

# Podstawy Informatyki

**Instytut Telekomunikacji, EiT**

dr inż. Jarosław Bułat

[kwant@agh.edu.pl](mailto:kwant@agh.edu.pl)

# Plan prezentacji

- » regulamin przedmiotu
- » zakres wiadomości/umiejętności
- » podstawowe pojęcia
- » sposoby reprezentacji informacji
- » czym jest komputer - maszyna Turinga
- » pierwszy program
- » GIT **TODO**
- » jak się uczyć informatyki?
- » AI/ML w programowaniu: Copilot, CodeX (**new 2022!!!**)
- » MOOC

# Regulamin przedmiotu - PI

- » wykład (15)\*1.5h + Lab (15) \*1.5h
- » wykład jest instrukcją do Labu (!)
- » lab jest obowiązkowy: L. Janowski, M. Grega
- » lab to praktyczna realizacja zadań
  - preferowany Linux
  - można przynieść własny laptop (**BYOD**)
  - podstawowy język C++, (~~Python~~)
- » ocena z Lab to dwa kolokwia\*: SAMODZIELNE pisanie kodu
  - można korzystać z notatek, książek, Internetu
  - nie można komunikować się z innymi (np. FB, fora, ChatGPT, etc...)
- » ocena z przedmiotu == ocena z Lab (nie ma egzaminu)
- » Konsultacje: e-mail/zoom/discord
- » about://me

# Regulamin przedmiotu - MiTP

- » MiTP I: wykład (14)\*1.5h + Lab (14) \*1.5h (egzamin)
- » MiTP II: wykład (7)\*1.5h + Lab (14) \*1.5h (nie ma egzaminu - ocena z lab.)
- » wykład jest instrukcją do Labu (!)
- » lab jest obowiązkowy: Andres Vejar, Jarosław Bułat, Krzysztof Rusek, Andrzej Matiolański
- » lab to praktyczna realizacja zadań
  - preferowany Linux
  - można przynieść własny laptop (**BYOD**)
  - podstawowy język C++, (Python, Matlab w semestrze letnim)
- » ocena z Lab: zasady poda prowadzący lab
- » egzamin: UPeL
  - piszą wszyscy, termin zerowy tylko dla 4.5+@Lab w terminie
- » ocena z przedmiotu:  $\text{floor}((\text{lab} + \text{egz})/2)$  ) przykładowo 4.5, 4 -> 4
- » Konsultacje: e-mail/zoom/discord
- » about://me

# Regulamin przedmiotu

Pytania?

# Motywacja :-)

- » informatyka to rozległa dziedzina
- » nie da się uczyć “chronologicznie”
- » część informacji musisz przyjąć “na wiarę”, później zostanie sprecyzowana
- » **nie zniechęcaj się jeżeli czegoś nie rozumiesz**
- » jeżeli umiesz programować, przychodź na wykład:
  - poznasz inny punkt widzenia
  - poprawisz mnie jak się pomylę :-)

Czegoś nie rozumiesz?  
zapytaj !!!

# Czegoś nie rozumiesz?

## zapytaj !!!

teraz (na wykładzie)  
później (e-mail, konsultacje)  
prowadzącego lab/wykład  
kolegi/koleżanki



# wiadomości/umiejętności

- » informatyka stosowana: **będziesz potrafił napisać program rozwiązujący określony problem**
- » duże zróżnicowanie Waszej wiedzy/doświadczenia (**możliwe projekty**)
- » zakres przedmiotu **NIE obejmuje**: teorii informacji, budowy systemu operacyjnego, budowy/działania kompilatorów, etc...
- » zakres przedmiotu **obejmuje**:
  - praktyczne umiejętności programowania (C++, python)
  - podstawowe techniki/metody
  - podstawowe algorytmy (umiejętność implementacji)
  - analiza, uruchomienie, debugowanie kodu
  - elementy pracy zespołowej (git, standardy kodowania, etc...)
- » Narzędzia: Linux, g++, GIT (gitlab), IDE: (vim ;-), VSC), UPeL, ...

# Workflow

- » Wszystkie materiały (lab/wykład) w repozytorium **git**
- » Używamy własnej instancji GitLab-a (kwestie RODO)  
<https://gitlab.tele.agh.edu.pl>
  - autoryzacja SSO (LDAP - FreeIPA, <https://ipa.kt.agh.edu.pl>)
  - na pierwszych labach dostaniecie login/password
  - na gitlabie logujecie się przez zakładkę **FreeIPA**
  - automatycznie zostanie utworzone konto i przypisane do odpowiednich grup (to trwa do 24h)
  - hasło traktujcie jako “transportowe” proszę je zmienić - FreeIPA
  - to samo SSO pozwala Wam dostęp do VPN’a Instytutu (dostęp do podsieci Instytutu) - to nie jest ten sam VPN który daje AGH

# ChatGPT

- » Modele generatywne zmieniły IT w ostatnich 2 latach
  - Copilot / Codex (GitHub-MS / OpenAI.com)
  - GPT-4 / Llama - instruction tune (07.2023)
- » **Teraz:**
  - autouzupełnianie kodu weszło na kolejny poziom
  - generowanie całych fragmentów (np. klas) kodu
  - bardzo dobra jakość kodu - czasami dydaktyczna
- » **Przyszłość** (za 5 lat, może za 5 miesięcy?):
  - zarządzanie projektem (program/product manager)
  - rewolucja w UI: mysz+klawiatura -> mikrofon+kamera
  - optymalizacja kodu dla konkretnego użytkownika (sprzęt + dane)
  - generowanie kompletnych programów, wręcz systemów?
- » Historia Informatyki: ASM -> C/C++ -> Python/Mojo -> Prompt (?)
- » Pytania:
  - czy trzeba się uczyć programować (w sensie pisać/rozumieć kod źródłowy)?
  - jak wykorzystywać AI w procesie tworzenia softu?
  - jak się uczyć?

# ChatGPT

- » Teraz (październik 2023) żyjemy już w epoce **post AI** (to już się wydarzyło!):
  - kto nie używa modeli generatywnych ten traci (wolnie tworzy soft)
  - kto nie rozumie kodu (nie umie programować) ten nie jest w stanie ocenić jakości tego co dostanie od ChatGPT
  - należy postawić na szerszą wiedzę, uczyć się/rozumieć wszystkiego po trochu
  - chyba warto założyć, że dużo kodu będzie pisane przez AI i nadzorowane przez osoby rozumiejące kod
  - ChatGPT to fantastyczne narzędzie do uczenia się programowania - podpowie, wytłumaczy kod (!!!), napisze ładny kod, o wiele efektywniej niż stackoverflow.com
- » Przepisy AGH:
  - kod źródłowy wygenerowany przez model językowy nie jest waszą pracą (!!!) pismo COK-42-25/2023 Prorektora ds. kształcenia prof. dr hab. inż. Wojciech Łużny: "[pkt 6. Co do zasady, wszelkie materiały uzyskane przez generator treści \(tekst, algorytm, kod programu komputerowego, obraz itd.\) nie są uznawane jako autorskie dzieło osoby korzystającej z tego narzędzia. \(...\).](#)"
- » Korzystajcie z AI (zachęcam!) ale uczcie się programować - **musicie rozumieć kod źródłowy**, jeżeli nie to nie macie żadnej przewagi nad "humanami"
- » Ewaluacja będzie polegać tym jak rozumiecie i umiecie pisać kod: **WY** nie **AI**.

# Podstawowe pojęcia - informatyka

- » dyscyplina nauki zaliczana do nauk ścisłych oraz techniki zajmująca się przetwarzaniem informacji (wikipedia),
- » wybrane zagadnienia:
  - teoria informacji
  - algorytmika (tworzenie i analizowanie algorytmów)
  - języki programowania (projektowanie)
  - sprzęt komputerowy (projektowanie, budowa)
  - **programowanie** komputerów ([implementacja algorytmu w wybranym języku programowania na sprzęcie komputerowym](#))
  - budowanie systemów informatycznych **software** i **hardware**
  - inżynieria oprogramowania (informatyka+zarządzanie)
  - administracja sieciowa
  - grafika komputerowa, symulacja, architektura CPU, AI, webmastering,

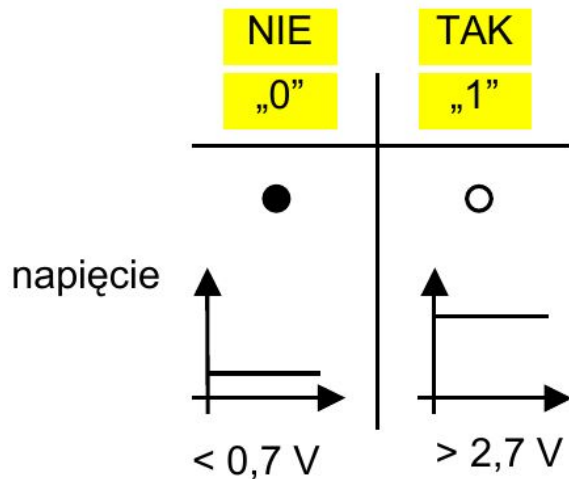
# Podstawowe pojęcia - **informacja**

**w kontekście informatyki**, używana w sensie obiektywnym

- » właściwość obiektu (niekoniecznie fizycznych)
  - “nr buta” (mojego), rozmiar pliku, kolor kredki, ...
- » wyróżniony stan obiektu (wskazany stan ze zbioru możliwych)
  - stan wyłącznika ściennego, **ON** lub **OFF**
- » można rozpatrywać informację w kontekście:
  - komunikacji (komunikat, szybkość komunikacji)
  - budowy (właściwości układu/obiektu)

# Reprezentacja informacji

- » większość komputerów jest cyfrowa
- » podstawowa reprezentacja informacji jest **binarna** (za tydzień teoria)



**sekwencje bitów:** ● ○ ○ ○ ● ● ● ○

**- liczby:**

*stałoprzecinkowe*

*zmiennoprzecinkowe*

**dane obliczeniowe** (wejście/wyjście)

**adresy**

**- znaki:**

alfanumeryczne (litery, cyfry, ...)

sterujące (spacja, backspace, ...)

**- instrukcje procesora**

# Reprezentacja informacji

x    xx    xxx    xxxx     $x=\{0,1\}$

--	---	----	-----
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
		1000	1000
		1001	1001
		1010	1010
		1011	1011
		1100	1100
		1101	1101
		1110	1110
		1111	1111

MSB                      LSB  
 0111001010001101

MSB = Most Significant Bit  
 LSB = Least Significant Bit



# Reprezentacja informacji

	x	xx	xxx	xxxx	$x=\{0,1\}$	liczba bitów	liczba stanów (komb.)
	--	---	----	-----			
0	00	000	0000				
1	01	001	0001		x	=2	
	10	010	0010		xx	=2*2	
	11	011	0011		xxx	=2*2*2	
		100	0100		xxxx	=2*2*2*2	
		101	0101				
		110	0110				
		111	0111				
			1000				
			1001				
			1010				
			1011				
			1100				
			1101				
			1110				
			1111				

# Reprezentacja informacji

	x	xx	xxx	xxxx	$x=\{0,1\}$	liczba bitów	liczba stanów (komb.)
	--	---	----	-----			
0	00	000	0000				
1	01	001	0001			x	=2
	10	010	0010			xx	=2*2
	11	011	0011			xxx	=2*2*2
		100	0100			xxxx	=2*2*2*2
		101	0101				
		110	0110				
		111	0111				
			1000				
			1001				
			1010				
			1011				
			1100				
			1101				
			1110				
			1111				

$$xx... (n) = 2^n$$

$$\mathbf{n\text{-}bitów = 2^n \text{ stanów}}$$

# Reprezentacja informacji

x	xx	xxx	xxxx	$x=\{0,1\}$	liczba bitów	liczba stanów (komb.)
0	00	000	0000			
1	01	001	0001		x	=2
	10	010	0010		xx	=2*2
	11	011	0011		xxx	=2*2*2
		100	0100		xxxx	=2*2*2*2
		101	0101			
		110	0110		xx... (n)	=2 <sup>n</sup>
		111	0111		<b>n-bitów</b>	<b>=2<sup>n</sup> stanów</b>
			1000		n=1	=2
			1001		n=2	=4
			1010		n=3	=8
			1011		n=4	=16
			1100		n=8	=256
			1101		n=16	=65536
			1110		n=32	=4294967296
			1111		n=64	=18446744073709551616

# Reprezentacja informacji

x	xx	xxx	xxxx	$x=\{0,1\}$	liczba bitów	liczba stanów (komb.)
0	00	000	0000			
1	01	001	0001		x	=2
	10	010	0010		xx	=2*2
	11	011	0011		xxx	=2*2*2
		100	0100		xxxx	=2*2*2*2
		101	0101			
		110	0110		xx... (n)	=2 <sup>n</sup>
		111	0111		<b>n-bitów</b>	<b>=2<sup>n</sup> stanów</b>
		1000				<b><math>n=\log_2(\text{liczba stanów})</math></b>
		1001			n=1	=2
		1010			n=2	=4
		1011			n=3	=8
		1100			n=4	=16
		1101			n=8	=256
		1110			n=16	=65536
		1111			n=32	=4294967296
					n=64	=18446744073709551616

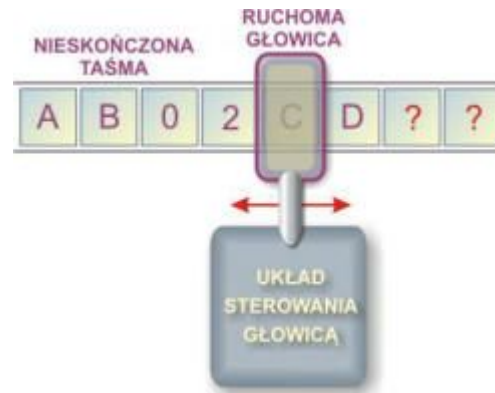
konwersja bin-dec-hex

*na następnych zajęciach*

# Jak jest zbudowany komputer?

# Maszyna Turinga

- » abstrakcyjny model komputera (Alan Turing 1936) służącego do wykonywania algorytmów
  - nieskończona taśma
  - głowica odczytująco/zapisująca
- » **taśma**: współczesna pamięć w komputerach
- » **głowica**: funkcja urządzenia we/wy
- » **układ sterowania**: procesor
- » ma znaczenie teoretyczne do dowodzenia twierdzeń
- » współczesne programy komputerowe dają się sprowadzić do maszyny Turinga



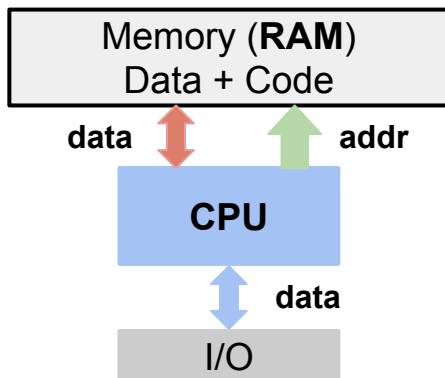
[http://eduinf.waw.pl/inf/prg/003\\_mt/0001.php](http://eduinf.waw.pl/inf/prg/003_mt/0001.php)

# Architektura komputera

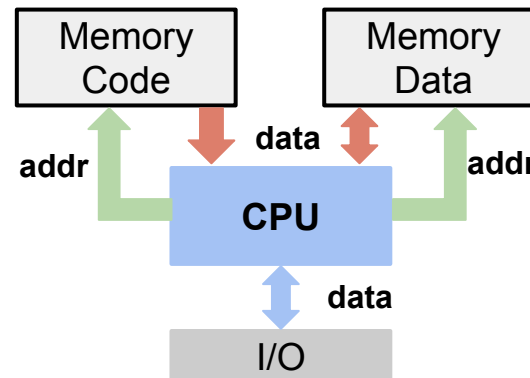
- » **komputer:**
  - maszyna (elektroniczna/cyfrowa) przeznaczona do przetwarzania informacji
  - **programowalna** (uniwersalna)
- » podstawowe elementy komputera:
  - CPU
  - RAM
  - I/O
- » architektura:
  - von Neumann
  - Harvard
  - mieszana



# von Neumann vs Harvard



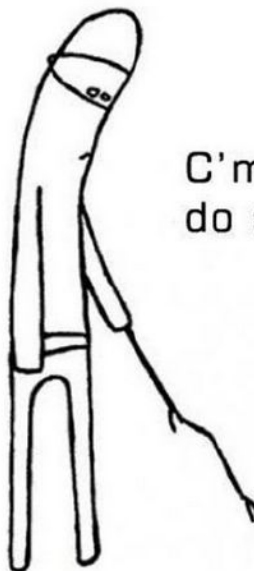
- » Jedna pamięć RAM, jedna magistrala - **taniej**
- » PC, serwery



- » Dwie pamięci, dwie magistrale: równoległy dostęp do danych i instrukcji (**szybciej**)
- » Kod chroniony przed zmianą
- » DSP, uC (krótki program)

# Jak mam “kazać” zrobić coś komputerowi?

# Jak mam “kazać” zrobić coś komputerowi?



C'mon,  
do something...

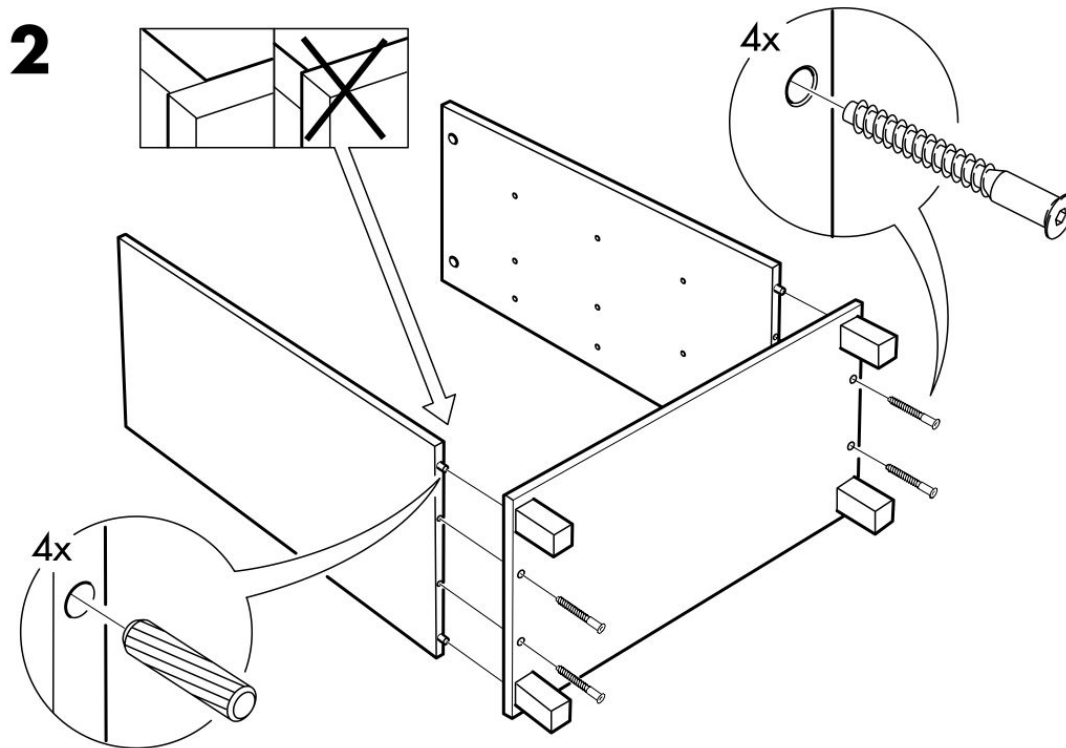
# Programowanie

- » **Programowanie** - proces projektowania, tworzenia (pisanie), testowania i utrzymywania kodu źródłowego programów
- » **Język programowania** to zasady (gramatyka) + instrukcje (słowa)
- » **Kod źródłowy** programu jest napisany w języku programowania
- » programowanie (kodowanie, ang. developing) jest **jednym z etapów** powstawania programu (inżynieria programowania)
  - zapisanie algorytmu w języku programowania
  - przetłumaczenie języka naturalnego na komputerowy

# Język programowania

- » **Język programowania** – zbiór zasad określających, kiedy ciąg symboli tworzy program komputerowy oraz jakie obliczenia opisuje  
(Mordechai Ben-Ari: *Understanding Programming Languages*)
- » Język programowania = składnia + instrukcje
  - jest językiem formalnym = **jednoznaczne reguły**
  - zapisany w postaci instrukcji, słów kluczowych zgodny z zasadami składni
  - kolejność zapisu instrukcji ma znaczenie

# “program” imperatywny



# Język programowania

- » Podział ze względu na cechy:
  - **funkcja**: służy do tworzenia programów komputerowych
  - **przeznaczenie**: wydawanie poleceń maszynom :-) ale też m2m
  - **konstrukcje składniowe**:
    - manipulowanie strukturami danych,
    - zarządzanie przepływem sterowania (kolejność wykonywania)
  - **moc**: zupełne w sensie Turinga, niezupełne (np. SQL)
- » HTML, XML - to nie są języki programowania (!obliczenia)

# Rodzaje języków - klasyfikacja

- » Poziom wykonania programu:
  - języki wysokiego poziomu (C/C++, C#, Java, Python, ...)
  - języki niskiego poziomu (assembler, Cg - GPU)
- » Sposób wykonania:
  - kompilowane (C/C++, Java\*)
  - interpretowane (JavaScript, Python, PHP, Perl, Matlab)
- » Podstawowe paradygmaty, programowanie/języki:
  - **imperatywne** (jakie instrukcje wykonać na danych aby osiągnąć cel)
  - funkcyjne (jak złożyć wyrażenia aby osiągnąć cel)
  - opisowe (dla jakiego stanu wejść i postaci systemu osiągnięty będzie cel)
  - logiczne (dowodem jakiego twierdzenia będzie oczekiwany rezultat?)



# Model programowania

- » liniowe (Basic)
- » proceduralne (COBOL, FORTRAN, BASIC, C)
- » **obiektywne** (C++, Java, Simula, Smalltalk, Python)
- » funkcyjne (Haskel, LISP)
- » stanowe (sterowniki PLC)
- » deklaratywne
- » logiczne
- » aspektowe
- » agentowe
- » \*\*\*: równoległość, bezpieczeństwo, szybkość pisania kodu, time-to-market :-/

# Programming languages

» TIOBE Index (September 2020)

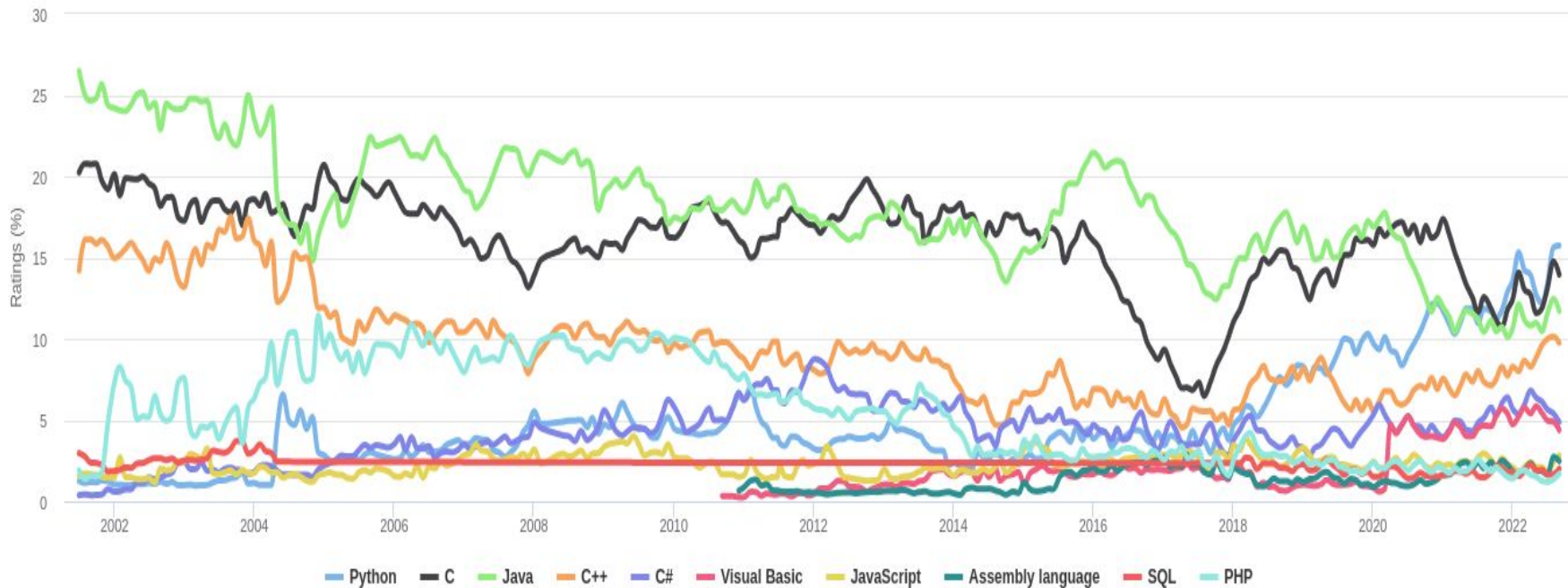
– C	11.8%	(-4.12%)
– Python	11.7%	(+1.20%)
– Java	11.1%	(-2.37%)
– C++	7.13%	(+0.01%)
– C#	5.78%	(+1.20%)
– VB	4.62%	(+0.50%)
– JS	2.55%	(+0.01%)

- » <https://www.tiobe.com/tiobe-index/> “TIOBE index is not about the best programming language or the language in which most lines of code have been written. The TIOBE Programming Community index is an indicator of the **popularity**”

# Programming languages

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# Dlaczego język X jest popularny?

- » <https://www.youtube.com/watch?v=QyJZzq0v7Z4>
- killer app
  - platform exclusivity
  - quick upgrade

# Pierwszy program C/C++

```
#include <iostream>
```

```
int main(){
```

```
    std::cout << "Hello world" << std::endl;
```

```
}
```

```
~/D/P/lab_02_fistCPP> g++ ex1.cpp -o ex1
```

```
~/D/P/lab_02_fistCPP> ls -al
```

```
razem 40
```

```
drwxrwxr-x 2 kwant kwant 4096 paź 7 18:33 ./
```

```
drwxrwxr-x 5 kwant kwant 4096 paź 7 18:29 ../
```

```
-rwxrwxr-x 1 kwant kwant 9216 paź 7 18:33 ex1*
```

```
-rw-rw-r-- 1 kwant kwant 76 paź 7 18:29 ex1.cpp
```

```
~/D/P/lab_02_fistCPP> ./ex1
```

```
Hello world
```

```
~/D/P/lab_02_fistCPP> █
```

```
#include <iostream>
```

```
int main(){
```

```
    std::cout << "Hello world" << std::endl;
```

```
}
```

» **#include <...>**

- instrukcja (dyrektywa) **preprocesora**

» **#include <iostream>**

- dołącza **plik nagłówkowy** “iostream”, w którym są definicje biblioteki we/wy

» **int main()**

- główna funkcja, uruchamiana automatycznie

» **{...}**

- nawiasy klamrowe: definiują blok kodu,

zasięg

» **std::cout**

- stdout, standardowy **strumień wyjściowy**

» **<<**

- operator wysłania (wstawienia w strumień)

» **“Hello world”**

- **stała** napisowa (ciąg znaków)

» **;**

- znacznik końca instrukcji (ang. statement)

```
#include <iostream>
```

```
int main(){  
    std::cout << "Hello world" << std::endl  
}
```

```
~/D/P/lab_02_fistCPP [1]> g++ ex2.cpp  
ex2.cpp: In function 'int main()':  
ex2.cpp:5:1: error: expected ';' before '}' token  
}  
^
```

- » kompilator podaje gdzie jest błąd, czasami bardzo dokładnie
- » błąd wystąpił w innym miejscu (innej linii)



```
#include <iostream>
```

```
int main()
```

```
    std::cout << "Hello world" << std::endl;
```

```
}
```

```
~/D/P/lab_02_fistCPP [1]> g++ ex2.cpp
ex2.cpp:4:2: error: expected initializer before 'std'
    std::cout << "Hello world\n";
    ^
ex2.cpp:5:1: error: expected declaration before '}' token
}
^
```

- » kompilator próbuje skompilować cały kod
- » jeden błąd może wywołać kaskadę błędów
  - czytać błędy chronologicznie, poprawiać pierwszy i kompilować

```
#include <stdio.h>
```

**język C**

```
int main(){  
    printf("%s\n", "Hello world");  
}
```

```
#include <iostream>
```

**język C++**

```
int main(){  
    std::cout << "Hello world" << std::endl;  
}
```



quiz

**PI01\_cout**

socrative.com

- login
- student login

Room name:

**KWANTAGH**

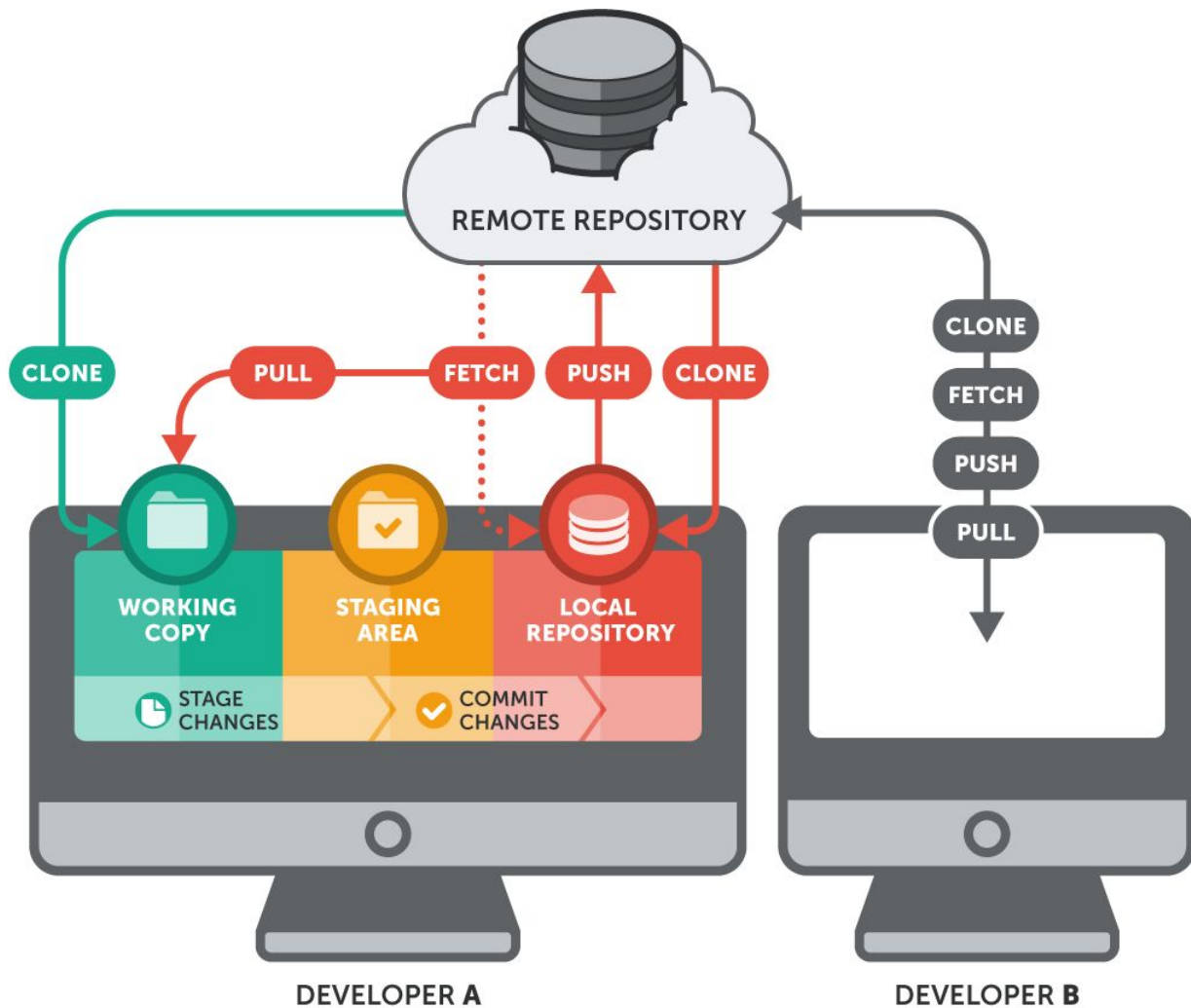
Gdzie mam zapisać \*.cc?  
w Git

# System kontroli wersji

- » Gdzie zapisać plik \*.cc? **Na dysku, w katalogu :-)**
  - jak go przesłać koledze-programiście?
  - jak efektywnie współdzielić z wieloma dev.?
  - jak zachować kolejne wersje?
- » VCS - Version Control System
  - **CSV** (Concurrent Versions System)
  - **SVN** (Subversion)
  - **GIT**

# System kontroli wersji

- » serwer (remote) przechowujący wszystkie wersje wszystkich developerów
- » lokalnie kopia serwera (na dysku)
- » funkcje:
  - wersjonowanie zmian
  - pamiętanie kto co zrobił
  - rozwiązywanie konfliktów (merge)
  - możliwość cofnięcia się do dowolnej wersji



# Literatura

## » Internet\*

- » cudzy kod (pryzwoity!)
- » <https://stackoverflow.com> C/C++, algorytmy, system, konfiguracja, ...
- » <http://www.cplusplus.com> C++, encyklopedia, rozróżnia wersje
- » <https://www.wikibooks.org> C/C++ (pryzwoite kompendium PL)
- » <http://cpp0x.pl/> C/C++ (niegłupie: dokumentacja + **kurs**)
- » **Bjarne Stroustrup**, *Język C++*, (zakłada znajomość C, PL)
- » Jerzy Grębosz, *Symfonia C++*, (od C do C++, popularne, przyzwoite)
- » Stephen Prata, *Język C++*
- » ~~Bruce Eckel, *Thinking in C++*~~



# AI/ML w informatyce

- » NLP (GPT-x) -> CodeX -> GitHub (MS) Copilot
  - turbo zaawansowany autocomplete
  - generuje body pętli/warunku/funkcji
  - generuje kod na podstawie komentarza (!)
  - generuje doc-string
  - rozumie kontekst kodu źródłowego (1-2k tokenów?)
  - implementuje best practice (nie pisze głupiego kodu)
  - trenowane na kodzie GitHub (dużo dobrego kodu)
  - podpowiada głupoty jak nie ma kontekstu (np. nie zna struktury dB)
  - tylko online, sekrety firmy?!?
  - problemy licencyjne kodu treningowego
  - zaakceptuj ten kierunek, wykorzystaj potencjalne możliwości albo odpuść karierę w dev/IT...



# MOOC /mu:k/



<https://coursera.org>

(Stanford, Princeton, ...)



<https://udacity.com>

(Georgia IofT, Google, Facebook, Nvidia, ...)



<https://sololearn.com>

(Android app.)

<reklama>



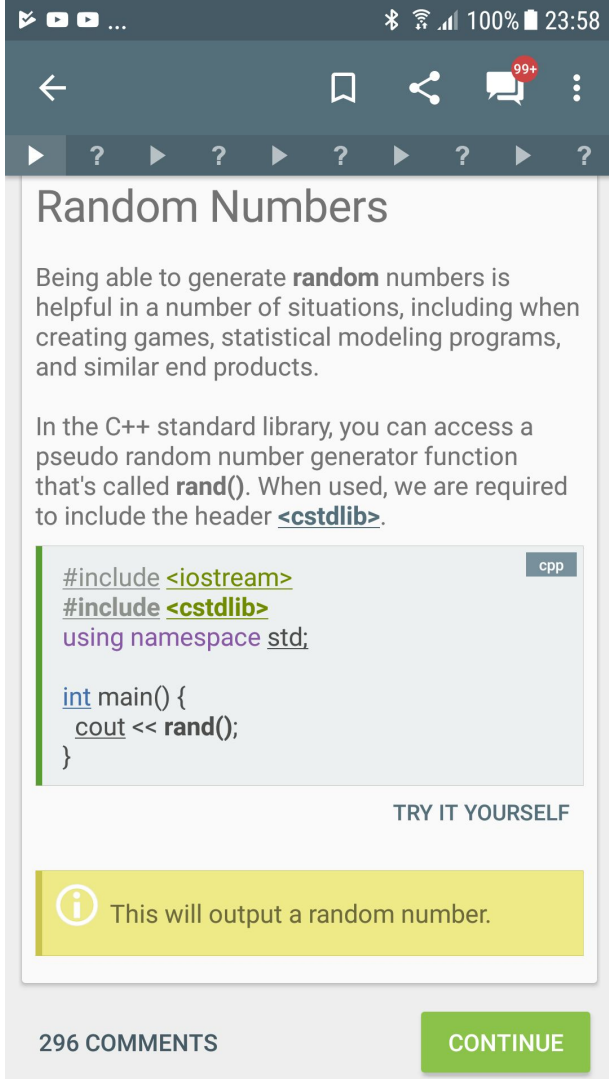
<https://www.sololearn.com>

</reklama>

# SoloLearn

**“Learn to code for FREE!  
Anytime and Anywhere, on Any Device“**

- » Krótkie lekcje z weryfikacją wiadomości (1-2 minuty)
- » Podstawowe wiadomości o języku i zasadach
- » Nie nauczysz się programować ale poznasz podstawy + wiele przydatnych wiadomości, odświeżysz wiadomości
- » Grywalizacja, rankingi, dyplomy, medale, etc...
- » Różne języki (C++, Python, HTML, etc...)
- » Nudne dla “zaawansowanych” - cały kurs C++ w kilka godzin
- » C++: błędy w “referencjach” i bezsensowne “wyjątki”, oprócz tego spoko-i-polecam



The screenshot shows the SoloLearn app interface. At the top, there's a status bar with icons for back, forward, and search, and a battery level of 100%. Below the status bar, there's a navigation bar with a back arrow, a bookmark icon, a share icon, a chat icon with a red notification bubble showing '99+', and a menu icon. The main content area is titled 'Random Numbers' and contains text explaining the importance of random numbers and how to generate them using the C++ standard library's `rand()` function. A code block shows the C++ code for generating a random number. Below the code block, there's a 'TRY IT YOURSELF' button. At the bottom, there's a yellow box with an information icon and the text 'This will output a random number.' The bottom of the screen shows '296 COMMENTS' and a green 'CONTINUE' button.

Random Numbers

Being able to generate **random** numbers is helpful in a number of situations, including when creating games, statistical modeling programs, and similar end products.

In the C++ standard library, you can access a pseudo random number generator function that's called **rand()**. When used, we are required to include the header `<cstdlib>`.

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
    cout << rand();
}
```

TRY IT YOURSELF

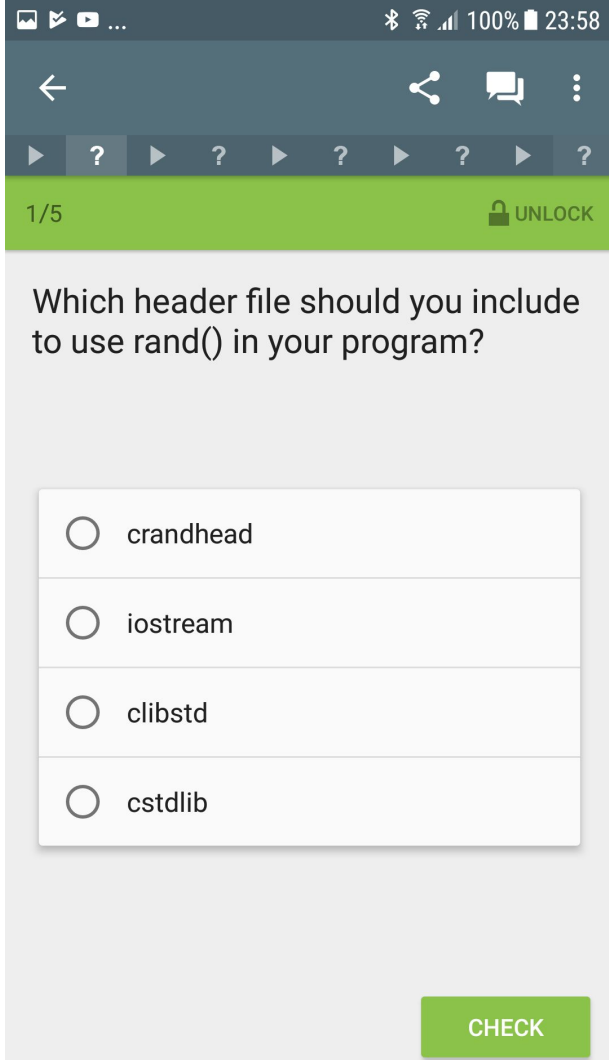
This will output a random number.

296 COMMENTS

CONTINUE

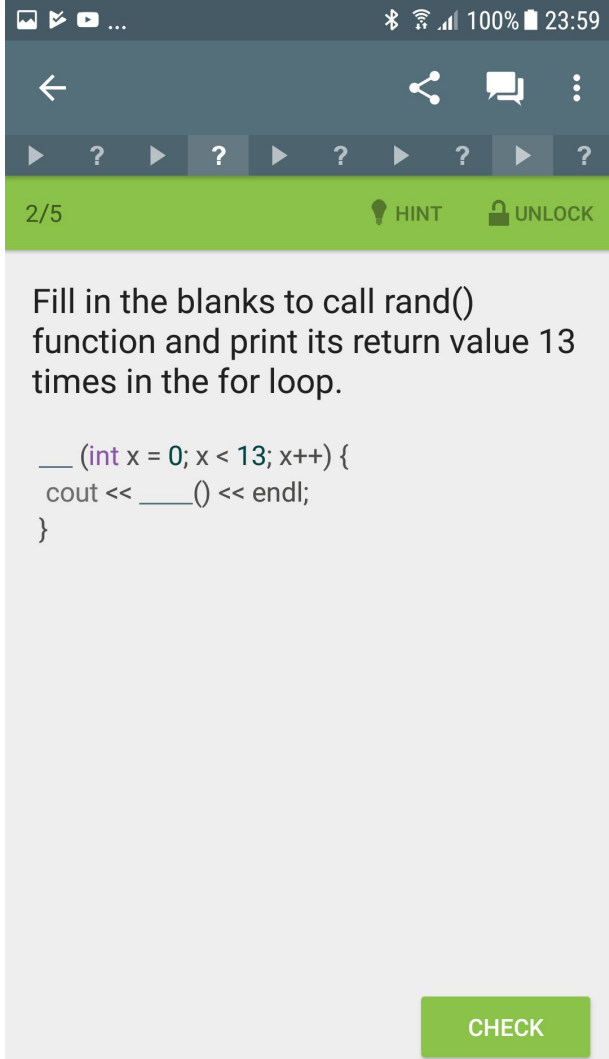
# SoloLearn

» Jedna lekcja - przeczytasz  
w 30 sekund



# SoloLearn

- » Jedna lekcja - przeczytasz w 30 sekund
- » Zaraz potem pytanie:
  - jeden z ....



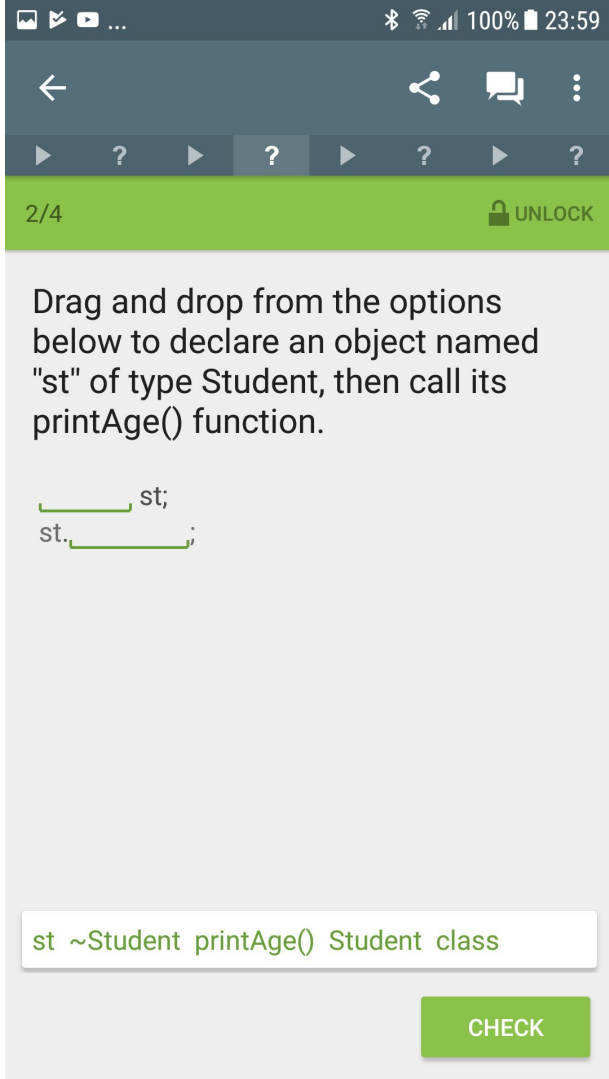
The screenshot shows the SoloLearn app interface. At the top, there's a status bar with icons for Bluetooth, Wi-Fi, cellular signal, 100% battery, and the time 23:59. Below that is a navigation bar with a back arrow, share icon, comment icon, and a menu icon. A progress bar shows 2/5 questions completed, with question markers (some with question marks, some with arrows). Below the progress bar are buttons for 'HINT' (lightbulb icon) and 'UNLOCK' (lock icon). The main content area contains the text: "Fill in the blanks to call rand() function and print its return value 13 times in the for loop." followed by a C++ code snippet: 

```
__ (int x = 0; x < 13; x++) {  
  cout << ____() << endl;  
}
```

 At the bottom right, there is a green 'CHECK' button.

# SoloLearn

- » Jedna lekcja - przeczytasz w 30 sekund
- » Zaraz potem pytanie:
  - jeden z ....
  - dopisz kod (warto mieć CodeBoard, Hacker's key..)



The screenshot shows the SoloLearn app interface. At the top, there's a status bar with icons for camera, gallery, and video, along with Bluetooth, Wi-Fi, cellular signal, 100% battery, and the time 23:59. Below this is a navigation bar with a back arrow, share icon, comment icon, and a menu icon. A progress bar shows 2/4 questions, with the current question highlighted. A green bar at the top of the question area says "2/4" and "UNLOCK". The question text is: "Drag and drop from the options below to declare an object named 'st' of type Student, then call its printAge() function." Below the text is a code editor with two lines: `_____ st;` and `st._____;`. At the bottom, there's a list of options: `st ~Student`, `printAge()`, `Student`, and `class`. A green "CHECK" button is at the bottom right.

# SoloLearn

- » Jedna lekcja - przeczytasz w 30 sekund
- » Zaraz potem pytanie:
  - jeden z ....
  - dopisz kod (warto mieć CodeBoard, Hacker's key..)
  - drag&drop



# SoloLearn

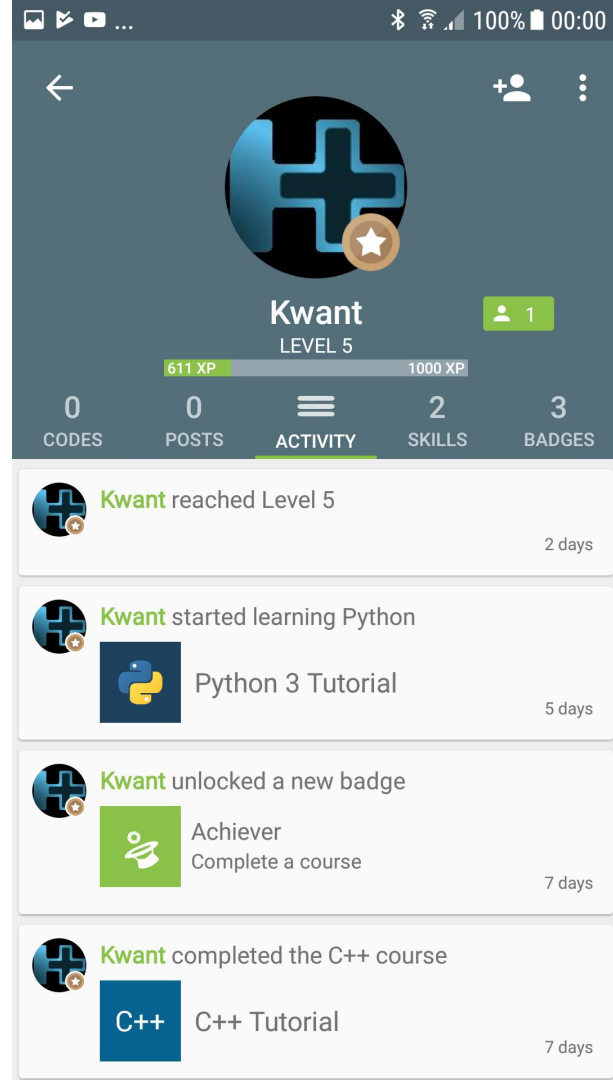
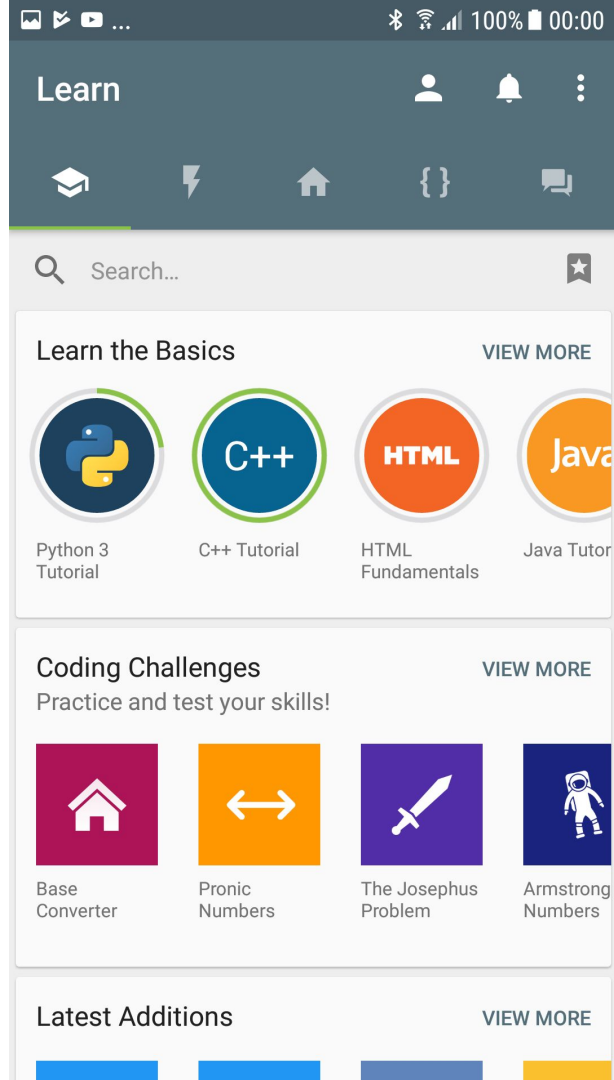
**“Learn to code for FREE!  
Anytime and Anywhere, on Any Device“**

- » Główna zaleta: **Anytime and Anywhere**
- » Zamiast tweetować/snapchatować o czekaniu na przystanku można rozwiązać 2-3 krótkie zadania
- » Płytkie: **nie nauczy cię to programować!!!**
- » Użyteczne: poznasz podstawową składnię i techniki
- » Challenge, Forum, **Andorid/iOS/WWW**

# SoloLearn

**Zamiast tracić czas na fejsie przypomnij sobie składnię  
języka albo zrób dwa “challenge”**

**W kolejce do kasy możesz uzyskać dwa punkty  
doświadczenia ;-)**



# zadanie domowe

- » znajdź jakiś komputer (może być PC), sprawdź ile ma:
  - pamięci operacyjnej (RAM)
  - pamięci masowej (HDD/SSD)
  - jak szybki jest CPU, ile FLOPS
  - jak szybko CPU komunikuje się z:
    - HDD
    - RAM
    - L1
    - L2
    - L3

# Załącznik

## informacja vs entropia

# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = - \sum_{i=1}^n p_i \log_r p_i$$

$H(X)$  - entropia bezwarunkowa zbioru  $X$

$n$  - liczb zdarzeń w zbiorze

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = - \sum_{i=1}^n p_i \log_r p_i$$

$H(X)$  - entropia bezwarunkowa zbioru  $X$

$n$  - liczb zdarzeń w zbiorze

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$x_i$

0

7

0

2

0

2

0

7

4

2

# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = - \sum_{i=1}^n p_i \log_r p_i$$

$H(X)$  - entropia bezwarunkowa zbioru  $X$

$n$  - liczb zdarzeń w zbiorze

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$x_i$

0

7

0

2

0

2

0

7

4

2



# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = - \sum_{i=1}^n p_i \log_r p_i$$

$H(X)$  - entropia bezwarunkowa zbioru  $X$

$n$  - liczb zdarzeń w zbiorze

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$x_i$

0

7

0

2

0

2

0

7

4

2

# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = -\sum_{i=1}^n p_i \log_r p_i$$

$H(X)$  - entropia bezwarunkowa zbioru  $X$

$n$  - liczb zdarzeń w zbiorze

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$x_i$

0

7

0

2

0

2

0

7

4

2

# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = -\sum_{i=1}^n p_i \log_r p_i$$

$p_i$        $x$

0.4      0

0.3      7

0.2      0

0.1      2

0

2

0

7

4

2

# Podstawowe pojęcia - informacja

miara ilości informacji:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

$I_i$  - to ilość otrzymanej informacji przy zajściu zdarzenia  $x_i$

$p_i$  - prawdopodobieństwo zajścia zdarzenia  $x_i$

$r$  - podstawa logarytmu

średnia ilość informacji (entropia):

$$H(X) = - \sum_{i=1}^n p_i \log_r p_i$$

$$-(0.1 \log_2(0.1) + 0.2 \log_2(0.2) + 0.3 \log_2(0.3) + 0.4 \log_2(0.4)) =$$

1.846...

= 2 bity

$p_i$        $x$

0.4      0

0.3      7

0.2      0

0.1      2

0

2

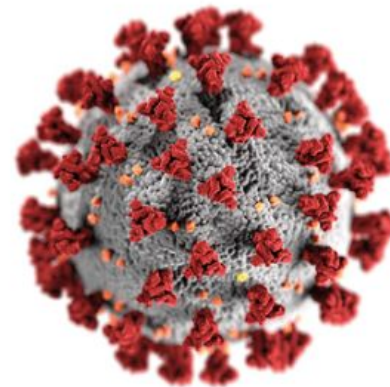
0

7

4

2

# COVID-19



## **ZARZĄDZENIE Nr 56/2021** Rektora Akademii Górniczo-Hutniczej

### **Regulacje dotyczące ochrony sanitarnej ludzi** **§5.**

1. Wszyscy członkowie Wspólnoty Uczelni: pracownicy, studenci i doktoranci oraz interesanci i goście, w trakcie przebywania w budynkach Uczelni winni stosować środki osobistej ochrony sanitarnej i wymogi dystansu:
  - a) maseczki zakrywające nos i usta,
  - b) środki chemiczne do dezynfekcji osobistej oraz przedmiotów,
  - c) w budynkach AGH obowiązuje zachowanie dystansu społecznego nie mniejszego niż 1,5 m.
2. W trakcie zajęć dydaktycznych, w trakcie spotkań i narad statutowych gremiów Uczelni oraz w trakcie spotkań lub zebrań służbowych lub zawodowych, określonych w § 1 ust. 3a, 3b i 3c możliwa jest modyfikacja zasad używania maseczek ochronnych:
  - a) z używania maseczek zwolniony jest prowadzący zajęcia oraz osoby wypowiadające się,
  - b) dla pozostałych osób używanie jest zalecane w szczególności, gdy nie jest możliwe zachowanie dystansu 1,5 m.

O zakresie odstępstw od używania maseczek decyduje prowadzący zajęcia lub spotkanie.

Dziękuję