

# PyLadies.start()



23-24.09.2017

Poznań

# O Pyladies

Kochane PyLadies,

Bardzo chciałabym być dzisiaj z Wami, ale zrządzeniem losu muszę być gdzie indziej. Kibicuję każdej i każdemu z Was. Jeżeli obawiacie się, że sobie nie poradzicie - wybijcie to sobie z głów. Niech każda pojedyńcza działająca linijka kodu będzie dla Was powodem do dumy.

Jeżeli nie będziecie czegoś rozumieć, jeżeli nasze tłumaczenie będzie niejasne, jeżeli tempo będzie za wysokie - to jest nasza wina! Pomóżcie nam się poprawić. Za każdym razem gdy na czymś utkniecie, zgłaszajcie się! Jeżeli w trakcie przedstawiania zadania coś będzie niejasne - pytajcie! Jeżeli na konsoli pokaże się Wam tajemniczy błąd, z którym nie będziecie wiedzieli co zrobić - proście o wsparcie!

Nie wstydźcie się żadnego pytania, nie ukrywajcie błędów, pokazujcie swój kulawy kod, nie myślcie, że powinno Wam iść lepiej. Jesteście tutaj po to żeby się czegoś nauczyć i wyciśnijcie z tych warsztatów co tylko możecie. Rozmawiajcie ze sobą! Może osoba z waszej prawej bądź lewej strony może Wam pomóc przeskoczyć jakąś przeszkodę? Może Wy możecie pomóc osobie po Waszej prawej lub lewej?

Osobie, która czuje lekką obawę przed zgłoszeniem się, albo która boi się przyznać, że się zgubiła w zadaniu. Może poznacie w ten sposób osoby, z którymi będącie uczyć się razem po warsztatach? Wszyscy mentorzy na sali są tutaj dla Was. Nam, może nawet bardziej niż Wam, zależy żebyście wyszli z tych warsztatów z energią, radością i zacięciem do dalszej nauki.

Jeżeli kiedykolwiek przez myśl przejdzie Wam, że się do programowania nie nadajecie - niech rozbrzmi Wam wtedy w głowie mój głos: BZDURA! Pytanie czy to programowanie się dla Was nadaje. Jeżeli macie z tego frajdę, jeżeli was to jara, to się nadajecie! Błędy w kodzie są nieodłącznym elementem programowania. Wgryzanie się w nie, żmudne zliczanie nawiasów, szukanie zagubionych przecinków i poprawianie wcieć jest obowiązkowym elementem

Rok temu siedziałam tu gdzie Wy. Wczoraj prezentowałam pierwszą aplikację jaką współtworzyłam jako programistka w pracy. Czy muszę Wam mówić jaka jestem szczęśliwa? PyLadies było pierwszym krokiem jaki zrobiłam w świat programowania. Pierwszym, ale nie jedynym. Mam nadzieję, że Was PyLadies również nie zawiedzie!

A gdy już kurz opadnie, adrenalina wróci do normy, a serotonina oby nie, napiszcie do mnie i powiedzcie jak Wam się podobało. Co możemy zrobić lepiej? Albo chociaż dajcie znać z czego jesteście najbardziej dumne i dumni. Piszcie na [magda@pyladiesstart.pl](mailto:magda@pyladiesstart.pl). Albo na naszym FB.

Ściskam Was  
wszystkie i  
wszystkich  
serdecznie! Do  
zobaczenia na  
zajęciach w ciągu  
roku!

Magda



# Instrukcja obsługi warsztatu

# Agenda

8:15 - 08:45 Rejestracja

9.00 - 11.30 Warsztat

11.30 - 11.45 Przerwa kawowa

11.45 - 12.00 Lightning talk

12.00 - 14.00 Warsztat

14.00 - 14.45 Przerwa obiadowa

14.45 - 15.00 Lightning talk

15.00 - 16.30 Warsztat

15.00 - 16.30 Warsztat

16.30 - 16.45 Lightning talk

16.45 - 17.00 Przerwa kawowa

16.55 - 18.00 Warsztat

19:00 - 22.00 Afterparty w Tandem Pub

# Kontrakt

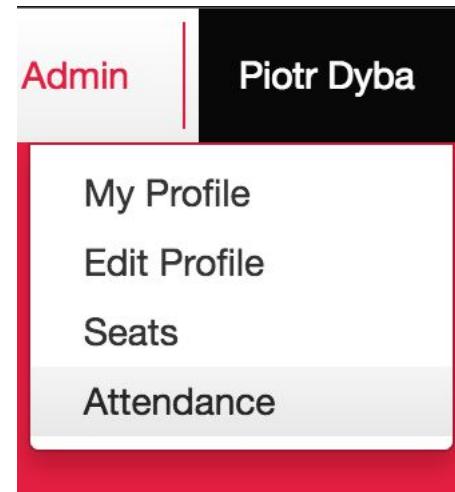
1. Co: nauka programowania.
2. Gdzie: MTP / ...
3. Kiedy: 23-24.09.2017 9:00-18:00
4. Kto: Mentorzy odpowiadają za naukę, a uczestnicy za wchłanianie wiedzy.

# Kontrakt

## 5. Jak:

- Uczymy się w Pythonie.
- Przynosimy ze sobą laptopy
- Kładziemy nacisk na praktykę.
- Prowadzący zajęcia jest wspierany przez mentorów.
- Na zakończenie zajęć dajemy feedback prowadzącemu i mentorom. Bądź też w czasie przerw.

# Aplikacja



# Aplikacja

Wybierz swoje miejsce!

SCENA



A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28 A29 A30 A31 A32 A33 A34 A35

B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 B16 B17 B18 B19 B20 B21 B22 B23 B24 B25 B26 B27 B28 B29 B30 B31 B32 B33 B34 B35

D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22 D23 D24 D25 D26 D27 D28 D29 D30 D31 D32 D33 D34 D35

E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12 E13 E14 E15 E16 E17 E18 E19 E20 E21 E22 E23 E24 E25 E26 E27 E28 E29 E30 E31 E32 E33 E34 E35

G0 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14 G15 G16 G17 G18 G19 G20 G21 G22 G23 G24 G25 G26 G27 G28 G29 G30 G31 G32 G33 G34 G35

[Regulamin](#)

[!\[\]\(9f63f5ec98cc2eddf66038fdc55c1091\_img.jpg\) GitHub](#)

Kto  
Czas się  
przedstawić 😊

# Przemek Pietrzak

Pracuje jako inżynier oprogramowania w F-Secure, używając Pythona, JavaScriptu a ostatnio Scali. Ten ostatni bo nazwał tak kotkę, więc wypadało nauczyć się języka. Na PyLadies mentoruje od 3 lat mentoruje, w tym czasie zdążył już wykształcić całe pokolenie pythonistów i kilku mentorów.

Po godzinach ogląda stare filmy, słucha Franka Sintary i pływa. Z wykształcenia maturzysta, bywa miły. Supermoc - funkcyjne programowanie i rozumienie JavaScriptu (tak, da się).

# Wojtek Mikołajczyk

Wojtek to absolwent Politechniki Poznańskiej, który obecnie pracuje jako Python Developer w Merixstudio. Ma ponad dwa lata doświadczenia w pracy nad projektami komercyjnymi.

Po pracy Wojtek relaksuje się grając na pianinie , czy słuchając muzyki (poza tym potrafi też grać na perkusji!). Dodatkowo w wolnym czasie zgłębia tajniki psychologii.



# Łukasz Banasiak

Łukasz jest full stack deweloperem. Na codzień tworzy rozwiązania napisane w Pythonie i JavaScript. Obecnie zaangażowany w Vocate.me , którego misja to “Unlocking Human Potential”. Jego kod pomaga studentom lepiej zrozumieć co chcą robić oraz pracodawcom, aby ich odnaleźć, połączyć razem i pozwolić im to robić

Wcześniej pracował zarówno dla dużych korporacji jak i małych startupów, jako administrator i programista. Duże, interdyscyplinarne doświadczenie to jego super moc.

Absolwent Politechniki Poznańskiej na kierunku Informatyka. Jego wymarzoną pracę jako dziecko było zostać koszykarzem NBA.

# Piotr Hankiewicz

Piotrek jest Back End Developerem w Merixstudio z ponad rocznym doświadczeniem zawodowym. Dwie technologie, w których czuje się jak ryba w wodzie . to Django i Docker.

Po godzinach uczy się także Reacta i Reduxa, albo spędza czas uderzając w struny gitary, czytając bądź podróżując.



# Paweł Hały

Paweł to Back End Developer z dwuletnim doświadczeniem. Obecnie pracuje w Merixstudio. Ukończył Elektrotechnikę na Politechnice Poznańskiej, gdzie stworzył koło naukowe zajmujące się programowaniem przy pomocy C i Python takiego sprzętu jak Arduino i Raspberry Pi, których jest fanem i które często wykorzystuje na własny, domowy użytek.

Paweł lubi porównywać programowanie do malowania - chociaż nośnik jest zupełnie inny, obie dziedziny wymagają kreatywnego podejścia, które jest podstawą do nieskończenie wielu możliwości. To oraz możliwość budowania znaczących i ciekawych projektów sprawia, że Paweł uwielbia to co robi



# Przemysław Gosek

Przemek pracuje jako JavaScript Developer. Pasjonują go nowe technologie i zawsze stara się być na bieżąco z nowinkami. Na co dzień zaraża innych swoją pogodą ducha, zawsze udaje mu się rozśmieszyć ludzi wokoło! W wolnym czasie lubi grać w baseball z kolegami z teamu.

# Grzegorz Prokopczyk

Dwa lata temu skończył studiować informatykę. Obecnie studiuje fortepian jazzowy na poznańskiej Akademii Muzycznej i pracuję w firmie F-Secure jako programista.

Oprócz informatyki i muzyki interesuję się książkami, głównie fantasy i science-fiction, lubię grać w tenisa i jeździć rowerem.

Lubi przekazywać swoją wiedzę innym, uważa że udział w warsztatach może być ciekawym i rozwijającym doświadczeniem - I tego Ci właśnie Grzegorz życzymy!



# Sabina Czopik

Sabina jest żywym dowodem na to, że takie pomysły jak PyLadies działają i mogą zmienić życie. Rok temu była Specjalistą ds. Marketingu, a aktualnie jako Test Engineer uczy się Pythona i zautomatyzowanych testów ..  
A w przyszłości? Kto wie ... Lubi piec chleb, gotowanie, podróżowanie i fitness

# Pablo Rodriguez

Pablo to Python Developer z rocznym doświadczeniem. Swoją przygodę z Pythonem rozpoczął kiedy pojawiła się potrzeba użycia tego języka programowania jako narzędzia dla badań doktoranckich związanych z bioinżynierią.

Pablo przyjechał do Polski z Hiszpanii już 6 lat temu - świetnie posługuje się nie tylko swoim ojczystym językiem, ale także angielskim i polskim.

Po godzinach z programisty przemienia się w muzyka - gra na gitarze, śpiewa i komponuje w zespole rockowym Tortuga.



# Justyna Kuziem ska

Od października studentka 4 roku bioinformatyki. Z kodowaniem zetknęła się dopiero na studiach, zaczynając od C++, którego porzuciła po 1,5 roku na rzecz Pythona. W przyszłości chciałaby zająć się analizą danych.

W wolnych chwilach uwielbia jeździć rowerem i czytać książki.

Jej supermoc - zdolności analityczne i upór (przydają się przy szukaniu błędów w kodzie)



# Krzysztof Marciak

Software Engineer w F-Secure zainteresowany tematyką bezpieczeństwa .. Przygodę z programowaniem zaczynał od C i C++, przechodząc przez PHP (ugh), Javę i BASHa, na Erlangu, Elixirze czy Scali (i Pythonie) kończąc.

Pythona używa głównie do jednorazowej automatyzacji zadań i tworzenia narzędzi przydatnych później w pracy (choć nie tylko). Absolwent specjalności Sieci Komputerowe i Systemy Rozproszone na Politechnice Poznańskiej.

Po godzinach gra na saksofonie altowym ..

Swoją supermoc określa jako szeroką wiedzę, bo jej nigdy nie za wiele!



# Tomasz Magulski

Gdy Tomasz opisuje siebie, od razu zaznacza, że nie jest idealny. Kiedy już się na czymś skupi, nie ma szans, żeby coś go zatrzymało w dążeniu do celu. Z entuzjazmem korzysta z Pythona i JavaScriptu. Jest znany ze swojego wkładu w poczytne książki na temat JS-a. Nie bez przyczyny jego pseudonim to "Semicolon Master".

Najnowsza obsesja Tomka? Założenie i prowadzenie ruchu [Code for Poznań](#)

# Bartosz Płociennik

Na co dzień Python Developer (a jakże!) skupiony na projektach aplikacji webowych, które często tworzy z wykorzystaniem Django. Jak nie znajduje się przy klawiaturze uwielbia sporty ekstremalne . w szczególności wspinaczkę, ale też dobre piwo , serial czy wielogodzinny trekking w Tatrach . (oczywiście jak tłumów nie ma!)

Jego super moc - potrafi słuchać AC/DC pisząc przy tym kod i usiedzieć na miejscu

# Martyna Urbanek

Z wykształcenia biotechnolog i informatyk, z zawodu biolog molekularny i analityk danych, z pasji nauczyciel (Pythona dla dorosłych i Scratcha dla dzieci w @Kids Code Fun).

Pythona wykorzystuje głównie do analizy i wizualizacji danych, w tworzeniu prostych aplikacji bioinformatycznych i ułatwianiu sobie codziennej pracy.

Jej super mocą jest dobra organizacja czasu bez niej nie mogłaby robić tyle co robi

# Miłosz Kusiciel

Pierwsze kroki w software developmencie Miłosz stawał jako PHP developer. Szybko jednak porzucił tę ścieżkę dla Pythona .. Ze swoim 8 letnim doświadczeniem, Miłosz jest nie tylko Django Developerem, ale także Team Leaderem zespołu Django w Merixstudio.

W swojej codziennej pracy skupia się na zarządzaniu pracą zespołu i poszukiwaniu usprawnień oraz nowych możliwości. Przed wszystkim jednak zajmuje się pisaniem aplikacji.

Jego zainteresowania to nowe technologie, procesy automatyzacji oraz branża startupowa.

Jego supermoc? Odpowiedź jest prosta - pasja do programowania, to od razu widać!



# Jacek Stachowiak

Test Automation Engineer i QA w Komputronik.pl. Na codzień zajmuje się pisaniem, utrzymaniem i rozwojem testów automatycznych pisanych w Pythonie (framework do testów - Selenium Web Driver). Z wykształcenia biolog - genetyk ..

Pasjonuje się motoryzacją i sportami motorowymi oraz militariami. Ponadto uwielbia czytać książki - głównie fantastyczne i sci-fi, gotuje. Lubi wiedzieć co jak dokładnie działa i jak to rozłożyć na czynniki pierwsze.

Supermoc - potrafi opowiadać o swoich pasjach (i nie tylko) godzinami

# Maciej Zięba

Maciej swoją przygodę z programowaniem zaczął całe wieki temu, bo w 1991 roku, kiedy okazało się, że jego wspaniałe 8-bitowe Commodore 64 potrafi coś więcej niż tylko uruchamiać gry .. Swoją wiedzę i umiejętności udało mu się poszerzyć dzięki studiom na Politechnice Poznańskiej.

Pythona używa już od 11 lat, przechodząc w tym czasie przez najróżniejsze frameworki (Plone, Pylons, Flask, CherryPy, Pyramid, Django . ). Na co dzień zajmuje się tworzeniem aplikacji webowych, gdzie oprócz Pythona, często wykorzystywany jest też Javascript.

Aby zupełnie nie przyrosnąć do krzesła, lubi uprawiać sport, zwłaszcza bieganie..

Jego supermoc to zatem duuuuuże doświadczenie, które będzie bezcenne dla naszych uczestników!

# Marek Pilarczyk

Marek jest doświadczonym Senior Fullstack Python Developerem. Swoją ciężko zdobytą wiedzę wykorzystuje z powodzeniem na co dzień i z satysfakcją dzieli się nią zarówno z zespołem, jak i osobami z zewnątrz. Po pracy Marek uczęszcza na siłownię i zbiera różne rzeczy, takie jak na przykład naklejki z samochodami po gumach Turbo. - można powiedzieć, że to jego hobby.

Jego supermoc to cierpliwość, co czyni go naprawdę wprawnym nauczycielem i mentorem.

# Jakub Senik

Programowaniem pasjonuje się od 5 lat, od 4 programuje w Pythonie. Pracuję jak full stack developer w STX Next

# Kamil Pazik

Developer Pythona, w wolnym czasie uprawia sporty, żegluje i jeździ motocyklem.

# Hubert Dworczyński

Na co dzień programista JavaScript w STX Next, ale zdarza mu się pisać przeróżne skrypty w Pythonie po to, aby ułatwić sobie życie. Interesuje się też muzyką, grafiką 3D i tworzeniem aplikacji VR.

# Marcin Kowiel

F-Secure



# Zuza Kunik

Na co dzień specjalizująca się w przetwarzaniu obrazów i sztucznej inteligencji studując na specjalności Systemy Wizyjne (Automatyka i Robotyka) na Politechnice Poznańskiej.

Jej miłość do Pythona zaczęła się od pracy inżynierskiej, częściowo napisanej właśnie w tym języku. Uczucie to udało jej się rozwinąć w poprzedniej, trzeciej edycji PyLadies na poziomie zaawansowanym. Dzięki temu dostała się na staż, który przerodził się w jej pierwszą programistyczną pracę w F-Secure!

W wolnym czasie pochłania książki w całości, uczy się ukraińskiego , lub siedzi w górach.

Jej super moc to najcieplejszy uśmiech na świecie, którym umie się podzielić z każdym.



# Monika Jankowiak

Monia zaczęła programować w Pythonie 2 lata temu i najczęściej wykorzystuje go do pisania krótkich skryptów, aby przyspieszać sobie żmudną pracę biurową - i oszczędzać w ten sposób czas na inne przyjemności.

Na PyLadies.start() debiutuje jako mentor, ale już od 1,5 roku jest uczestniczką, twarzą i koordynatorką Pyladies Poznań. Trzyma nas w kupie i sprawia, że warsztaty odbywają się systematycznie i w fajnej atmosferze!

Lubi takie nudy jak książka, koty , rower, kino i teatr. Ma dwóch całkiem już dorosłych synów.

Jej supermoc? - Buduje reputację matki-gamerki w takich grach online jak WoW, HoTS, czy Hearthstone

# Piotr Dyba

Piotr to przede wszystkim osoba, która od początku wspiera Pyladies, teraz jest jednym z głównych organizatorów poznańskiego ruchu, co ważne, jest pomysłodawcą prawdopobnie największego jak dotąd w Polsce warsztatu z Pythona, czyli Pyladies.start().

Kilka lat temu to właśnie on sam zasiadał na miejscu uczestnika i teraz stara się oddać to, co otrzymał swoim zaangażowaniem.

Aktualnie jest Team Leaderem w firmie F-Secure, zajmującej się cyberbezpieczeństwem. Wraz z teamem zajmuje się rozwojem sieci Honeypotów (pułapek na cyberprzestępco), sensorem sieciowym i różnymi zadaniami specjalnymi.

Po pracy Piotr swój czas poświęca Pyladies, poza tym odwiedza różne branżowe konferencje i występuje jako prelegent na np. EuroPythonie, PyConPL, PyConCZ oraz PyCon Japan.

Tak zupełnie już prywatnie Pastafarianin, człowiek gór zakochany w Japonii.

Jego super moc? Jest ich kilka, ale ważniejsze z nich to zaangażowanie i upartość w dążeniu do obranych celów!

# Role



# Ściągawka

<http://dyba.it/py.pdf>

Co to jest programowanie,  
Python, i do czego można  
go użyć

# Python: czym nie jest?

To nie jest aplikacja !

To nie jest program !

To nie jest framework !



python

# Python: pyhistria ?



Guido van Rossum,  
twórca Pythona  
1991

# Python - używany m.in.:

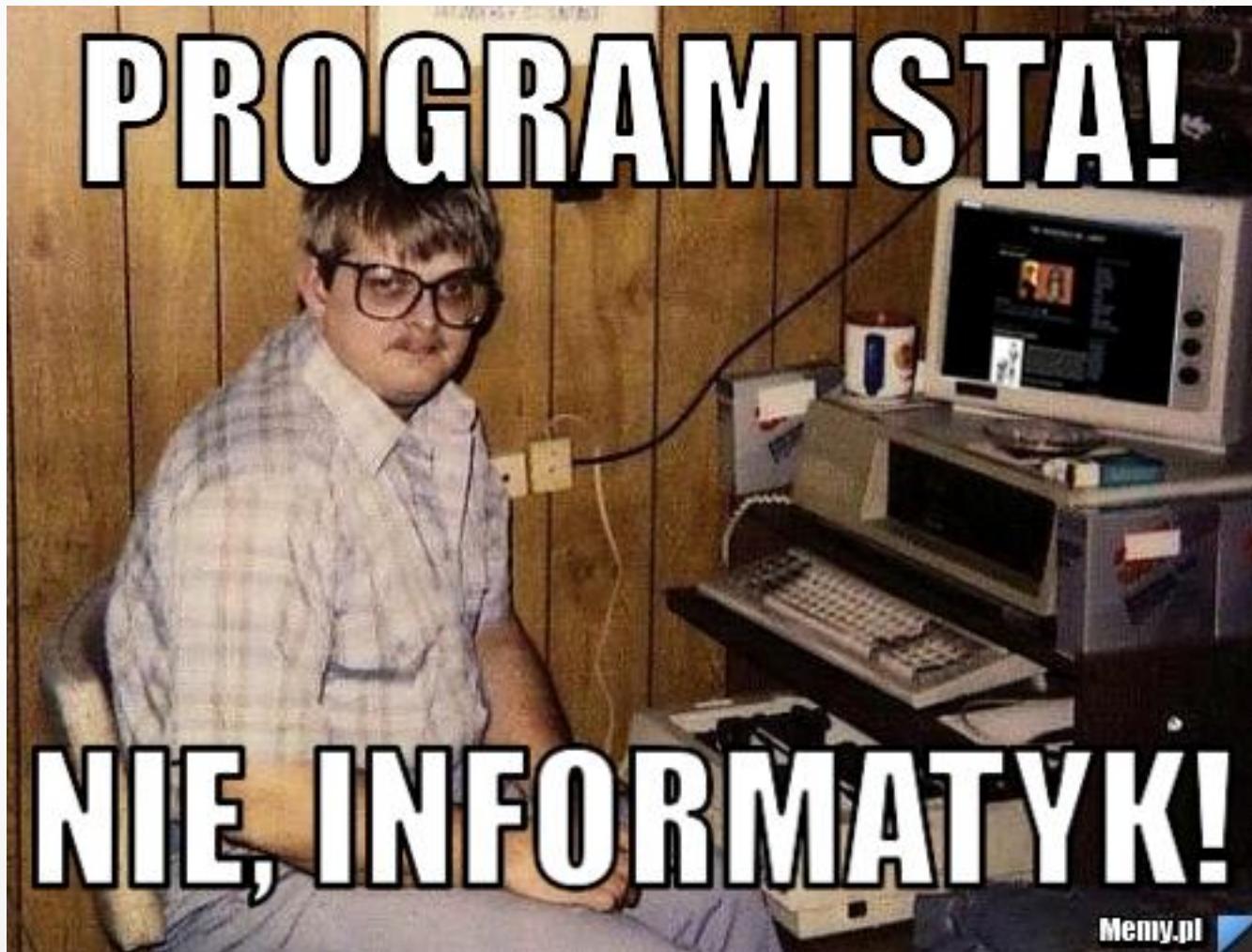
- Apture
- Disqus
- Dropbox
- Google
- Hunch



# Python: kompendium

<https://python.zeef.com/>

# Programista: Kto to jest ?



# Programista: Kto to jest ?

- Wynalazca
  - Designer
  - Górnik
  - Detektyw
  - Matematyk
  - Anglista
- 
- Ewolucja: Architekt
  - Ewolucja: Manager

# Programista: Jak zostać?

- Angielski, angielski, angielski.
- Umiejętność kodowania w danym języku.
- Profil na Linkedin, Stackoverflow, Github.
- Umiejętności miękkie, szczególnie komunikatywność.

Co może pomóc:

- Znajomość Linuksa/Basha.
- Szczerość i bezpośredniość.

# Programista: Jak zostać?

Ciążka praca nad  
kodem

# Programista w pracy: zagadki

„Dlaczego ten kod  
działa”?

„Jakim cudem nie  
działa?!”

O co chodzi

# Programista w pracy: Warunki

Praca zdalna

Elastyczne (bardzo) godziny pracy

Zadaniowy czas pracy

Integracje

Okresy wypowiedzenia wg. pracownika

\$\$\$

kozy

# Programista w pracy: Warunki

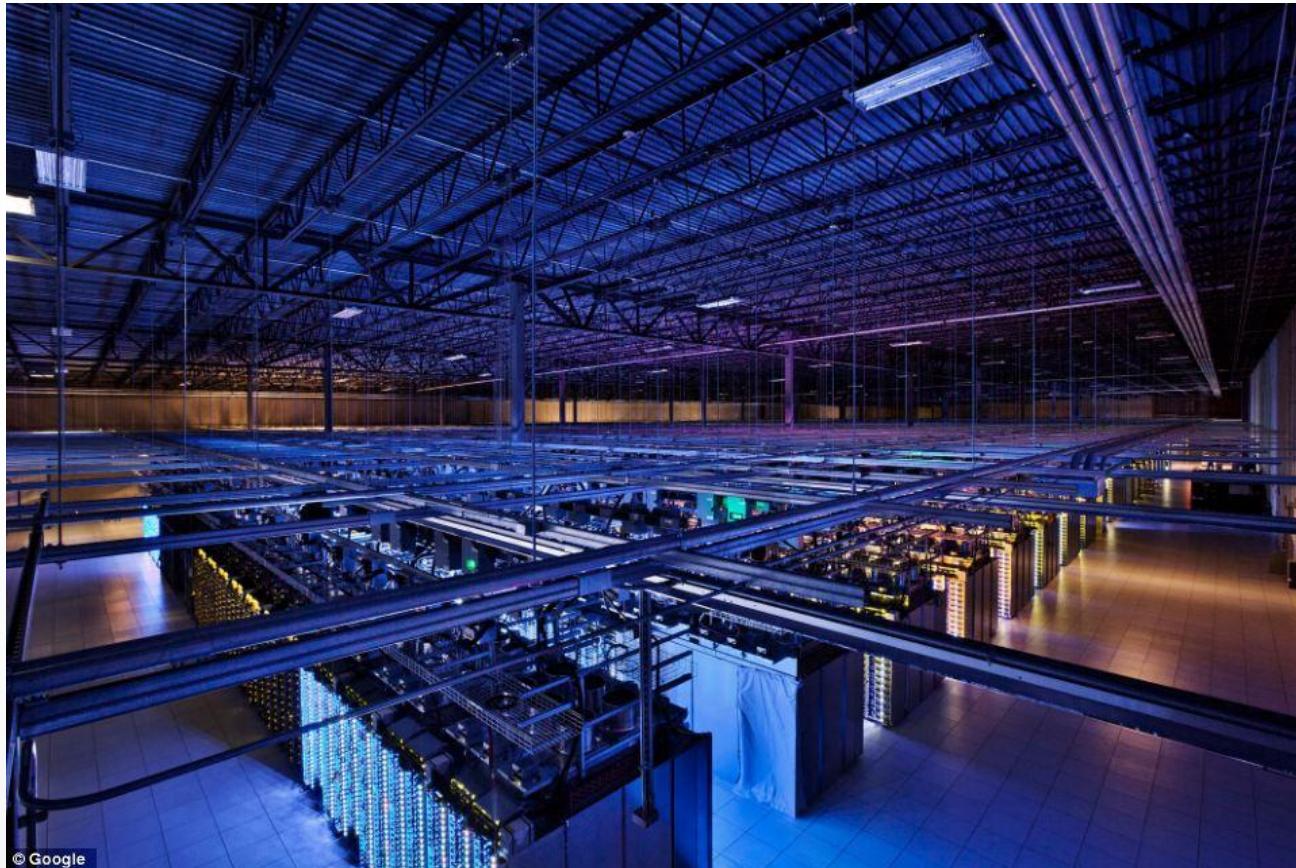


BYŁEM BEZROBOTNYM PROGRAMISTĄ!

PRZEZ 30 MINUT!

MANAGE IT - PRAKTYCZNIE O KARIERZE W IT  
[HTTP://BJD.PL](http://bjd.pl)

# Programista w pracy: Przyszłość



# Przyjazne wprowadzenie do algorytmiki

# Algorytm: co to ?

**Skończony** ciąg jasno zdefiniowanych czynności, koniecznych do wykonania pewnego zadania.

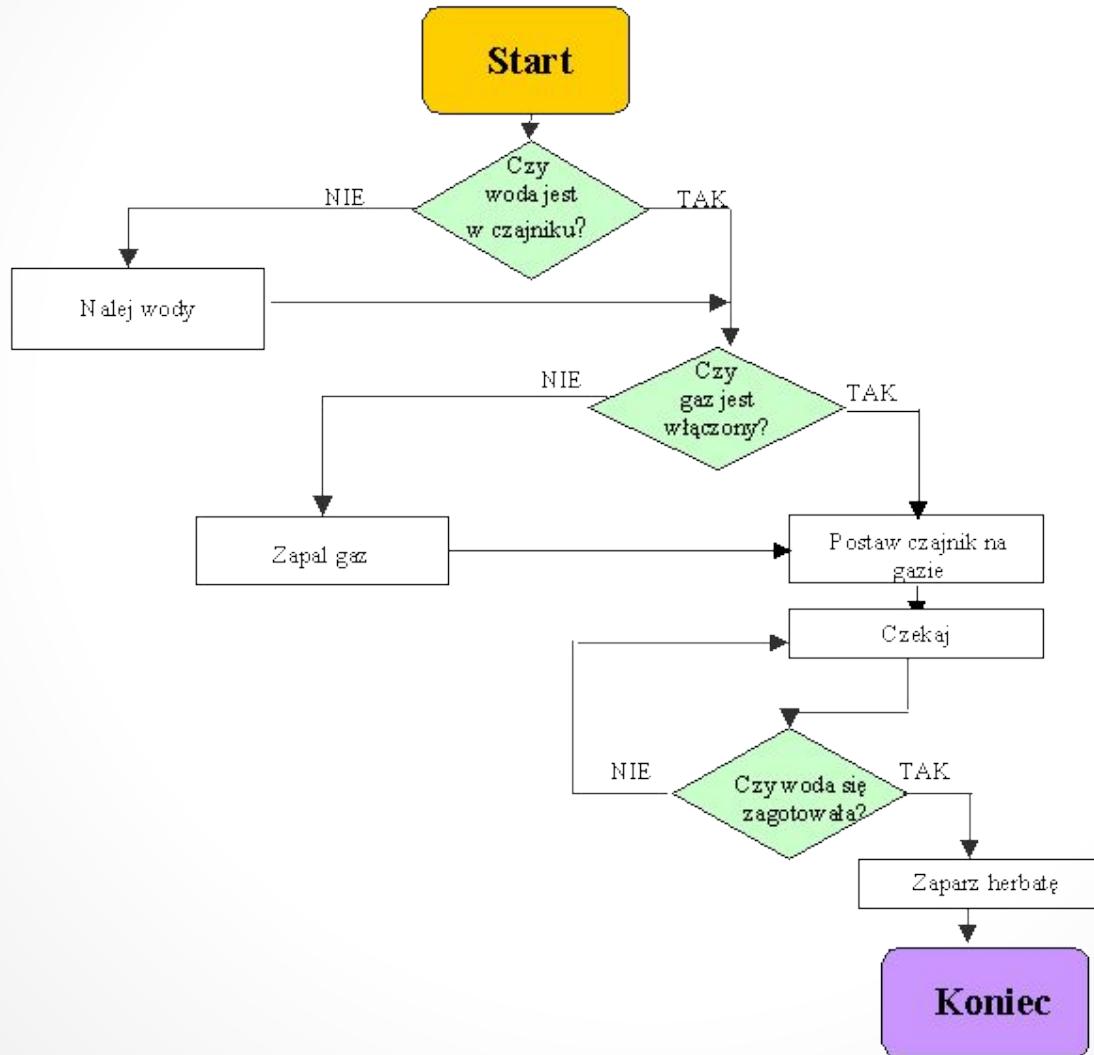
Mожет быть он представлен мин.::

- Słownie
- Graficznie np. schemat blokowy
  - Pseudokod
  - Kod w j. programowania

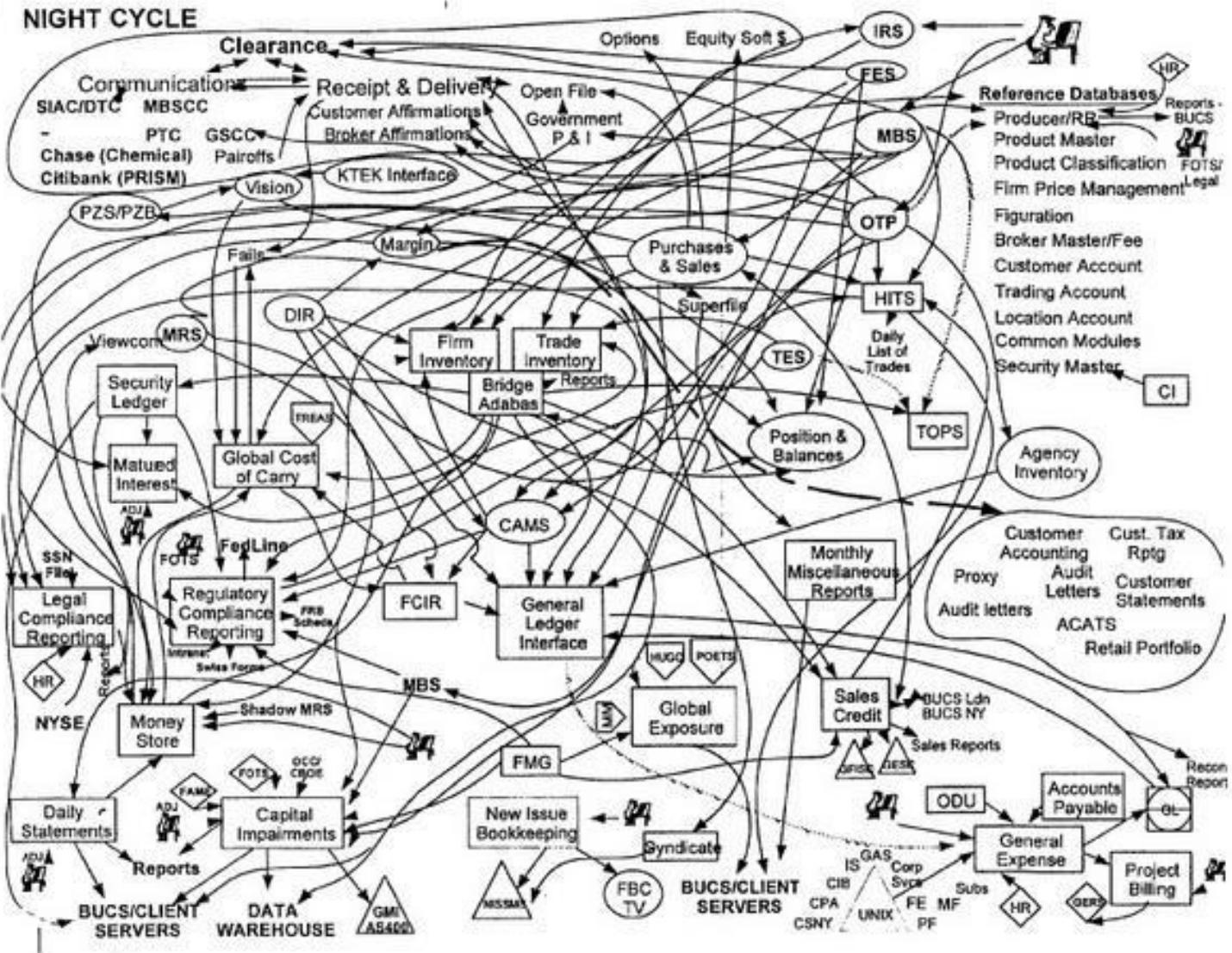
# Algorytm: co to ?



# Algorytm: co to ?

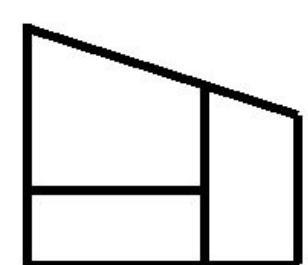
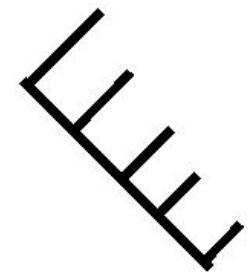
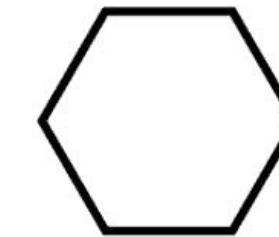
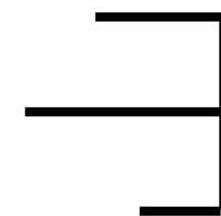
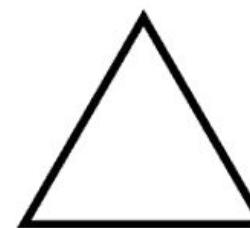
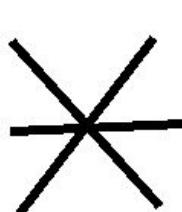


# Algorytm: co to ?



# Algorytm: Ćwiczenie 1/3

Każdy rysuje na 1 kartce figurę składającą się z maksymalnie 6 kresek np.  
Podpisz ją swoim numerem.



# Algorytm: Ćwiczenie 2/3

Kartkę z Obrazkiem osobie po lewej stronie. Osoba, która otrzymała kartkę piszę algorytm jak narysować daną figurę na **NOWEJ** kartce !

Przepisz numer uczestnika od którego dostałeś kartkę.

6min.

# Algorytm: Ćwiczenie 3a

Osoba, która ma kartkę z obrazkiem składa ją na pół, tak aby narysowana na niej figura była niewidoczna.

I podaj ją z powrotem autorowi.

30s

# Algorytm: Ćwiczenie 3b/3

Przekazujemy tylko kartkę z instrukcją 3 osoby w lewo.

Po otrzymaniu dużej kartki staramy się narysować figurę zgodnie z algorytmem.

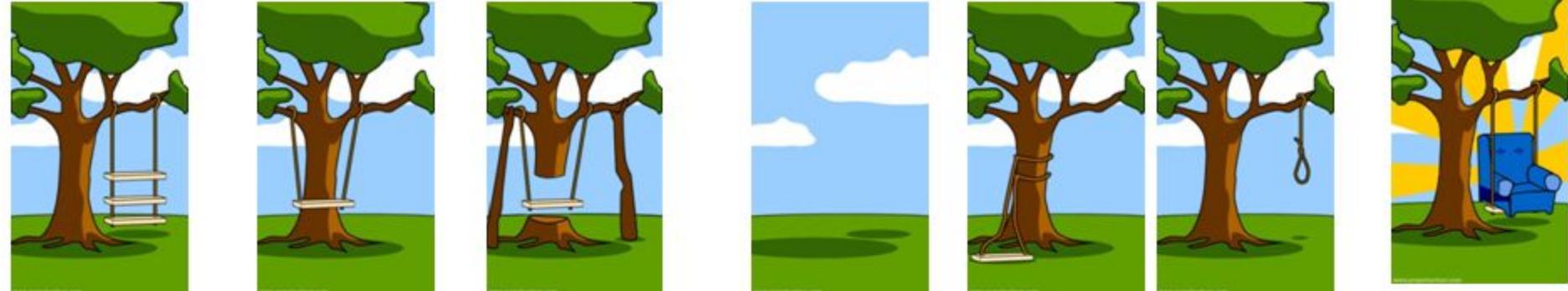
Podpisujemy kartkę numerem z instrukcji

# Algorytm: Ćwiczenie 3c/3

Osoba, która ma obie kartki zwraca je obie do autora.

Każdy porównuje wyniki z tym co sami narysowali.

# Algorytm



Jak wyjaśnić  
**Klient**

Jak zrozumiał  
**Kierownik Projektu**

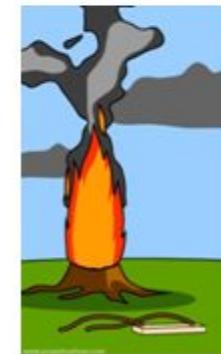
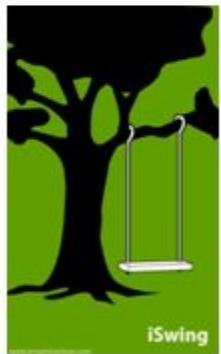
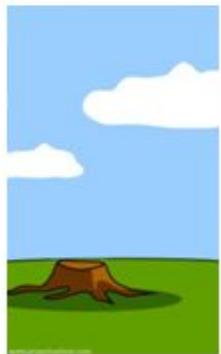
Jak to zanalizował  
**Analityk**

Jaka była  
**dokumentacja**

Jak napisał  
**Programista**

Co otrzymał  
**Tester**

Jak to przedstawił  
**Konsultant**



Za co zapłacił  
**Klient**

Kiedy zostało  
**wydane**

Jakie było  
**wsparcie**

Co przyjął do  
reklamacji  
**Dział  
Handlowy**

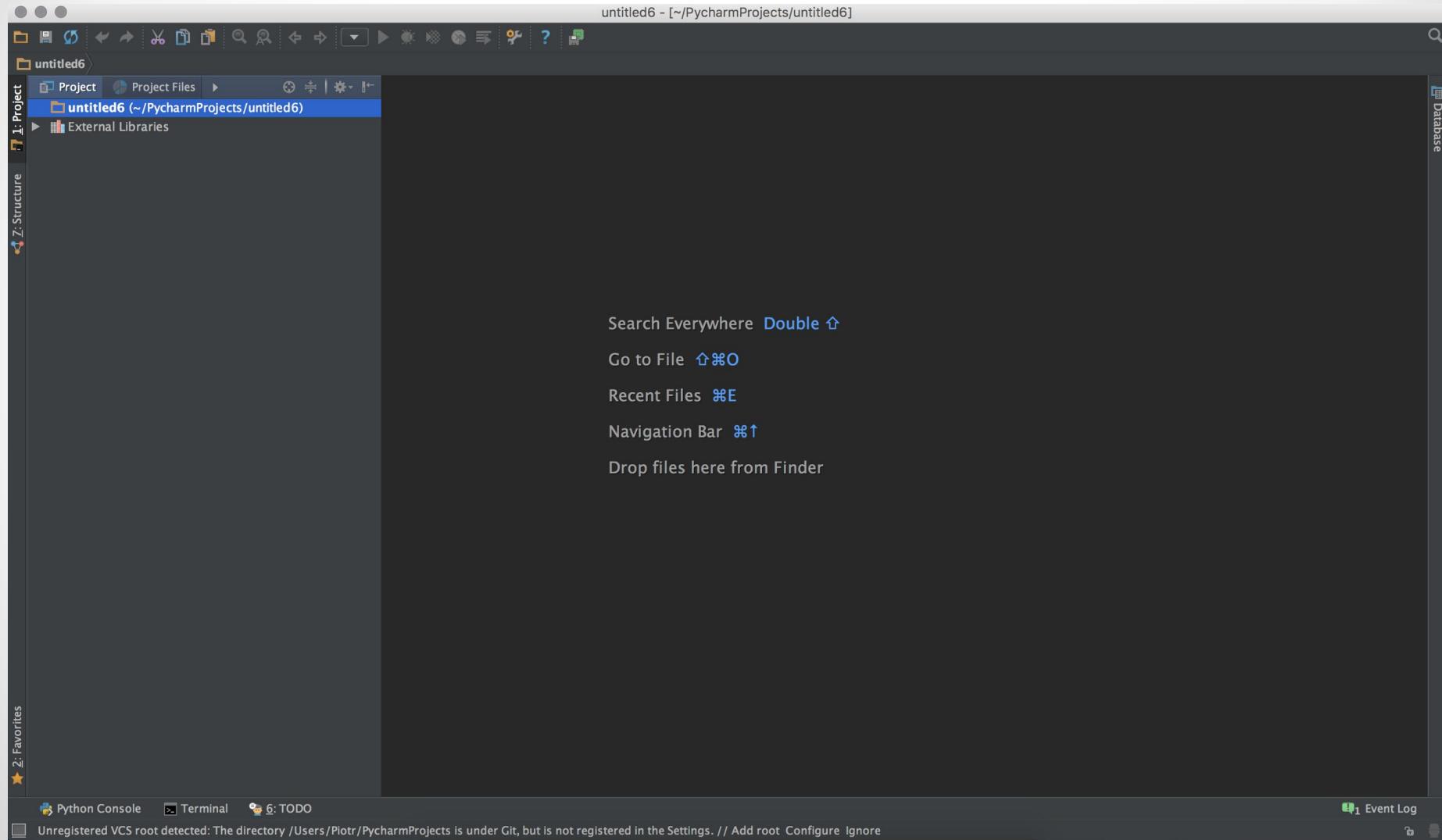
Czego **Klient**  
potrzebował

Jakie było  
**zabezpieczenie**

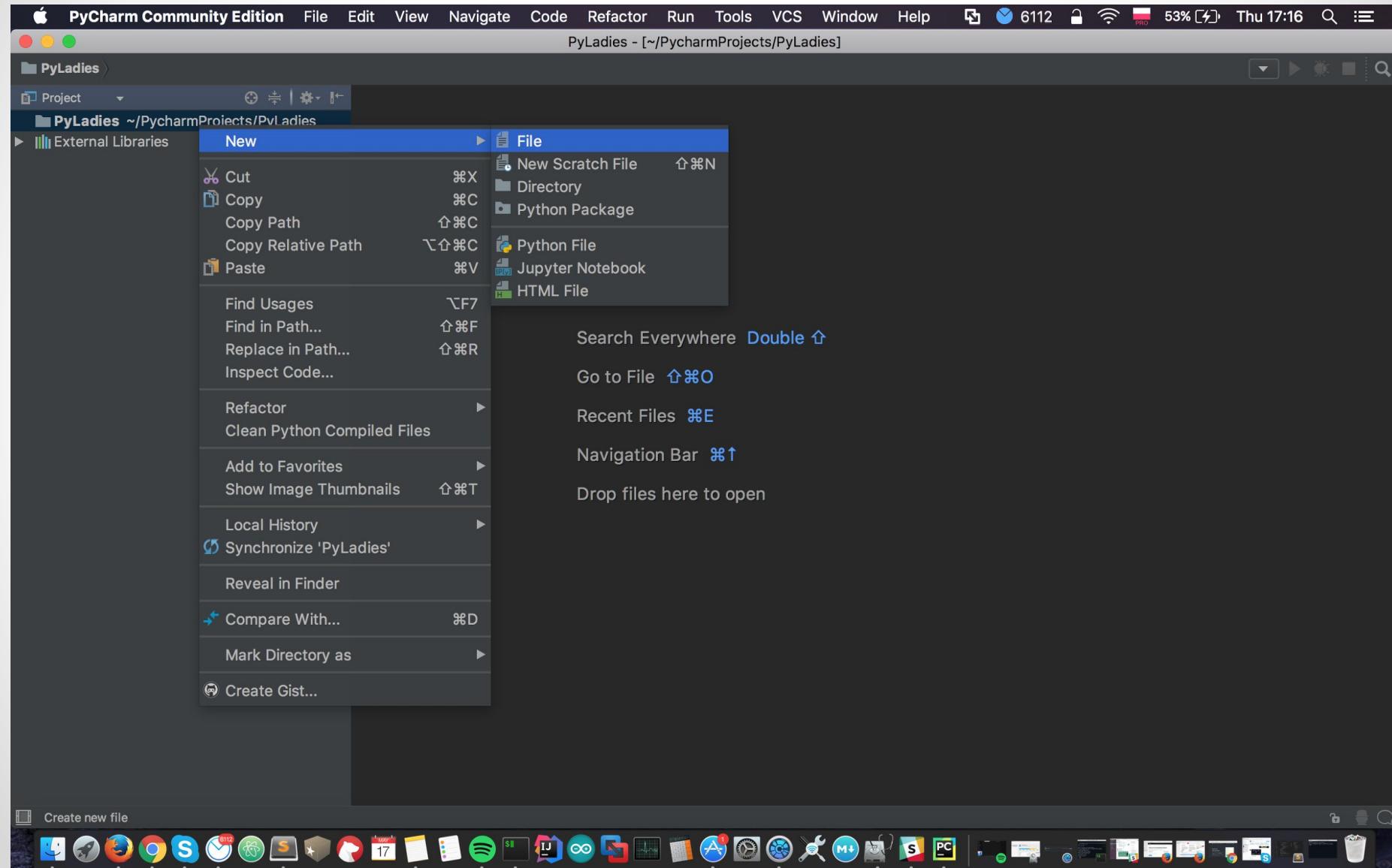
Jaki był plan  
**odbudowy**

# Pierwsze kroki z interpreterem Pythona i IDE Pycharma

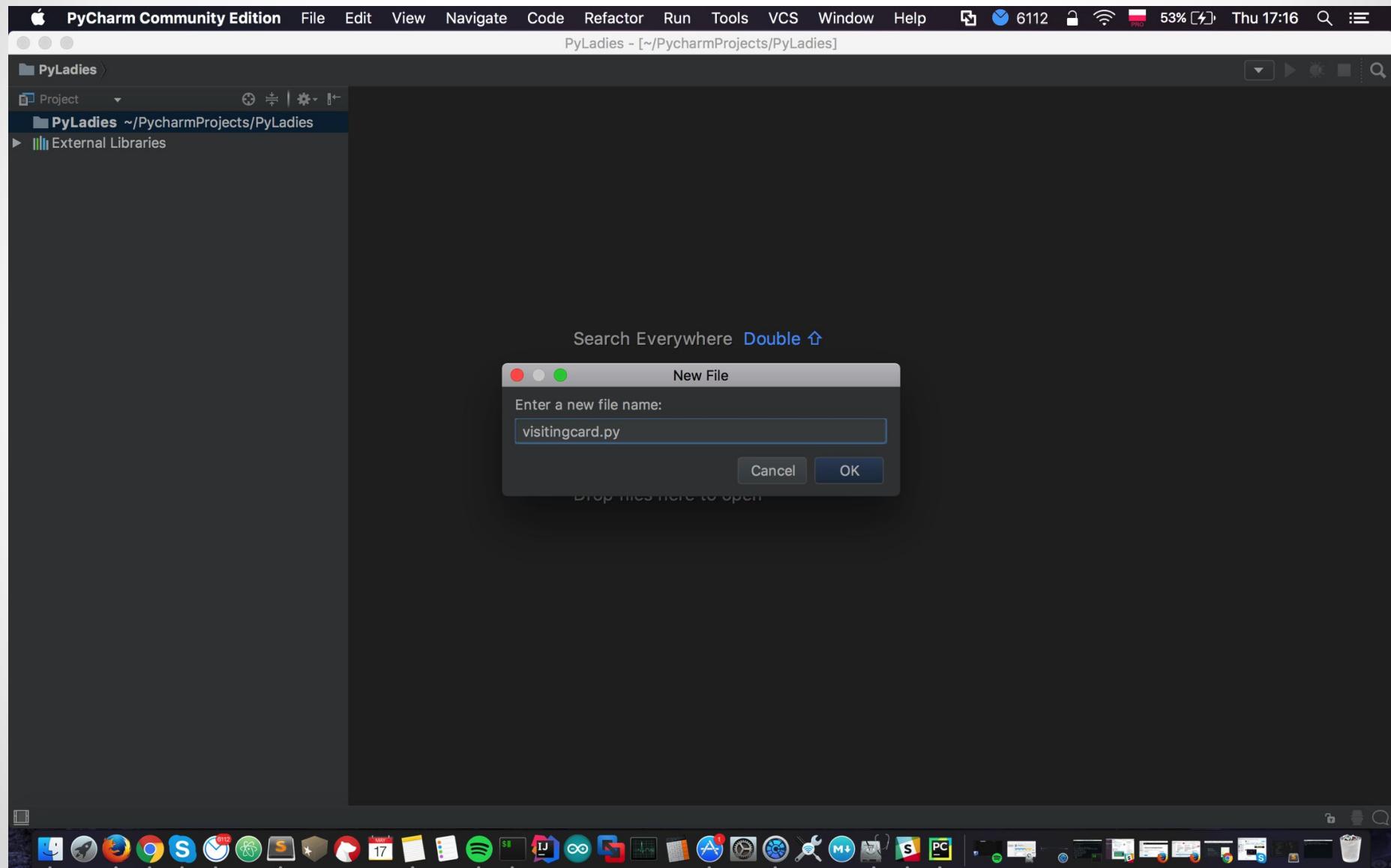
# PyCharm



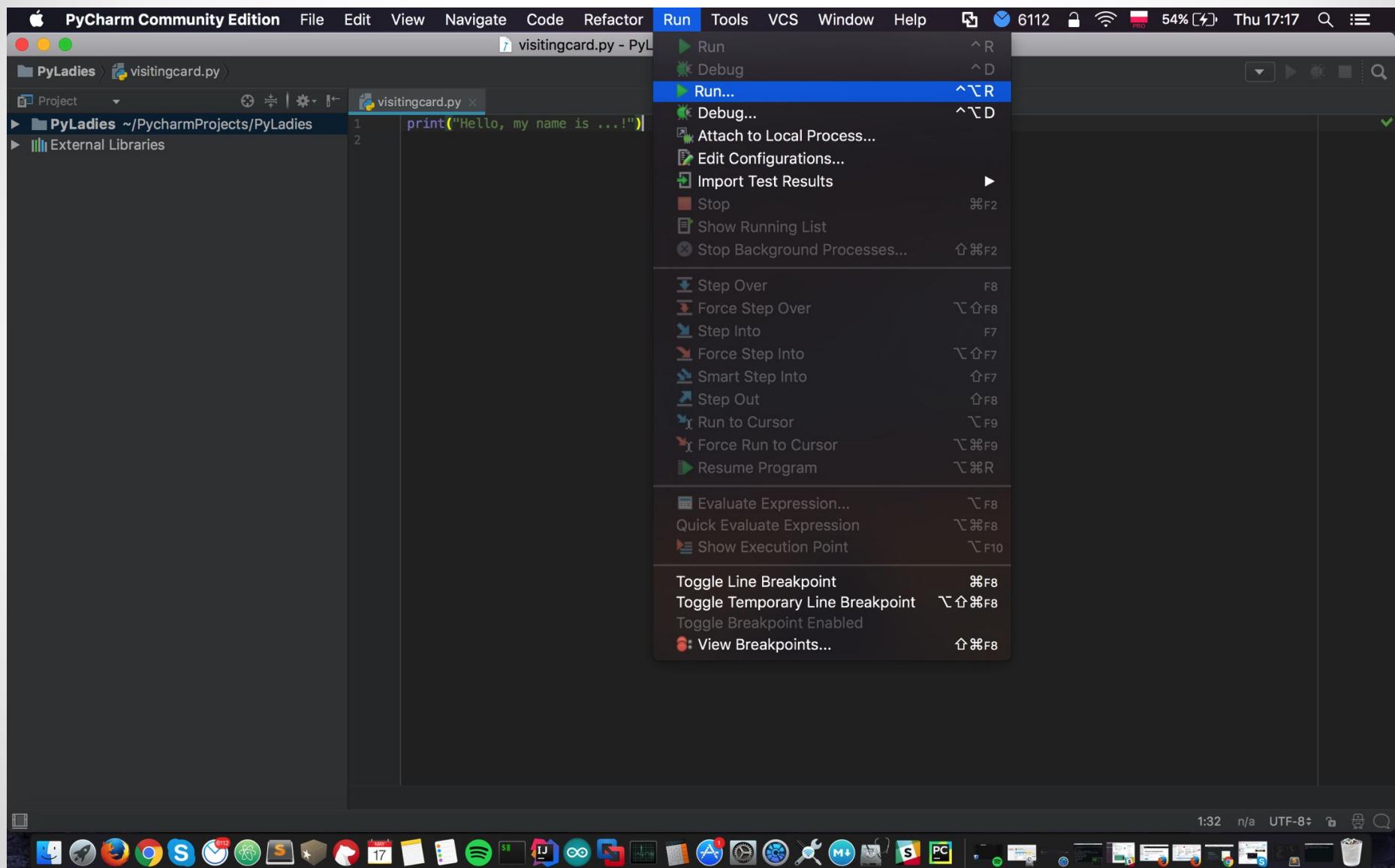
# PyCharm



# PyCharm



# PyCharm



# PyCharm

The screenshot shows the PyCharm Community Edition interface on a Mac OS X desktop. The window title is "visitingcard.py - PyLadies - [~/PycharmProjects/PyLadies]". The code editor contains the following Python code:

```
1 print("Hello, my name is ...!")
```

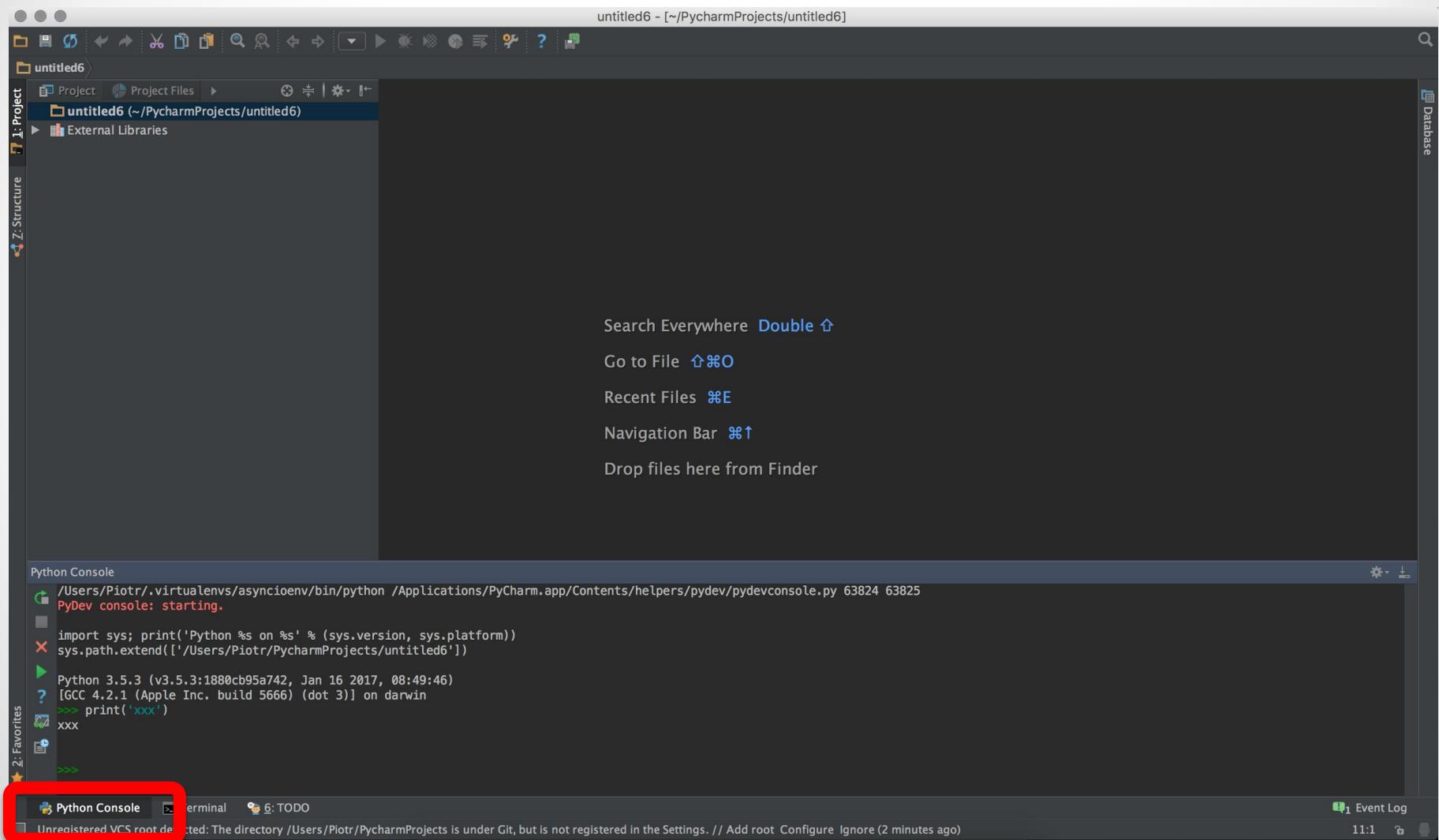
In the bottom-left corner of the code editor, there is a small icon of a person with a blue circle on their head.

The "Run" tool window at the bottom shows the output of running the script:

```
/usr/local/Cellar/python3/3.6.1/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/kris_m/PycharmProjects/PyLadies/visitingcard.py
Hello, my name is ...!
Process finished with exit code 0
```

The status bar at the bottom right indicates the time is 1:32, the battery level is n/a, the encoding is UTF-8, and the system is connected to a Wi-Fi network.

# PyCharm



# Interpreter Pythona

```
pdyba:~ Piotr$ python3
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('test')
test
>>>
```

# Interpreter Pythona

```
pdyba:nauka Piotr$ python3 prosty_print.py
Taki tam printx
pdyba:nauka Piotr$ █
```

# Interpreter Pythona

```
prosty_print.py *  
1 print('Taki tam printx')  
2 print('Dodatkowy print po savie')  
  
nauka — -bash — 80×24  
pdyba:nauka Piotr$ python3 prosty_print.py  
Taki tam printx  
pdyba:nauka Piotr$ python3 prosty_print.py  
Taki tam printx  
Dodatkowy print po savie  
pdyba:nauka Piotr$
```

# PyCharm

Python – konsola interaktywna	PyCharm - IDE
Kod wykonywany od razu	Kod wykonywany w całości po kliknięciu Run
Brak kolorowania składni	Kolorowanie składni
Brak historii po zamknięciu sesji	Wiele plików z różnym kodem, działający ctrl+z ;)
	Wiele udogodnień, które się będzie później pojawiać ☺

# PyCharm

- Wykonując zadania starajcie się aby każde zadanie robić w nowym pliku tak żeby mentor mógł sprawnie wam pomóc.

>>> - oznacza że kod był wpisywany w konsolę interaktywną.

Klikajcie, startujcie ile razy  
chcecie żadne polecenie  
na tych warsztatach nie  
ma prawa wam uszkodzić  
komputera !

# Python jako Kalkulator

# funkcja

Funkcja – zbiór operacji.  
Zazwyczaj składa się z kilku czynności wykonywanych przez komputer.

# funkcja

Tworzenie omówimy później.

Funkcję wywołujemy poprzez podanie jej nazwy i dodanie dwóch nawiasów okrągłych, w których **możemy** przekazać parametry.

# print

Funkcja wypisująca dane na ekran.

print()

print('abc')

print('hallo world')

print(123)

# print

Zadanie:

Napisać print'a ( print() ), który wypisze na ekranie wasz wiek:

# print

Odpowiedz:

print(99)

# Matematyka

+ - dodawania

- - odejmowanie

\* - mnożenie

/ - dzielenie

\*\* - potęgowanie

// - dzielenie do części całych zaokrąglając zawsze w dół

% - modulo czyli reszta z dzielenia

# Matematyka – przykłady:

```
>>> 10 + 2
```

```
12
```

```
>>> 10 - 2
```

```
8
```

```
>>> 10 * 2
```

```
20
```

```
>>> 10 / 2
```

```
5
```

```
>>> 10 ** 2
```

```
100
```

```
>>> 10 // 3
```

# Matematyka – przykłady:

```
>>> 10 + 0.1
```

```
10.1
```

```
>>> 10 - 0.001
```

```
9.999
```

```
>>> 10 * 1.23
```

```
12.3
```

```
>>> 10 / 27
```

```
0
```

```
>>> 10 / 0.1
```

```
100.0
```

```
>>> 10 / 9
```

Jak wytłumaczyć wężowi  
o czym mówimy czyli -  
typy danych: znakowy,  
liczbowy i boolowski

# int i float

Typy danych odpowiadające za przechowywanie liczb:

**int** – całkowitych np. 1, 1234, -1231

**float** – ułamki dziesiętne

(zmiennoprzecinkowe) np. 1,01;  
123,999; 0,01

# Matematyka – zadania:

Dla każdego z operatorów + - \* / \*\* // %  
wykonaj po jednym działaniu na zestawieniu:

- a) Dwóch intów np. 9 + 14
- b) Inta i float np. 9 + 14.001
- c) Dwóch floatów np. 9.1 + 1.4001

# String(i)

Typ danych do przechowywania ciągu znaków.

Typ tekstowy.

'Ala ma kota'

"Kot ma Ale"

'123'

"999"

'...' == "..."

'...' == "..." == """..."""" == """..."""

# String(i)

Czasami potrzebujemy apostrof lub cudzysłów w środku np.

“I’m famous”

‘Widziałam “władcę pierdzieli” ! i był super’

Ale co zrobić jeśli potrzebujemy obu znaków ?

Magiczny znak \

```
>>> print('\\')
```

,

```
>>> print('\\') – wypisanie magicznego \
```

# Matematyka Stringów

Udogodnienia:

```
>>> "ala" + "ma" + "kota"
```

```
>>> "ma" * 2
```

Niestety nie działają:

```
>>> "ma" + 2
```

```
>>> "ma" * "ala"
```

# Stringi zadanie.

Sprawdź co wypiszę linijka:  
`print('Na' * 20 + ' Batman')`

Napisz tak stringa żeby wypisał:

- AaAaAaAaAaAaAaAaAaAaAaAa
- Nowy film Ama'deusza pt. „xyz”
- A\A' == A\A”

# Stringi zadanie - odp

AaAaAaAaAaAaAaAaAaAaAaAa

```
print('Aa' * 12)
```

Nowy film Ama'deusza pt. „xyz”

```
print(„Nowy film Ama'deusza pt. \„xyz\”")
```

A\A' == A\A"

```
print(„A\\A' == A\\A\”")
```

# input

Funkcja służąca do pobrania danych od użytkownika: np.:

```
>>> input()
```

```
'test'
```

```
'test'
```

```
>>> input('podaj rozmiar swojego buta: ')
```

```
podaj rozmiar swojego buta: 2
```

```
2
```

Funkcja po pobraniu danych wypisuje je.

# input

Input połączony z printem:

```
>>> print('Lubie jesc: ' + input('co lubisz jesc? '))  
co lubisz jesc? 'banany'  
Lubie jesc: banany
```

```
>>> print(input('ile liter ma twoje imie? ') * 5)  
ile liter ma twoje imie? 5  
55555  
Bo 5 * 5 to nie to samo co 5 * '5'
```

# Input ćwiczenie

Korzystając z printa i inputa napisz funkcję, która pobierze i policzy BMI lub pole prostokąta:

BMI waga podzielona przez kwadrat wzrostu

$$\text{bmi} = m / h \cdot h$$

Pole prostokąta bok \* bok

$$p = a \cdot b$$

# Input ćwiczenie – odp.

```
print('Twoje BMI wynosi: ' + input('Podaj wage: ') /  
      input('Podaj wzrost: ')**2)
```

```
print('Pole prostokat: ' + input('Bok A: ') *  
      input('Bok B: '))
```

I mamy błąd ☺ Stringa nie można od razu połączyć z floatem.

# Zmienianie typów danych

`int()`

`int('12')`      `int(12.901)`

`float()`

`float('12')`      `float('123.1')`      `float(12)`

`str()`

`str(12)`      `str(0.001)`

# Zmiana typu danych ćwiczenie

1. Usuń błąd z poprzedniego zadania:

```
print('Twoje BMI wynosi: ' + input('Podaj wage: ') /  
      input('Podaj wzrost: ')**2)
```

```
print('Pole prostokat: ' + input('Bok A: ') *  
      input('Bok B: '))
```

2. Zamień string '123.12' na int.

# Zmiana typu danych odpowidzi

1. Usuń błąd z poprzedniego zadania:

```
print('Twoje BMI wynosi: ' + str(input('Podaj wage:')) / input('Podaj wzrost: ')**2))
```

```
print('Twoje BMI wynosi: ' + str(int(input('Podaj wage: '))) / float(input('Podaj wzrost: '))**2))
```

```
print('Pole prostkoat: ' + str(input('Bok A: ') * input('Bok B: ')))
```

```
print('Pole prostokat: ' + str(float(input('Bok A: ')) * float(input('Bok B: '))))
```

2. Zamień string '123.12' na int.  
int(float('123.12'))

# Bool

Typ logiczny może przyjmować tylko jeden z dwóch stanów:

Prawda – True, 1

Fałsz – False, 0

# Zmienna niczym kobieta

# Zmienna

konstrukcja programistyczna posiadająca atrybuty:

1. symboliczną nazwę
2. wartość
3. miejsce przechowywania\*

pozwalająca w kodzie odwoływać się przy pomocy nazwy do wartości (lub miejsca przechowywania)

# Zmienna

Do zmiennej w Pythonie możemy przypisać prawie wszystko np.:

- liczbę, string
- Strukturę danych
- Funkcję

Tworzymy zmienną w formie:

**nazwa =  
wartość/obiekt/etc**

# Zmienna

Liczba, string:

```
masa = 79
wzrost = 1.85
imie = 'Piotr'
nazwisko = 'Dyba'
dane = imie + nazwisko + str(masa) + 'kg' + str(wzrost) + 'm'
```

# Zmienna

Funkcja np.:

```
l = len
l('Ala ma kota')
i = input
i('podaj swój wiek')
bardzo_krotka_nazwa_dla_funkcji_abs = abs
bardzo_krotka_nazwa_dla_funkcji_abs(-13131)
```

# Zmienna

```
bok_tresc = 'Podaj dlugosc boku'  
bok_wynik = 'powierzchnia kwadratu wynosi: '  
|bok = float(input(bok_tresc))  
print(bok_wynik + str(bok**2))
```

# Zmienna

Przykład czego nie robić:

```
1 i = input
2 l = len
3 p = print
4 s = str
5 f = float
6 m = 'Podaj swoja mase: '
7 w = 'Podaj swój wzrost: '
8 o = 'Twoje BMI wynosi: '
9 p(o + s(f(i(m))/(f(i(w))**2)))
```

# Zmienność - zadanie

Wykorzystując zmienne wykonaj jedno z poniższych.

1. Pobierz dane od użytkownika o jego wzroście i masie i zwróć wyliczone BMI (masa / wzrost  $^{**} 2$ )
2. Oblicz objętość prostopadłościanu ( $bok\_a * bok\_b * wysokość$ ) na podstawie danych od użytkownika
3. \* Oblicz powierzchnię wszystkich boków, powierzchnię podstawy oraz objętość stożka.

# Zmienna – zadanie/odp

1. Pobierz dane od użytkownika o jego wzroście i masie i zwróć wyliczone BMI (masa / wzrost \*\* 2)

```
masa = float(input('Podaj swoja mase: '))
wzrost = float(input('Podaj swój wzrost: '))
bmi = str(masa / (wzrost ** 2))
print('Twoje BMI wynosi: ' + bmi)
```

# Zmienna - zadanie

2. Oblicz objętość prostopadłościanu ( $bok_a * bok_b * wysokość$ ) na podstawie danych od użytkownika

```
a = float(input('Bok A: '))
b = float(input('Bok B: '))
h = float(input('Wysokosc: '))
v = str(a * b * h)
print('Objetosc wynosi: ' + v)
```

# Zmienna - zadanie

3. \* Oblicz powierzchnię wszystkich boków, powierzchnię podstawy oraz objętość stożka.

```
r = float(input('promien: '))
h = float(input('Wysokosc: '))
ppodstawy = r ** 2 * 3.14
dlbokow = str(2 * 3.14 * r)
obj = str(ppodstawy * h / 3)
print('Powierzchnia podstawy wynosi: ' + str(ppodstawy))
print('Dlugosc bokow wynosi: ' + str(dlbokow))
print('Objetosc wynosi: ' + str(obj))
```

# CRUD

C – Create - Stwórz

R – Read - Odczytaj

U – Update - Zmodyfikuj

D – Delete - Usuń

# Zabawy ze stringami

# String: tuple i \*\*

```
txt = 'To jest {0} tekst o {1} i jest on {0}'.format('fajny', 'mnie')
print(txt)

x = ('fajny', 'mnie')
txt = 'To jest {0} tekst o {1} i jest on {0}'.format(*x)
print(txt)

txt = 'To jest {jaki} tekst o {o} i jest on {jaki}'.format(**{
    'jaki': 'fajny',
    'o': 'mnie'
})
print(txt)

x = {'jaki': 'fajny', 'o': 'mnie'}
txt = 'To jest {jaki} tekst o {o} i jest on {jaki}'.format(**x)
print(txt)
```

# String - zadanie

- Wykorzystując formatowanie stringów napisz zaproszenie urodzinowe, które będzie generowane na podstawie danych:  
imie, nazwisko, tytuł, przymiotnik, ulubiony smak.  
Przykład (czerwonym automatyczne wypełnienie):

Prof. Andrzej Ruda,

Najblzszy przyjacielu chciałbym Cię zaprosić na swoje urodziny na których będę serwował Twój ulubiony **kokosowy** tort.

Piotr

# String

- capitalize
- casefold
- center
- count
- encode
- endswith
- expandtabs
- find
- format
- format\_map
- index
- isalnum
- isalpha
- isdecimal
- isdigit
- isidentifier
- islower
- isnumeric
- isprintable

# String - join

```
>>> a = ['Ala', 'ma', 'kota']
>>> print(' '.join(a))
Ala ma kota
>>> print(' nie '.join(a))
Ala nie ma nie kota
```

# String - zadanie

1. Odszyfruj wiadomość ze strony:

<http://dyba.it/encoded>  
msg

Wiedząc że litery zgubiły się w odmęcie innych znaków.

# String – zadanie: odp

<http://dyba.it/encoded>

```
1  a = "..."
2  odp = []
3  for i in a:
4      if i.isalpha():
5          odp.append(i)
6  print(''.join(odp))
```

# Zadanie domowe

W interpreterze pythona wpisz: 'import this'

Skopiuj tekst i przypisz go do zmiennej.

Następnie wypisz:

- a) Wszystkie zdania które mają słówko "is" w sobie.
- b) Wszystkie zdania złożone (z ,)
- c) Zmień wszystkie słowa rozpoczynające się z wielkiej litery na słowo Python
- d) Dowiedz się czemu Holendrzy mogą nie stosować zasad o tym że jest tylko jedna słuszna droga aby rozwiązać dany problem.)

# Zadanie domowe

W interpreterze pythona wpisz: 'import this'

Skopiuj tekst i przypisz go do zmiennej.

Następnie wypisz:

- a) Wszystkie zdania które mają słówko "is" w sobie.
- b) Wszystkie zdania złożone (z ,)
- c) Zmień wszystkie słowa rozpoczynające się z wielkiej litery na słowo Python
- d) Dowiedz się czemu Holendrzy mogą nie stosować zasad o tym że jest tylko jedna słuszna droga aby rozwiązać dany problem.)

# Zadanie domowe - odp

```
this = 'string from this'

print('ZAD A')
for line in this.split('\n'):
    if ' is ' in line:
        print(line)

print('ZAD B')
for line in this.split('\n'):
    if ', ' in line:
        print(line)

print('ZAD C')
for line in this.split('\n'):
    for word in line.split(' '):
        if word.istitle():
            line = line.replace(word, 'Python')
print(line)
```

# Stringi - slice

Pamiętacie jak się dostać do elementów listy ?

# Stringi - slice

```
>>> lista = ['a', 'b']
>>> lista[0]
'a'
```

# Stringi – slice zaw.

```
tekst[tekst.find('Beautiful'):tekst.find('Complex')]

tekst[:len(tekst) // 2]

tekst[::-int(len(tekst) ** 0.5)]

tekst[:tekst.find('.', len(tekst) // 2) + 1]
```

# Stringi – slice zad.

Używając tego samego tekstu z import this:

a) Wypisz wszystkie części zdań zawarte między , a .

b) Wypisz jedno słowo po słowie "is"

c) Wypisz jedno słowo przed słowem "is"

```
tekst[tekst.find('Beautiful'):tekst.find('Complex')]
```

```
tekst[:len(tekst) // 2]
```

```
tekst[::-int(len(tekst) * 0.5)]
```

```
tekst[:tekst.find('. ', len(tekst) // 2) + 1]
```

# Stringi – slice zad - odp

```
for line in tekst.split('\n'):
    to_print = line[line.find(', '):line.find('.')]
    if to_print:
        print(to_print)

for line in tekst.split('\n'):
    if ' is ' in line:
        to_print = line[line.find(' is ') + len(' is '):]
        to_print = to_print[:to_print.find(' ')]
        if to_print:
            print(to_print)

for line in tekst.split('\n'):
    if ' is ' in line:
        to_print = line[:line.find(' is ')]
        if to_print.count(' ') >= 1:
            to_print = to_print[to_print.rfind(' '):]
        if to_print:
            print(to_print)
```

# Funkcje wbudowane

# len

funkcja licząca długość np. stringu. Zwraca zawsze inta.

```
>>> len('test')
```

```
4
```

```
>>> len('Ala ma kota')
```

```
11
```

# Len zadania

1. Sprawdź czy len działa na intach lub floatach
2. Zmień przykład wykorzystując len aby pobierał imię:

```
>>> print('x' * input('ile liter ma twoje imie? '))  
ile liter ma twoje imie? 5
```

xxxxx

# Len zadania - odpowiedzi

1. Sprawdź czy len działa na intach lub floatach  
Nie, nie

2. Zmień przykład wykorzystując len aby pobierał imię:

```
>>> print('x' * len(input('podaj swoje imie: ')))  
podaj swoje imie: piotr  
xxxxxx
```

# Stringi - slice

```
>>> text = 'Python'  
>>> print(text[1])  
?
```

# Stringi - slice

```
>>> text = 'Python'  
>>> print(text[1])  
'y'
```

# Stringi – zadanie odp

string[początek:koniec:  
krok]

# Stringi – zadanie odp

string[początek:koniec:  
krok]

# Stringi – zadanie odp

```
x = 'text'  
x[1:]  
x[:3]  
x[1:2]
```

# Stringi slice - zadanie

text\_1 = "Ala ma Kota a kota ma Ale"

text\_2 = 'PyLadies.start() ma przerwe  
obiadowa o 14:00'

Napisz kod ktory po przekazaniu text\_1 lub  
text\_2 zwroci polowe stringa wykorzystujac  
sliceowanie i funkcje len.

```
x = 'text'  
x[1:]  
x[:3]  
x[1:2]
```

The image shows a dark-themed code editor window. At the top, the variable 'x' is assigned the value 'text'. Below it, four examples of string slicing are shown: 'x[1:]', 'x[:3]', and 'x[1:2]'. The first two examples result in the output 'text', while the last one results in 't'. The code editor has a green background for the code area and a dark grey header bar.

# abs, round

abs() – zwraca wartość absolutną liczby

round() – zaokrąglą liczbę domyślnie do l. całkowitych, lub do określonej liczby miejsc po przecinku jeśli przekażemy to.

```
>>> abs(-12)
```

```
12
```

```
>>> round(12.123311)
```

```
12.0
```

```
>>> round(12.123311, 2)
```

```
12.12
```

# Type, help, dir

type() – zwraca typ danych.

```
>>> type(12)
<type 'int'>
>>> type(12.123311)
<type 'float'>
>>> type('12.12')
<type 'str'>
>>> help(1)
>>> dir("")
```

# Zadanie: Pitagoras

1. Zmodyfikuj swoja skrypt do liczenia BMI aby zaokrąglą wynik do dwóch miejsc po przecinku.
2. Wykorzystując wiedzę z dzisiaj napisz skrypt który policzy przeciwnostokątną w trójkącie prostokątnym ze wzoru i zaokrąglij wynik do dwóch miejsc po przecinku:

$$A^{**2} + B^{**2} = C^{**2}$$

$$\sqrt{a^2 + b^2} = c$$

Z praw potęgowania pierwiastek kwadratowy to to samo co podniesienie do potęgi 0,5.

**abs, round**

# Logiczna logika

# Porównania

<  
>  
==  
!=  
<=  
>=  
is  
in  
not

# Porównania

```
2 < 4
1 > 3
'a' == True
0 != 1
2 <= 2.5
3 >= -1
'a' is 'b'
'k' in 'kot'
not 1
'c' not in 'kot'
```

```
2 < 5 <= 6
2 < 4 != 0
3 > 4 > 1
```

# Logika

OR  
AND

a	b	a OR b
True	True	True
True	False	True
False	True	True
False	False	False

a	b	a AND b
True	True	True
True	False	False
False	True	False
False	False	False

# Logika

```
'k' in 'kot' and 'o' in 'kot' and 't' in 'kot'
```

```
'k' in 'kot' or 'c' in 'kot'
```

```
not ('c' in 'kot' or 'd' in 'kot')
```

```
1 < 3 or 1 > 3
```

```
1 < 3 and 1 > 3
```

# Warunkowe warunki

# Warunek - if

element języka programowania, który pozwala na wykonanie różnych kroków algorytmu w zależności od tego czy zdefiniowane przez nas wyrażenie logiczne jest prawdziwe, czy fałszywe.

# Warunek - if

```
if 'Warunek':  
    print('Zrob cos')
```

Indentacja czyli wcięcia

# Warunek - if

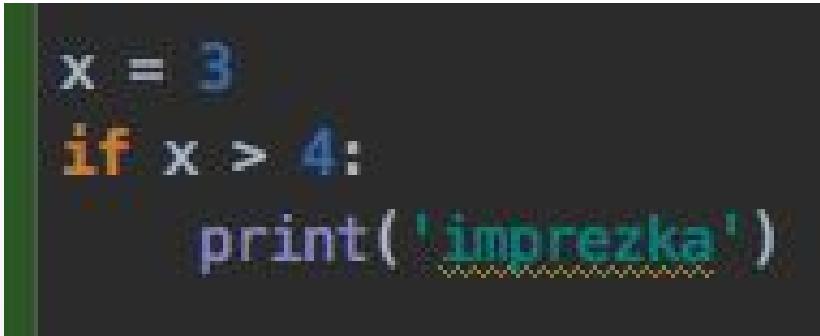
```
if 'Ala ma Kota':  
    print('Tak Ala ma kota')  
  
if True:  
    print('To zawsze jest prawda')  
  
if 'ala' in 'ma kota' or 'a' == 'a':  
    print('aaa')  
    print('bbb')  
  
if 'a':  
    if 'b':  
        if 'c':  
            print('d')
```

Indentacja czyli wcięcia

# if - zadanie

1. W naszym kalkulatorze BMI dodaj warunek, który w zależności od wyniku wypisze:  
niedowaga dla poniżej 20 pkt.  
norma dla wyników 20-25pkt  
nadwaga dla wyników powyżej 25pkt
2. Napisz kalkulator objętości prostopadłościanu oraz napisz warunek który sprawdź czy podstawa jest kwadratem i czy nasza bryła jest sześcianem idealnym.

print, input,float, str, int, if, <, >, <=, >=, ==, !=



```
x = 3
if x > 4:
    print('imprezka')
```

# if - zadanie

1. W naszym kalkulatorze BMI dodaj warunek, który w zależności od wyniku wypisze:
  - niedowaga dla poniżej 20 pkt.
  - norma dla wyników 20-25pkt
  - nadwaga dla wyników powyżej 25pkt

```
masa = float(input('Podaj swoja mase: '))
wzrost = float(input('Podaj swój wzrost: '))
bmi = masa / (wzrost ** 2)
if bmi < 20:
    odp = ' masz niedowage'
if 20 <= bmi <= 25:
    odp = ' jesteś w normie'
if bmi > 25:
    odp = ' masz nadwage'
print('Twoje BMI wynosi: ' + str(bmi) + odp)
```

# if - zadanie

2. Napisz kalkulator objętości prostopadłościanu oraz napisz warunek który sprawdź czy podstawa jest kwadratem i czy nasza bryła jest sześcianem idealnym.

```
a = float(input('bok a: '))
b = float(input('bok b: '))
h = float(input('Wysokosc: '))
obj = str(a * b * h)
print('Objetosc wynosi: ' + obj)
if a == b:
    print('podstawa jest kwadratem yey :)')
if a == b == c:
    print('to jest szescian idealny ' + 'yey!*3)
```

# Warunek cd.

if, elif, else

```
x = input('podaj literke')
if x == 'a':
    print('1')
elif x == 'b':
    print('2')
else:
    print('3')
```

# elif/else - zadanie

1. Poprzednie zadanie dostosuj używając elif i else zamiast tylko ifów.

print, input, float, str, int, if, **elif, else** <, >, <=, >=,  
==, !=

```
x = input('podaj literke')
if x == 'a':
    print('1')
elif x == 'b':
    print('2')
else:
    print('3')
```

# elif/else – zadanie: odp

1. Poprzednie zadanie dostosuj używając elif i else zamiast tylko ifów.

```
masa = float(input('Podaj swoja mase: '))
wzrost = float(input('Podaj swój wzrost: '))
bmi = masa / (wzrost ** 2)
if bmi < 20:
    odp = ' masz niedowage'
elif bmi <= 25:
    odp = ' jesteś w normie'
else:
    odp = ' masz nadwage'
print('Twoje BMI wynosi: ' + str(bmi) + odp)
```

# Listy zakupów

# Lista

Uporządkowana struktura danych służąca do reprezentacji zbiorów, w której elementy ułożone są w liniowym porządku.

Pusta lista w pythonie to:

[]

Lista z elementami to:

[1, 'Piotr', 2.12, 'co stam']

# Lista przykłady

```
['Kasia', 'Ania', 'Monika', 'Grzegorz']
```

```
[1, 1, 2, 3, 5, 8, 13]
```

```
[[['Kasia', 'Ania']], ['Grzegorz', 'Piotr']]
```

```
[3, 3.1, 3.14, 3.143]
```

# CRUD

C – Create - Stwórz

R – Read - Odczytaj

U – Update - Zmodyfikuj

D – Delete - Usuń

# Tworzenie listy

```
lista_imion = ['Kasia', 'Ania', 'Monika', 'Grzegorz']  
lista_fibo =[1, 1, 2, 3, 5, 8, 13]  
lista_list_imion = [['Kasia', 'Ania'], ['Grzegorz', 'Piotr']]  
lista_pi = [3, 3.1, 3.14, 3.143]
```

# C – tworzenie

```
lista_imion = ['Kasia', 'Ania', 'Monika', 'Grzegorz']
lista_imion.append('Natalia')
print(lista_imion)
>>> ['Kasia', 'Ania', 'Monika', 'Grzegorz', 'Natalia']

lista_fibo =[1, 1, 2, 3, 5, 8, 13]
lista_fibo.append(21)
print(lista_fibo)
>>> [1, 1, 2, 3, 5, 8, 13, 21]
```

# R – odczyt

```
lista_imion = ['Kasia', 'Ania', 'Monika', 'Grzegorz']
print(lista_imion[0])
>>> 'Kasia'
print(lista_imion[2])
>>> 'Monika'
print(lista_imion[-1])
>>> 'Grzegorz'
print(lista_imion[-2])
>>> 'Monika'
```

Programiści liczą od 0 !

# U - modyfikacja

```
>>> lista_imion = ['Kasia', 'Ania', 'Monika', 'Grzegorz']
>>> lista_imion[2]
'Monika'
>>> lista_imion[2] = 'Ewa'
>>> print(lista_imion)
['Kasia', 'Ania', 'Ewa', 'Grzegorz']
>>> lista_imion[-1] = 'Piotr'
>>> print(lista_imion)
['Kasia', 'Ania', 'Ewa', 'Piotr']
```

# D - usuwanie

```
>>> lista_imion = ['Kasia', 'Ania', 'Monika', 'Piotr', 'Ewa']
>>> lista_imion.pop()
'Ewa'
>>> lista_imion.pop(0)
'Kasia'
>>> print(lista_imion)
['Ania', 'Monika', 'Piotr']
>>> lista_imion.pop(-2)
'Monika'
>>> print(lista_imion)
['Ania', 'Piotr']
```

# D – usuwanie 2

```
>>> lista_imion = ['Kasia', 'Ania', 'Monika', 'Piotr', 'Ewa']
>>> lista_imion.remove('Ewa')
>>> print(lista_imion)
['Kasia', 'Ania', 'Monika', 'Piotr']
>>> lista_imion.remove('Piotr')
>>> print(lista_imion)
['Kasia', 'Ania', 'Monika']
>>> lista_imion.remove('Piotr')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
```

# D – usuwanie 3

```
lista_imion = ['Kasia', 'Ania', 'Monika', 'Piotr', 'Ewa']
imie_do_usuniecia = input('Podaj imię do usuniecia')
if imie_do_usuniecia in lista_imion:
    lista_imion.remove(imie_do_usuniecia)
```

# CRUD: lista - przykłady

```
>>> lista_fibo = [1, 1, 2, 3, 5, 8, 13]
>>> lista_fibo.append(lista_fibo[-2] + lista_fibo[-1])
>>> print(lista_fibo)
[1, 1, 2, 3, 5, 8, 13, 21]
```

```
>>> list_list = [['Kasia', 'Ania'], ['Grzegorz', 'Piotr']]
>>> list_list[0].append('Ewa')
>>> list_list[1].append(list_list[0].pop(0))
>>> print(list_list)
[['Ania', 'Ewa'], ['Grzegorz', 'Piotr', 'Kasia']]
```

# Lista - zadania

1. Napisz skrypt, który stworzy listę składającą się z trzech list a w każdej znajdzie się imię i wiek osoby.

Następnie wyświetl listę składającą się tylko z imion, tylko z wieków oraz każdą osobę osobno.

print, input, float, str, int, if, **elif, else <, >, <=, >=, ==, !=**

listy:

C [].append()

R [][-999 – 999]

U [][-999 – 999] = 'nowa wartosc'

D [].pop()

```
>>> lista_fibo = [1, 1, 2, 3, 5, 8, 13]
>>> lista_fibo.append(lista_fibo[-2] + lista_fibo[-1])
>>> print(lista_fibo)
[1, 1, 2, 3, 5, 8, 13, 21]
```

# Lista – zadania: odp

```
dane = [
    [input('Podaje imie'), input('Podaje wiek')],
    [input('Podaje imie'), input('Podaje wiek')],
    [input('Podaje imie'), input('Podaje wiek')],
]
print(dane[0][0], dane[1][0], dane[2][0])
print(dane[0][1], dane[1][1], dane[2][1])
print(dane[0])
print(dane[1])
print(dane[2])
```

# Zadanie: BMI

Napisz skrypt, który policzy BMI i w zależności od płci zgodnie z tabelką wypisze diagnozę.

Kobiety:

BMI	Diagnoza
< 16,5	Anoreksja
16,5 - 20	Niedowaga
20 - 25	Norma
25 - 30	Nadwaga
30 +	Otyłość

Mężczyźni:

BMI	Diagnoza
< 18,5	Anoreksja
18,5 – 22,5	Niedowaga
22,5 - 27,5	Norma
27,5 - 32,5	Nadwaga
32,5 +	Otyłość

# Listy - slice

Slice w listach działa dokładnie jak w stringach ☺

# Listy- zadanie

Stwórz liste wykorzystując lista =  
list(range(0,100))

Sprawdź co robią następujące rzeczy:

1. lista [35:44]
2. lista [64:]
3. lista [:34]
4. lista [::-1]
5. lista [:34:4]
6. lista [-65:-28]

# Lista – zadanie odp

```
lista[początek:koniec:kr  
ok]
```

# Lista - zadania

1. Stwórz skrypt 'Lista zakupów', który najpierw poprosi o 5 rzeczy, po które żona wysłała męża a potem o kolejne 5, które to ten mąż kupił a niekoniecznie znajdowały się na liście. Na zakończenie wyświetli podsumowanie zakupów.

print, input, float, str, int, if, **elif**, **else** <, >, <=, >=,  
==, !=

listy:

C [].append()

R [][-999 – 999]

U [][-999 – 999] = 'nowa wartosc'

D [].pop()

```
# Skrypt, który liczy BMI i w zależności od płci wypisze diagnozę.  
plec = input('\nplec (k/m): ')  
bmi = round((float(input('waga (kg): ')) / int(input('wzrost (cm): ')) ** 2) * 10000, 2)  
print('\nTwoje BMI wynosi' + str(bmi))  
if plec == 'k':  
    if bmi < 16.5:  
        print('Anoreksja')  
    elif 16.5 <= bmi < 20:  
        print('Niedowaga')  
    elif 20 <= bmi < 25:  
        print('Norma')  
    elif 25 <= bmi < 30:  
        print('Nadwaga')  
    else:  
        print('Otyłość')  
else:  
    if bmi < 18.5:  
        print('Anoreksja')  
    elif 18.5 <= bmi < 22.5:  
        print('Niedowaga')  
    elif 22.5 <= bmi < 27.5:  
        print('Norma')  
    elif 27.5 <= bmi < 32.5:  
        print('Nadwaga')  
    else:  
        print('Otyłość')
```

# Zadanie domowe

```
print("Zona poprosila meza o kupno nastepujacych rzeczy: ")
lista_zona = [input('1: '), input('2: '), input('3: '), input('4: '), input('5: ')]
print('Oto co zakupil maz: ')
lista_maz = [input('1: '), input('2: '), input('3: '), input('4: '), input('5: ')]
for z in lista_zona:
    if z not in lista_maz:
        print('Maz nie kupil' + z)

for z in lista_maz:
    if z not in lista_zona:
        print('Maz kupil dodatkowo' + z)
```

# Tuple czyli krotka

Tuple (po polsku krotka)

To niemutowalna lista tzn. że po jej stworzeniu nie możemy jej edytować

przykład:

```
>>> krotka = ('a', 'b')
>>> krotka_2 = (1, 4 ,6)
>>> krotka_3 = ([1, 2, 3], [7, 8 ,9])
>>> krotka_3[0]
[1, 2, 3]
>>> krotka_3[0][1] = 5
>>> print(krotka_3)
([1, 5, 3], [7, 8 ,9])
```

# Słowniki od A do Z

# Słownik - dict

Ostatnia z podstawowych struktur danych.  
Słownik przechowuje pary (unikatowy klucz, wartość) i umożliwia dostęp do wartości poprzez podanie klucza.

Klucze: int, floaty, stringi

# Tworzenie Słownika

```
>>> moj_pusty_slownik = {}
>>> print(moj_pusty_slownik)
{}
>>> moj_nowy_slownik = {'klucz': 'wartosc'}
>>> print(moj_nowy_slownik)
{'klucz': 'wartosc'}
```

# Read - odczyt

```
>>> moj_slownik = {'klucz': 'wartosc', 'klucz_2': 'wartosc_2', 1: 'jeden'}
>>> print(moj_slownik['klucz'])
'wartosc'
>>> print(moj_slownik['klucz_2'])
'wartosc_2'
>>> print(moj_slownik[1])
'jeden'
```

# Create – tworzenie wpisu

```
>>> moj_slownik = {}
>>> moj_slownik['klucz'] = 'b'
>>> print(moj_slownik)
{'klucz': 'b'}
>>> moj_slownik['klucz_f'] = 'c'
>>> print(moj_slownik)
{'klucz': 'b', 'klucz_f': 'c'}
>>> moj_slownik[1] = 'd'
>>> print(moj_slownik)
{'klucz': 'b', 'klucz_f': 'c', 1: 'd'}
```

# Update – modyfikacja wpisu

```
>>> moj_slownik = {'klucz': 'b', 'klucz_f': 'c', 1: 'd'}
>>> moj_slownik['klucz'] = 'bbbbbb'
>>> print(moj_slownik)
{'klucz': 'bbbbbb', 'klucz_f': 'c', 1: 'd'}
>>> moj_slownik['klucz_f'] = 'f jak felieton'
>>> print(moj_slownik)
{'klucz': 'bbbbbb', 'klucz_f': 'f jak felieton', 1: 'd'}
>>> moj_slownik[1] = 'LOL ^^'
>>> print(moj_slownik)
{'klucz': 'bbbbbb', 'klucz_f': 'c', 1: 'LOL ^^'}
```

# Delete – Usuwanie wpisu

```
>>> moj_slownik = {'klucz': 'b', 'klucz_f': 'c', 1: 'd'}
>>> del moj_slownik['klucz']
>>> print(moj_slownik)
{'klucz_f': 'c', 1: 'd'}
>>> del moj_slownik[1]
>>> print(moj_slownik)
{'klucz_f': 'c'}
>>> del moj_slownik['klucz_f']
>>> print(moj_slownik)
{}
```

# Słownik Przykłady

```
{  
    'tekst': 'jakis tekst',  
    'liczba': 123,  
    'mala liczba': 0.001,  
    'funkcja': input('powiedz cos')  
    'slownik': {  
        'slownik': {  
            'w': 'slowniku'  
        }  
    }  
    'lista': [1, 1, 2, 3, 5]  
}
```

# Słownik Przykłady

```
slownik = {'a': 1, 'b': 2}
klucz = 'c'
if klucz in slownik:
    print(slownik[klucz])
else:
    print('Brak takiego klucza')
```

# Słownik Przykłady

```
>>> moj_slownik = {'imie': 'Piotr', 'nazwisko': 'Dyba', 'wiek': 99}
>>> moj_slownik['wiek'] += 1
>>> print(moj_slownik)
{'imie': 'Piotr', 'nazwisko': 'Dyba', 'wiek': 100}
>>> moj_slownik['zainteresowania'] = []
>>> print(moj_slownik)
{'imie': 'Piotr', 'nazwisko': 'Dyba', 'wiek': 100, 'zainteresowania': []}
>>> del moj_slownik['nazwisko']
>>> print(moj_slownik)
{'imie': 'Piotr', 'wiek': 100, 'zainteresowania': []}
>>> moj_slownik['zainteresowania'].append('fotografia')
>>> moj_slownik['zainteresowania'].append('wspinaczka')
>>> moj_slownik['zainteresowania'] += ['gory', 'nurkowanie']
>>> from pprint import pprint as pp
>>> pp(moj_slownik)
{
    'imie': 'Piotr',
    'nazwisko': 'Dyba',
    'wiek': 100,
    'zainteresowania': [
        'fotografia',
        'wspinaczka',
        'gory',
        'nurkowanie',
    ]
}
```

# Słownik Przykłady

```
>>> moj_slownik = {'imie': 'Piotr', 'nazwisko': 'Dyba', 'wiek': 99}
>>> moj_slownik['wiek'] += 1
>>> print(moj_slownik)
{'imie': 'Piotr', 'nazwisko': 'Dyba', 'wiek': 100}
>>> moj_slownik['zainteresowania'] = []
>>> print(moj_slownik)
{'imie': 'Piotr', 'nazwisko': 'Dyba', 'wiek': 100, 'zainteresowania': []}
>>> del moj_slownik['nazwisko']
>>> print(moj_slownik)
{'imie': 'Piotr', 'wiek': 100, 'zainteresowania': []}
>>> moj_slownik['zainteresowania'].append('fotografia')
>>> moj_slownik['zainteresowania'].append('wspinaczka')
>>> moj_slownik['zainteresowania'].append('gory')
>>> moj_slownik['zainteresowania'].append('nurkowanie')
>>> from pprint import pprint as pp
>>> pp(moj_slownik)
{
    'imie': 'Piotr',
    'nazwisko': 'Dyba',
    'wiek': 100,
    'zainteresowania': [
        'fotografia',
        'wspinaczka',
        'gory',
        'nurkowanie',
    ]
}
```

# Słownik zadanie

1. Stwórz małą bazę danych, w której będziesz przechowywała takie informacje jak:

- Imię
- Nazwisko
- Data urodzenia
- Zawód
- Zainteresowania
- Fikcyjny stan konta

Siebie i koleżanki/kolegi obok.

```
dane = {
    'dyba': {
        'imie': 'Piotr',
        'data ur': '01092017',
        'zainteresowania': [
            'fotografia',
            'wspinaczka',
            'gory',
            'nurkowanie',
        ],
        'zawod': 'programista',
        'stan konta': -123.12
    },
    'nowak': {
        'imie': 'Kamila',
        'data ur': '01011990',
        'zainteresowania': [
            'spawanie',
            'boks',
            'pilka nozna',
            'jednorozce',
        ],
        'zawod': 'lekarz',
        'stan konta': 999123.12
    }
}
```

# Słownik

## zadanie -

## odp

Pętla dla każdego (for)  
zrobię coś fajnego

# Pętla

to jedna z podstawowych konstrukcji programowania (obok instrukcji warunkowej - if). Umożliwia cykliczne wykonywanie ciągu instrukcji określoną liczbę razy aż do momentu zajścia pewnych warunków lub w nieskończoność.

# Pętla for – konstrukcja

```
for cos in wiele_cosow:  
    zrob_cos(cos)
```

# Pętla for

```
i = 1
for letter in 'Ala ma kota':
    print(letter * i)
    i += 1
```

```
A
ll
aaa
```

```
mmmmm
...  
.
```

# od do - range

```
for liczba in range(10):  
    print(liczba)
```

```
0  
1  
2  
...  
9
```

# Petla for przyklady

```
from pprint import pprint as pp
zliczacz = {}
wyraz == input('Podaj wyraz')
for let in wyraz:
    if let in zliczacz:
        zliczacz[let] += 1
    else:
        zliczacz[let] = 1
pp(zliczacz)
```

# Petla for przyklady

```
for a in range(10):
    for b in range(9):
        for c in range(8):
            d = a * b + c
            print (d)
```

# for - zadanie

Wykorzystując pętle narysuj prostą choinkę:

```
^  
/\\  
//\\\\  
///\\\\  
///\\\\\  
////\\\\\  
/////\\\\\  
//////\\\\\  
////////\\\\
```

\* Dodaj bombki  
(\*)

```
^  
/\\  
//\*\\  
/*/\\\\\  
///\\\\  
///\\\\*\  
////\\\\*\  
//////\\\\  
*****\\\\*
```

# for - zadanie

```
for a in range(10):
    galazka = ' ' * (10 - a) + '/' * a + '\\' * a
    print(galazka)
```

# for - zadanie

```
for a in range(10):
    galazka = ' ' * (10 - a)
    if a % 4 == 0:
        galazka += '/' + '*' + '/' * (a-2) + '\\\\' * a
    elif a % 3 == 0:
        galazka += '/' * a + '\\\\' * (a-2) + '*' + '\\\\'
    else:
        galazka += '/' * a + '\\\\' * a
print(galazka)
```

Pętla dopóki mogę (while)  
będę robił coś fajnego

# Pętla while

```
while True:  
    zrob_cos()
```

# Pętla while - przykłady

```
czy_zakonczyc = False
while not czy_zakonczyc:
    odpowiedz = input('Czy zakonczyc program T/N')
    if odpowiedz == 'T'
        czy_zakonczyc =True
```

# while - zadanie

1. Napisz skrypt do liczenia BMI, który po policzeniu spyta się czy policzyć dla kolejnej osoby czy wyjść z aplikacji.

```
while True:  
    zrob_cos()
```

# while – zadanie - odp

1. Napisz skrypt do liczenia BMI, który po policzeniu spyta się czy policzyć dla kolejnej osoby o:

```
wlaczone = True
while wlaczone:
    masa = float(input('Podaj swoja mase: '))
    wzrost = float(input('Podaj swój wzrost: '))
    bmi = masa / (wzrost ** 2)
    if bmi < 20:
        odp = ' masz niedowage'
    elif bmi <= 25:
        odp = ' jesteś w normie'
    else:
        odp = ' masz nadwage'
    print('Twoje BMI wynosi: ' + str(bmi) + odp)
    odpowiedz = input('Czy zakończyć program T/N')
    if odpowiedz == 'T':
        wlaczone = False
```

# Zadanie domowe

Oblicz BMI wszystkich członków rodziny

Stwórz program który spyta o imię, wagę oraz wzrost każdego z członków rodziny i wypisze ich BMI. Przykład użycia:

Podaj swoje imię: Mieczysław

Podaj wagę: 85

Podaj wzrost: 181

Czy dodać kolejną osobę? Tak

Podaj swoje imię: Marian

Podaj wagę: 75

Podaj wzrost: 165

Czy dodać kolejną osobę? Nie

Mieczysław BMI = 25.95

Marian BMI = 27.55

# `nasza_funkcja()`

# Funkcja

to wydzielona część programu wykonująca wcześniej zdefiniowane operacje.

Podprogramy stosuje się, aby uprościć program główny i zwiększyć czytelność kodu.

Funkcje wbudowane w Pythona to np.: len(), abs()

# Funkcja – konstrukcja, wywołanie

```
def moja_funkcja():
    print('To jest moja funkcja')

moja_funkcja()
moja_funkcja()
```

# Funkcja z argumentem

```
def moje_dodawanie(a, b):
    c = a + b
    print('Twoim wynikiem jest liczba: ' + str(c))

moje_dodawanie(2, 3)
```

# Funkcja z argumentem - kwargs

```
def moje_potegowanie(a, b=2):
    c = a ** b
    print('Liczba' + str(a) + ' do ' + str(b) +' jest liczba: ' + str(c))

moje_potegowanie(2)
moje_potegowanie(2, b=10)
```

# Funkcja - zadania

1. Napisz funkcję, która policzy objętość prostopadłościanu z podanych długości boków podstawy i wysokości.

# Funkcja - zadania

```
def objetosc():
    bok_a = float(input('Podaj bok a: '))
    bok_b = float(input('Podaj bok b: '))
    wysokosc = float(input('Podaj wysokosc: '))
    wynik_str = str(bok_a * bok_b * wysokosc)
    print('Twoje objetosc wynosi: ' + wynik_str)

objetosc()
```

# Funkcja - zadania

```
def objetosc(bok_a, bok_b, wysokosc):  
    print('Twoje objetosc wynosi: ' + str(bok_a * bok_b * wysokosc))  
  
bok_a = float(input('Podaj bok a: '))  
bok_b = float(input('Podaj bok b: '))  
wysokosc = float(input('Podaj wysokosc: '))  
objetosc(bok_a, bok_b, wysokosc)
```

# Funkcja - zadania

```
def objetosc(bok_a, bok_b, wysokosc):
    print('Twoje objetosc wynosi: ' + str(bok_a * bok_b * wysokosc))

def policz_obj():
    bok_a = float(input('Podaj bok a: '))
    bok_b = float(input('Podaj bok b: '))
    wysokosc = float(input('Podaj wysokosc: '))
    objetosc(bok_a, bok_b, wysokosc)

policz_obj()
```

# Funkcja \*args

```
def funkcja_a(*args):
    for ar in args:
        print(ar)

>>> funkcja_a('a', 'b', 2, 3)
a
b
2
3
```

# Funkcja \*args - zadanie

- Wykorzystując funkcje i \*args zaimplementuj funkcję range samemu

range([początek], koniec, [skok]) która zwróci listę w podanym zakresie.

W nawiasach kwadratowych zostały ujęte argumenty nieobowiązkowe (opcjonalne)

```
>>> print(moj_range(5))
[0, 1, 2, 3, 4]
>>> print(moj_range(5, 10))
[5, 6, 7, 8, 9]
>>> print(moj_range(0, 100, 9))
[0, 9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99]
```

```
def moj_range(*args):
    a_tab = []
    i = 0
    s = 1
    if len(args) == 1:
        end = args[0]
    elif 3 >= len(args) >= 2:
        end = args[1]
        i = args[0]
    else:
        raise TypeError
    if len(args) == 3:
        s = args[2]
    while i < end:
        a_tab.append(i)
        i += s
    return a_tab
```

Funkcja  
\*args - odp

# Funkcja zwracająca wiele

```
def funkcja_z2():
    return 'ala', 'ma', 'kota'

>>> a = funkcja_z2()
>>> a
('ala', 'ma', 'kota')
>>> a, b = funkcja_z2()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)
>>> a, b, c = funkcja_z2()
>>> a
'ala'
>>> b
'ma'
>>> c
'kota'
```

# Łapanie wyjątkowych błędów i trolli

# Po co wyłapywać błędy

- Aby program działał mimo drobnych błędów, szczególnie tych których się spodziewamy.
- Aby zabezpieczyć program przed trollami.
- Aby zabezpieczyć aplikację przed atakami.

# Try / Except

```
try:  
    print(z)  
except:  
    print("Here is a random error")
```

# Try / Except

1. Na początku wykonywana jest klauzula **try** (czyli instrukcje pomiędzy **try** a **except**).
2. Jeżeli nie pojawi się żaden wyjątek klauzula **except** jest pomijana. Wykonanie instrukcji **try** uważa się za zakończone.
3. Jeżeli podczas wykonywania klauzuli **try** pojawi się wyjątek, reszta niewykonanych instrukcji jest pomijana.

# Try / Except - Konkrety:

```
x = input()
y = input()

try:
    z = x/y
    print(z)

except ZeroDivisionError:
    print("divide by zero")
```

# Try / Except Przykłady:

```
try:  
    do_smth()  
except:  
    pass
```

# Try / Except Przykłady:

Nie rob tak nigdy !

```
try:  
    do_smtth()  
except:  
    errors_should_never_pass  
    silently.
```

# Try / Except Przykłady:

```
for x in some_container:  
    try:  
        x += 1  
    except KeyError:  
        print("There is no key like that")  
    except IndexError:  
        print("There is no key like that")
```

# Try / Except Przykłady:

```
try:  
    x += 1  
except Exception as e:  
    print(e)
```

# Try / Except Przykłady:

```
def czy_to_jest_cyfra(tekst):
    if isinstance(tekst, (float, int)):
        return tekst
    if not isinstance(tekst, str):
        return False
    tekst = tekst.replace(',', '.')
    try:
        float(tekst)
        return tekst
    except ValueError:
        return False

def czy_ala_ma_kota(wiek_ali):
    wiek_ali = czy_ala_ma_kota(wiek_ali)
    if wiek_ali and wiek_ali < 18 or wiek_ali > 40:
        return True
    return False

czy_ala_ma_kota('25,5')
```

# Try / Except zadanie:

W naszej ulubionej aplikacji do BMI zrób szereg zmian:  
dopytywała się o poprawne dane aż nie zostaną one podane przez użytkownika.

```
def czy_to_jest_cyfra(tekst):
    if isinstance(tekst, (float, int)):
        return tekst
    if not isinstance(tekst, str):
        return False
    tekst = tekst.replace(',', '.')
    try:
        float(tekst)
        return tekst
    except ValueError:
        return False

def czy_ala_ma_kota(wiek_ali):
    wiek_ali = czy_ala_ma_kota(wiek_ali)
    if wiek_ali and wiek_ali < 18 or wiek_ali > 40:
        return True
    return False

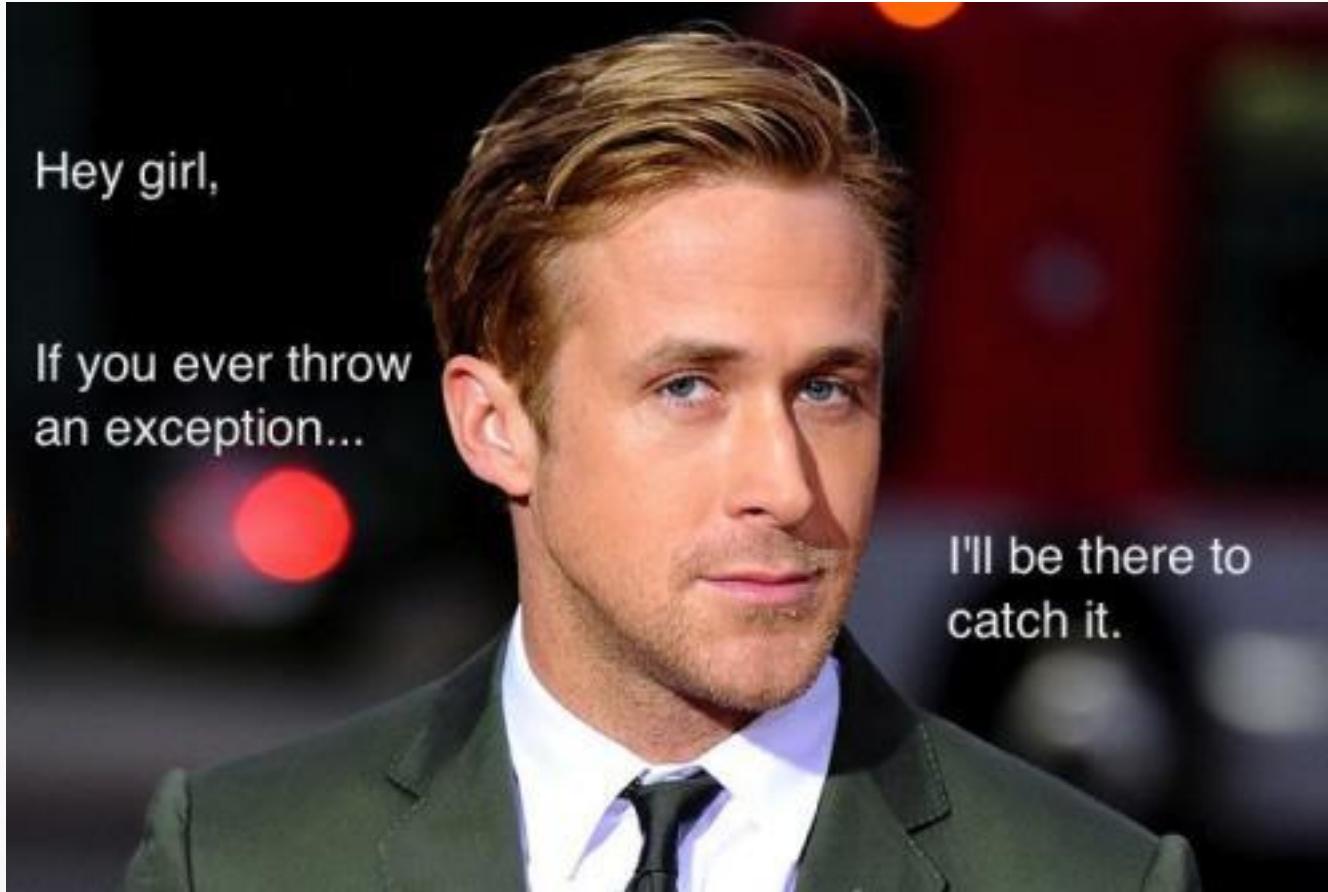
czy_ala_ma_kota('25,5')
```

# Try / Except zadanie-odp :

```
def pobierz_floata(pytanie):
    while True:
        try:
            return float(input(pytanie))
        except ValueError:
            print('!!! Podaj poprawna cyfre !!!')

masa = pobierz_floata('Podaj swoja mase: ')
wzrost = pobierz_floata('Podaj swój wzrost: ')
bmi = str(masa / (wzrost ** 2))
print('Twoje BMI wynosi: ' + bmi)
```

# Try / Except



# Try / Except

Hey Girl

You must be  
an exception

Because I want  
to handle you

# Try / Except Zadanie:

Zabezpiecz naszą grę zgadnij liczbę ze stu w 5 szansach i zabezpiecz przed Trollami:

Sprawdź czy podane dane przez użytkownika są:

- a) Liczbą, całkowitą
- b) Z zakresu przewidzianego przez grę

Jeśli nie są to spróbuj je 'ogarnąć' lub jeśli się nie da to powiadom gracza o błędnych danych i pobierz nowe.

# Try / Except Zadanie:

```
def jest_ok(cyfra, zakres):
    while True:
        try:
            cyfra = int(cyfra)
            if cyfra in range(zakres[0], zakres[1]):
                return cyfra
            else:
                print('Liczba nie znajduje sie w zakresie od {} do {}'.format(*zakres))
        except ValueError:
            print('!!! Nie podales poprawnej Liczby !!!')
    cyfra = input('Podaj liczbe w zakresie od {} do {}'.format(*zakres))
```

# Import losowości i innych Modułów

# Moduły

Python posiada  
~245  
wbudowanych  
modułów

# Moduły - jak

```
import this
import math
import random

from math import sqrt
from collections import namedtuple
from random import randint, choice

from pprint import pprint as pp
```

# Moduły – jak nie

```
from math import *
```

# Moduły – random

```
import random
random.choice(list, dict, str)
random.randint(int, int)
random.random()
random.shuffle(list, str)
```

# Moduły – random

```
from random import choice, randint, random, shuffle
choice(list, dict, str)
randint(int, int)
random()
shuffle(list, str)
```

# Moduły – random

Napisz prostą grę, która po wylosowaniu liczby w zakresie 1, 100 pozwoli graczowi ją odgadnąć w 5 próbach zwracając jedynie informację, że podana przez gracza liczba jest mniejsza, większa lub że trafił i wygrał lub skończyły mu się próby i przegrał. Przykładowy dialog:

Użytkownik: 33

Komputer: moja liczba jest mniejsza

U: 15

K: moja liczba jest większa

U: 22

K: Zgadłeś ! Wygrałeś 22 to moja liczba.

```
from random import ...
choice(list, dict, str)
randint(int, int)
random()
shuffle(list, str)
```

\* Dodaj możliwość gry w zakresie 1,1000 , po zakończeniu gry gra powinna się spytać czy zagrać jeszcze raz.

# Zad dom

Napisz prostszy wariant gry wisielec. Gra powinna:

1. Wybrać losowe hasło z puli min. 20 haseł i zaprezentować je w postaci zakrytej np.:  
\_ \_ \_ lub \* \* \* dla słowa „kot”
2. Gracz w swojej turze powinien móc próbować zgadnąć literkę lub całe hasło.
3. Po odgadnięciu literki wszystkie literki w słowie powinny się odkryć dla hasła MAMA:

Hasło: \_ \_ \_ \_

Gracz: M

Hasło M \_ M \_

4. Gra trwa aż gracz odgadnie hasło na końcu podaje wynik w postaci ilości tur które gracz potrzebował aby rozwiązać zagadkę.
5. Gracz wpisując słowo np. exit powinien przegrać i dostać odpowiedź i aktualny wynik

# Powtórka całości

# Zadanie Domowe 1

Napisz prostą grę, która po wylosowaniu liczby w zakresie 1, 100 pozwoli graczowi ją odgadnąć w 5 próbach zwracając jedynie informacje że podana przez gracza liczba jest mniejsza, większa lub że trafił i wygrał lub skończyły się próby i przegrał. Przykładowy dialog:

Użytkownik: 33

Komputer: moja liczba jest mniejsza

U: 15

K: moja liczba jest większa

U: 22

K: Zgadłeś ! Wygrałeś 22 to moja liczba.

```
from random import ...  
choice(list, dict, str)  
randint(int, int)  
random()  
shuffle(list, str)
```

\* Dodaj możliwość gry w zakresie 1,1000 , po zakończeniu gry gra powinna się spytać czy zagrać jeszcze raz.

# Zadanie Domowe 2

Napisz prostszy wariant gry wisielec. Gra powinna:

1. Wybrać losowe hasło z puli min. 20 haseł i zaprezentować je w postaci zakrytej np.:  
\_ \_ \_ lub \* \* \* dla słowa „kot”
2. Gracz w swojej turze powinien móc próbować zgadnąć literkę lub całe hasło.
3. Po odgadnięciu literki wszystkie literki w słowie powinny się odkryć dla hasła MAMA:

Hasło: \_ \_ \_ \_

Gracz: M

Hasło M \_ M \_

4. Gra trwa aż gracz odgadnie hasło na końcu podaje wynik w postaci ilości tur które gracz potrzebował aby rozwiązać zagadkę.
5. Gracz wpisując słowo np. exit powinien przegrać i dostać odpowiedź i aktualny wynik

# Zadanie 1

Napisz funkcję, która sprawdzi czy dana liczba jest liczbą brzydką.

Liczby brzydkie to takie które są podzielne przez 2, 3 lub 5. Liczba 1 również jest liczbą brzydką.

Przykład:

Input : ugly(12)

Output : True

Input : ugly(11)

Output : False

# Zadanie 2

Napisz funkcję, która zamieni czas (HH:MM:SS) na sekundy, minuty, godziny.

Przykład:

Input : get\_time('2:00:00', 's')

Output : 7200

Input : get\_time('12:00:00', 'm')

Output : 720

Input : get\_time('1:15:00', 'h')

Output : 1.25

# Zadanie 3

Napisz funkcję która spłaszczy strukturę zagnieżdzonych list do płaskiej listy:

Input : flatten([[[[[[[[[ 'a' ] 'b' ]]] 'c' ]]] 'd' ]] 'e'])

Output : ['a', 'b', 'c', 'd', 'e']

# Zadanie 4

Napisz algorytm który sprawdzi czy podane dane są dopuszczalne – validator:

a) adresu e-mail:

Input : validate\_email('piotr@dyba.com.pl')

Output : True

Input : validate\_email('piotr@dyba.com.pl')

Output : False

Input : validate\_email('piotr@dyba')

Output : False

b) kodu pocztowego

Input : validate\_zip('12-345')

Output : True

Input : validate\_zip('12345')

Output : False

Input : validate\_zip('1233-3345')

Output : False

# Zadanie 5

Napisz funkcje która zamieni cyfre arabska na cyfre rzymska:

$1 = I$   $5 = V$   $10 = X$   $50 = L$   $100 = C$   $500 = D$   $1000 = M$

Input : arb\_2\_rom(123)

Output : “CXXIII”

Input : arb\_2\_rom(1164)

Output : “MCLXIV”

Input : arb\_2\_rom(3338)

Output : “MMMCCCXXXVIII”

# Zadanie 6

W Długim tekście znajdź najdłuższe słowo(a):

Input : longest\_word('ala ma kota a kot ma pas ktory mowi blablabla')

Output : "blablabla"

# Q&A

