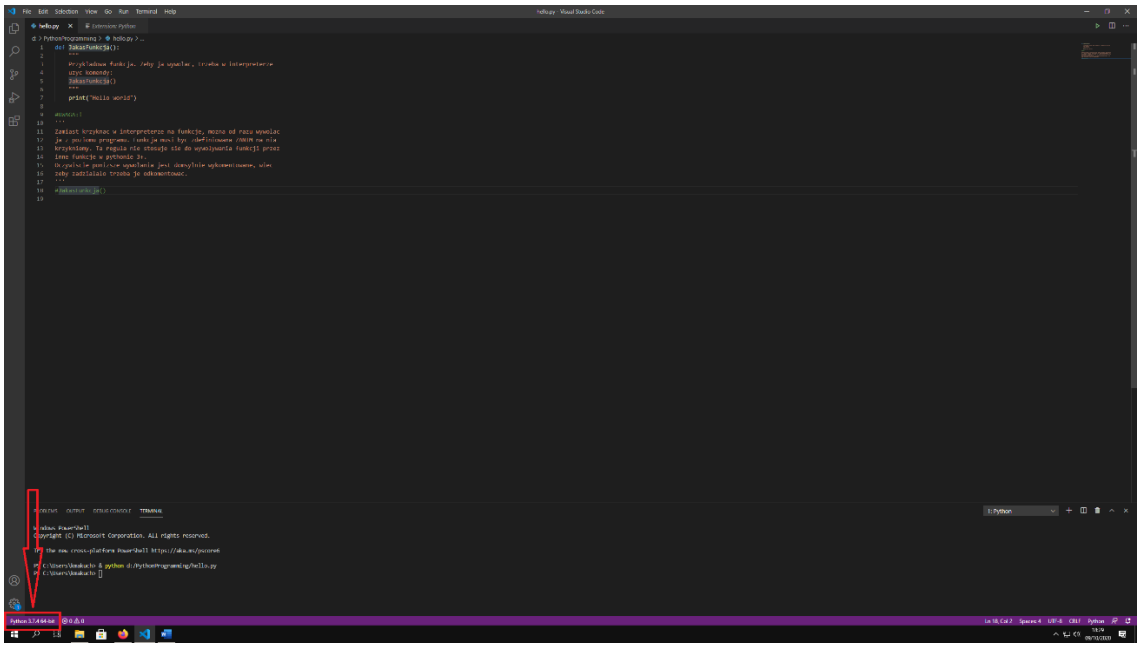
Opis paczki wygląda mniej więcej jak na Rys. 6. Na Ubuntu zamiast obrazków mogą pojawiać się dziwne, puste okienka – nie przejmujemy się tym. Na potrzeby kursu ta paczka będzie dla nas wystarczająca. Na kursach zaawansowanych prawdopodobnie wykorzystacie Anacondę.



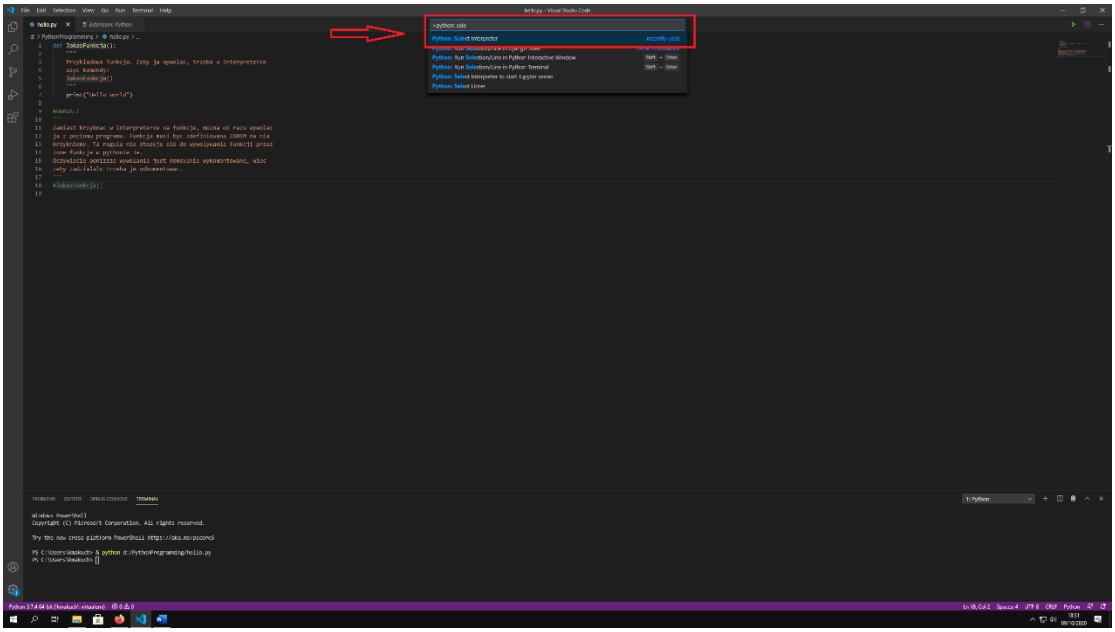
Rys. 6. Paczka Pythona, której szukamy.

4. Wybieramy właściwy interpreter

Są dwa sposoby przedstawione na Rys. 7 i Rys. 8.



Rys. 7. Klikacie na napis Python X.X. w lewym dolnym rogu i w wyświetlonym oknie wybieracie jakąś wersję Pythona 3. Na potrzeby kursu 3.7 lub wyższa.



Rys. 8. Wciskacie Ctrl+Shift+P i pokazuje wam się terminal podobny do tego na screenie. Wpisujecie python: select interpreter po czym wybieracie.

5. Uruchamianie programów

Samo uruchamianie interpretera jest domyślne i prowadzi do uruchomienia napisanego programu we wbudowanym terminalu. Z napisanym programem można pracować podobnie jak wcześniej opisano przy interpreterze uruchamianym bezpośrednio z terminala.

Informacje dodatkowe:

1. Nie używamy polskich znaków, chyba że ktoś jest masochistą. Nawet w komentarzach. Po prostu nie.
2. Nie używamy spacji w nazwach plików, ścieżkach itd.

Dodatkowe informacje dodatkowe – czyli truizmy prowadzącego:

Programowanie jest jak matematyka – albo jesteście systematyczni, albo kompletnie nie wiecie co się dzieje. Jeżeli nie opanujecie zajęć 1, to nie poradzicie sobie z 2. Jeżeli nie opanujecie zajęć 1 i 2, to nie poradzicie sobie z 3 itd.

Materiały dodatkowe to nie są zadania ‘z gwiazdką, dla chętnych, na szóstkę’. To są materiały, które mają ułatwić Wam samodzielne nauczenie się programowania.

Choćbyście pili 10 espresso dziennie, nie da się nauczyć programowania ‘na raz’, przed zaliczeniem w tydzień.

Jesteście na bardzo intensywnym kursie programowania. Proszę nie myśleć przez pryzmat ‘zakuć, zdać, zapomnieć’, bo to nie biochemia. 95% pracy to jest Wasza praca własna. My skupiamy się na przekazaniu pewnych umiejętności i praktyk, które dobry programista powinien posiadać.

Kody są proste i wydawałoby się, że 2 osoby mogą napisać dokładnie to samo. Mogą, jasne. A my możemy zapytać o ten kod. Jak ktoś nie umie obronić swojego kodu to dostaje 0.

Jak zawsze znajdą się osoby, które uznają, że ich to nie dotyczy. Jak zawsze znowu się spotkamy.