

UNIVERZA V MARIBORU  
FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO  
Oddelek za matematiko in računalništvo

## **DIPLOMSKO DELO**

Gregor Nemeč

Maribor, 2016

UNIVERZA V MARIBORU  
FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO  
Oddelek za matematiko in računalništvo

## **DIPLOMSKO DELO**

# **Spletni portali za učenje programiranja: klasifikacija in možnosti uporabe v izobraževanj**

Mentor:  
doc. dr. Igor Pesek

Kandidat:  
Gregor Nemeč

Maribor, 2016

## ZAHVALA

*Citat*

*verz (možna opcija)(\*ustrezno spremeniti\*)*

*Zahvala...(\*ustrezno spremeniti\*)*

*Posebna hvala še...(\*ustrezno spremeniti\*)*

*Vsem iskreno hvala.(\*ustrezno spremeniti\*)*

UNIVERZA V MARIBORU  
FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

**IZJAVA**

Podpisani Gregor Nemeč, rojen 15. oktobra 1983, študent Fakultete za naravoslovje in matematiko Univerze v Mariboru, študijskega programa Fizike in Računalništva, izjavljam, da je diplomsko delo z naslovom

**Spletni portali za učenje programiranja: klasifikacija in možnosti uporabe v izobraževanj**

pri mentorju doc. dr Igor Pesku avtorsko delo. V diplomskem delu so uporabljeni viri in literatura korektno navedeni; teksti in druge oblike zapisov niso uporabljeni brez navedbe avtorjev.

Maribor, 2016

Gregor Nemeč  
(*podpis kandidata*)

**Nemec, G.: Spletни portali za učenje programiranja: klasifikacija in možnosti uporabe v izobraževanj. Diplomsko delo, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Oddelek za matematiko in računalništvo, 2016.**

## Povzetek

V zadnjem času so se pojavili številni spletni portali, ki ponujajo učenje programiranja. V diplomske delu nas zanima kateri so ti spletni portali in kako jih lahko umestimo v pouk v osnovnih in srednjih šolah. V ta namen smo pregledali spletne portale, ki so nastali v akademskem okolju ter smo določili kriterije s ki so bili glavno vodilo pri klasifikaciji in vrednotenju spletnih portalov. Novo nastalih portalov je precej zato smo izbiro omejili z kriteriji, kot je brezplačen dostop do gradiv in da so vrste vsebin napredno kombinirane. Pregled portalov je pokazal, da vsak izmed njih prinaša številne zmožnosti pri uporabi, izpostavili smo tudi nekatere slabosti. Z konkretnima primeroma pokažemo kako bi lahko uporabili spletni portal pri pouku.

Ključne besede: Učenje programiranja, spletni portali, programski jeziki, ...

**Nemec, G.: angleški naslov. Graduation Thesis, University of Maribor, Faculty of Natural Sciences and Mathematics, Department of Computer Science, 2016.**

## **Abstract**

Key words:

# Kazalo

<b>1 Uvod</b>	<b>1</b>
<b>2 Uporaba računalnika v izobraževanju</b>	<b>2</b>
2.1 Splošnoizobraževalno področje . . . . .	2
2.2 Strokovno izobraževalno področje . . . . .	3
2.3 Programiranje v OŠ . . . . .	4
2.3.1 Izbirni predmet računalništva . . . . .	4
2.3.2 Neobvezno izbirni predmet računalništva . . . . .	5
2.4 Programiranje v SŠ . . . . .	8
2.4.1 Informatika - Splošni gimnazijski program . . . . .	8
2.4.2 Računalništvo - Tehniška gimnazija . . . . .	9
<b>3 Računalniška znanost in programiranje</b>	<b>11</b>
3.1 Zgodovina programskih jezikov . . . . .	11
3.2 Osnovni pojmi . . . . .	12
3.2.1 Program . . . . .	12
3.2.2 Algoritem . . . . .	12
3.2.3 Programiranje in kodiranje . . . . .	13
3.2.4 Urejevalnik besedil . . . . .	14
3.2.5 Integrirano razvojno okolje . . . . .	14
3.3 Programske paradigmе . . . . .	15
3.3.1 Objektno orientirano programiranje . . . . .	15
3.4 Programski jeziki . . . . .	16
3.4.1 Java . . . . .	16
3.4.2 C++ . . . . .	17

3.4.3 Java Script . . . . .	18
3.4.4 Python . . . . .	19
3.5 Osnovni koncepti programiranja . . . . .	19
3.5.1 Izpis in računske operacije . . . . .	20
3.5.2 Spremenljivke . . . . .	21
3.5.3 Pogojni stavki in vejitve . . . . .	23
3.5.4 Zanke . . . . .	23
<b>4 Pristopi in strategije poučevanja</b>	<b>24</b>
4.1 Model aktivnega učenja . . . . .	24
4.2 Strategije reševanja problemov . . . . .	25
4.2.1 Razumevaje problema . . . . .	26
4.2.2 Načrtovanje rešitve . . . . .	26
4.2.3 Preverjanje rešitve . . . . .	27
4.2.4 Refleksija . . . . .	27
<b>5 Spletni portali za učenje programiranja</b>	<b>29</b>
5.1 Razlogi za nastanek spletnih portalov . . . . .	29
5.2 Primeri implementacije in sistemska arhitektura . . . . .	31
5.3 Pregled delovanja in interakcija s SPUP . . . . .	32
5.4 Rezultati izvedenih rešitev SPUP . . . . .	34
5.5 Značilnosti SPUP . . . . .	35
<b>6 Kriteriji za klasifikacijo spletnih portalov</b>	<b>36</b>
6.1 Vrsta vsebine . . . . .	36
6.1.1 Tekstovni vodiči . . . . .	36
6.1.2 Video vodiči . . . . .	37

6.1.3	Spletna aplikacija za programiranje . . . . .	37
6.1.4	Spletne igre . . . . .	38
6.1.5	Kombinirane vrste vsebin . . . . .	39
6.2	Jezik spletnega portala . . . . .	41
6.3	Ponujena znanja . . . . .	41
6.4	Programski jeziki . . . . .	41
6.5	Težavnostna stopnja . . . . .	41
6.6	Upoštevanj učnih načel . . . . .	42
6.7	Uporaba ocenjevanja dosežkov značilnih za igre . . . . .	42
6.8	Dodajanje lastnih vsebin . . . . .	43
6.9	Upravljanje razreda . . . . .	43
6.10	Dostop do gradiv . . . . .	43
<b>7</b>	<b>Pregled spletnih portalov</b>	<b>45</b>
7.1	Code.org . . . . .	45
7.1.1	Ura kode ( <i>ang. Hour of code</i> ) . . . . .	45
7.1.2	Code studio . . . . .	47
7.1.3	Samostojni projekti . . . . .	48
7.1.4	Uporaba za učitelje . . . . .	50
7.1.5	Povzetek . . . . .	50
7.2	Codeacademy . . . . .	51
7.2.1	Uporaba za učitelje . . . . .	54
7.2.2	Povzetek . . . . .	56
7.3	Scratch . . . . .	58
7.3.1	Uporaba Scratcha . . . . .	59
7.3.2	Deljenje in raziskovanje projektov . . . . .	60

7.3.3	Povzetek . . . . .	60
7.4	Repl.it . . . . .	61
7.4.1	Ustvarjanje razredov in nalog . . . . .	62
7.4.2	Povzetek . . . . .	62
7.5	Tutorialspoint . . . . .	63
7.5.1	Coding ground . . . . .	64
7.5.2	Povzetek . . . . .	66
7.6	Thimble . . . . .	66
7.6.1	Povzetek . . . . .	68
7.7	Code combat . . . . .	68
7.7.1	Upravljanje razreda . . . . .	73
7.7.2	Povzetek . . . . .	74
7.8	Codingame . . . . .	75
7.8.1	Povzetek . . . . .	78
<b>8</b>	<b>Možni načini uporabe spletnih portalov pri pouku</b>	<b>79</b>
8.1	Primer uresničevanja ciljev učnega načrta v osnovni šoli . . . . .	79
8.2	Primer uresničevanja ciljev učnega načrta v srednji šoli . . . . .	81
<b>9</b>	<b>Zaključek</b>	<b>83</b>

# Slike

1	Sistemska arhitektura spletnega portala za učenje programiranja, kot so jo naredili na OUHK [3]. . . . .	31
2	Prikaz med interakcijo študenta in mentorja s spletnim portalom [3]. . . . .	32
3	Zaslonski posnetki poglavja z vodiča <i>The Python Tutorial</i> s primerom [24]. . . . .	37
4	Uvodna stran enega izmed video vodičev za učenje pythona [26]. . . . .	38
5	Zaslonska slika spletne aplikacije za programiranje <i>Python Fiddle</i> [27]. . . . .	38
6	Zaslonska slika spletne strani Fightcode [31]. . . . .	39
7	Zaslonska slika spletne strani <i>w3School</i> [30] v poglavju HTML. . . . .	40
8	Zaslonska slika spletne strani <i>Codeschool</i> [32]. . . . .	40
9	Zaslonska slika dela začetne spletne strani <i>code.org</i> [37], iz katerega je razvidna razdeljenost vsebin. . . . .	46
10	Spletna aplikacija Code studio v kateri so zgrajene vadnice [37]. . . . .	46
11	Pod stran Code studio za katere lahko nadaljujemo na različne vsebinske sklope [38]. . . . .	47
12	Pod stran vsebinskega sklopa [38]. . . . .	48
13	Samostojni aplikaciji za programiranje primerni za osnovno šolo [38]. . . . .	49
14	Spletna aplikacija za programiranje - Ustvarjanje aplikacij. Primer, enostavno računalo [38]. . . . .	49
15	Zaslonska slika spletne strani <i>Codeacademy</i> [39]. Začetna, nadzorna stran po prijavi, od tu nadaljujemo na vsebinske sklope, ki smo jih že začeli. . . . .	51
16	Zaslonska slika spletne strani <i>Codeacademy</i> [39]. Seznam znanj/veščin programskih jezikov, ki jih ponuja spletni portal. . . . .	52
17	Zaslonska slika pod strani spletne strani <i>Codeacademy</i> [39] na kateri lahko pregledujemo posamezne teme in nadaljujemo tam, kjer smo ostali. . . . .	53
18	Zaslonska slika <i>Codeacademy</i> [39] vadnice, ki jo sestavlja urejevalnika z navodili in oknom za izpis v programu. . . . .	53

19	Plakat z povezavami enot različnih tematskih sklopov po stopnjah, ki vodijo do posameznih ciljev ( <i>ang. Level prograsion mapa</i> ) [39]. . . . .	55
20	Zaslonska slika <i>Codeacademy</i> [39]. Prikazuje tabelo za sledneje napredku učen-cem. . . . .	56
21	Zaslonski posnetek glavne strani <i>Scratch</i> [41]. . . . .	58
22	Zaslonska slika <i>Scratch</i> [41]. . . . .	59
23	Pod stran za deljenje projekta na <i>Scratch</i> [41]. . . . .	60
24	Zaslonska slika spletnne aplikacije za programiranje <i>repl.it</i> [43]. . . . .	61
25	Zaslonska slika pogleda mentorja v načinu priprave naloge [43]. . . . .	62
26	Del seznama oz knjižnica vodičev, ki ga ponuja spletna stran <i>Tutorials point</i> [45].	64
27	Zaslonski izrez vodiča za Python3. S like je razvidno kazalo in gumb za <b>Preizkus!</b> [45]. . . . .	64
28	Pod okno za preizkus primera programske kode [45]. . . . .	64
29	Del seznama različnih programskih jezikov katere lahko uporabljamo z spletno aplikacijo za programiranje <i>Codingground</i> [46]. . . . .	65
30	Spletna aplikacija za programiranje - <i>Codingground</i> [46]. . . . .	65
31	Urejanje projekta na strani <i>Thimble</i> [49]. . . . .	67
32	Izbira junaka in programskega jezika [47]. . . . .	69
33	Izbor zemljevida na katerem bomo reševali naloge [47]. . . . .	69
34	Podroben zemljevid za izbiro nalog [47]. . . . .	70
35	Oprema junaka in njen opis [47]. . . . .	70
36	Primerjava med prejšnjo različico opreme in njeno nadgradnjo, ki jo lahko zamenjamo junaku [47]. . . . .	71
37	Postavitev igre [47]. . . . .	72
38	Uspešno končan izid igre s napisano programsko kodo [47]. . . . .	72
39	Končni rezultat in pregled nad dobljenimi dostžki ob koncu igre[47]. . . . .	72
40	Učiteljev pogled na upravljanje razreda [47]. . . . .	73

41	Podstran <i>Codingame</i> [50] - sestavljanke. . . . .	76
42	Reševanje sestavljanke <i>The Bridge</i> [50] . . . . .	77
43	Primerjava programskih blokov napisanih v <b>Scratchu</b> in enačic klica metod napisanih v <b>Pythonu</b> na spletni strani <i>Codecombat</i> . . . . .	80
44	Miselni vzorec z povzetkom pregledanih spletnih portalov s predstavniki in značilnostmi za posamezno vrsto vsebine. . . . .	85

## Pregled uporabljenih simbolov in označb

1. **OŠ** - Osnovna šola;
2. **SŠ** - Srednja šola;
3. **IKT** - informacijsko komunikacijska tehnologija;
4. **IRO** - integrirano razvojno okolje (*ang. Integrated development environment IDE*);
5. **OUHK** - Odprta univerza Hong Kong;
6. **USVB** - Univerze Strathclyde Velika Britanija;
7. **QUTA** - Queensland University of Technology, Australia;
8. **OO** - objektno orientirano;
9. **SPUP** - spletni portal za učenje programiranja;
10. **SPZP** - spletna aplikacija za programiranje;
11. **AVP** - aplikacijski programski vmesnik (*ang. application programming interface API*);

# 1 Uvod

V svetovnem merilu se pojavlja trend po popularizaciji programiranja in kodiranja. V zadnjem obdobju so se na spletu pojavili številni portali, kot je na primer *Codeacademy*, ki ponujajo učenje programiranja. Zanima nas kateri so ti portali, katere vsebine ponujajo, na kakšen način so vsebine predstavljene in ali so uporabni za uporabo pri pouku ter na kakšen način bi jih lahko uporabili.

Najprej bomo raziskali kako je definirana računalniška pismenost in uporaba računalnika pri pouku. Ogledali si bomo kje se uči programiranja na osnovni (**OŠ**) in srednji šoli (**SŠ**). Pri katerih izbirnih vsebinah, predmetih in kakšna je vsebina, ki jo predvideva učni načrt. Podrobneje bomo preučili vse tiste pojme, ki se pojavljajo v računalniški znanosti in programiranju, ki nam bodo pomagali bolje razumeti spletnne portale in vsebino, ki jo predstavljajo. Preučili bodo tudi sodobne pristope in strategije, ki se uporabljajo pri učenju programiranja, saj bomo tako lažje ocenili ali jih spletni portali tudi znajo upoštevati.

Zanimali nas bodo **novinci** in njihove težave pri začetnih korakih učenja programiranja. Ko govorimo o novincih imamo v mislih vse tiste učence, dijake in študente, ki se šele srečujejo s programiranjem, ne glede na spol. Prav tako nas bodo zanimali **mentorji** s katerim imamo v mislih učitelje in profesorje.

V ta namen bomo pregledali nastanek spletnih portalov za namen učenja programiranja, ki so se pojavila v akademskem okolju na posameznih univerzah. Zanimal nas bo razlog za nastanek takšnih spletnih portalov na univerzah, zato bomo pregledali literaturo in poskusimo ugotoviti, zakaj in kako se na višje šolskem področju uporabljajo spletnne tehnologije za poučevanje programiranja in kako so skušale premostiti nekatere težave, ki jih imajo novinci pri učenju programiranja.

Izluščili bomo predlagane rešitve za uporabo spletnih tehnologij pri učenje programiranja. Na podlagi pregledanega bomo lahko določili kriterije s katerimi bomo lahko klasificirali spletnne portale in jih tudi ovrednotili ter uspešno umestili v pouk.

## 2 Uporaba računalnika v izobraževanju

Model uporabe računalnika v izobraževanju je Gerlič [1] razdelil na tri področja. V **primarno področje** lahko uvrstimo učenje računalništva in programiranja, saj sem prištevamo aktivnosti s katerimi želimo uporabnike seznaniti z delovanjem in uporabo računalnika oz. sodobno informacijsko komunikacijsko tehnologijo (**IKT**). Računalnik je tista učna vsebina, ki jo obravnavamo. **Sekundarno področje** predstavlja vse tiste aktivnosti, katere so vezane neposredno na izobraževalni proces katerega koli predmetnega področja. Računalnik in IKT nastopata kot učno sredstvo ali pripomoček v oblikah tradicionalnih računalniško podprtih učnih sistemov ali inteligentnih ekspertnih sistemov. V **terciarno področje**, spadajo vse aktivnosti, ki spremljajo izobraževanje. Sem se štejejo aktivnosti izobraževanja, vodenja in upravljanja izobraževalnega sistema.

V tem diplomskem delu nas bo zanimalo le **primarno področje** uporabe računalnika v izobraževanju, ki je razdeljeno v dveh pomembnih področjih [1]:

- kot element **splošne izobrazbe**,
- kot element **ožje strokovne** - poklicne izobrazbe oz. usposabljanja.

### 2.1 Splošnoizobraževalno področje

V današnjem času se računalnik kot element splošne izobrazbe kaže kot velika potreba oz. se zdi znanje njegove uporabe samoumevno. Že pri najmlajših otrocih računalnik vzbuja zanimanje in interes. Računalnik je postal intelektualno orodje in pripomoček v vsaki sferi človekove dejavnosti in je prodrl že dolgo nazaj tudi v šolo. Tako imenovana **računalniška pismenost** postaja nuja in zajema vse to kar bi človek moral znati o računalniku in to, kako je potrebno z njim delati, da bo uspešno živel v družbi, ki je osnovana na informacijah oz. informacijski družbi. V zvezi z definicijo in pojmovanjem **računalniške pismenosti** se kažeta dve usmeritvi, Gerlič [1] navaja številne avtorje obeh usmeritev in povzema:

Prva poudarja **sposobnost računalniškega programiranja** in opredeljuje s pojmom pismenosti sposobnost branja in pisanja podobno kot je to značilno za jezikovno pismenost. Tako zagovorniki, te smeri poudarjajo, da je cilj računalniške pismenosti, učenje in večina programiranja z novim načinom mišljenja in strategijami ugotavljanja in popravljanja računalniških programov.

Druga smer poudarja **splošno usposobljenost** za delo z računalnikom in to, da ni smiselno, da vsak kdo postane programer, zaradi tega, ker se bo računalnik uporabljal v najširšem smislu v praksi. Pomembno za učenca je da razume, delovanje računalnika in se zaveda njegovega vpliva na razvoj družbe. Učencu moramo pomagati, da dejanske probleme identificira in jih

lahko reši že z narejeno komercialno programsko opremo.

V zgodnjih letih sta bili značilni obe usmeritvi. Pozneje je prišlo do preobrata leta 1987 po mednarodnem simpoziju na Univerzi v Stanfordu. Eden od sklepov simpozija je bil ta, da se v splošnoizobraževalne programe, več ne uči programiranja, še posebej ne v strukturiranih verzij programskega jezika, kot je na primer **BASIC**, temveč naj se uči uslužnostne programske opreme, kot je urejevalnik besedil, orodja za delo z podatkovnimi bazami, grafična orodja itd .... Ta dejstva je Gerlič [1] povzel leta 2000 in je predlagal isto usmeritev z naslednjim navedkom:

*“Učencem vseh stopenj želimo ob čim večjem številu ur praktičnega dela z računalnikom, ob določenem problemu in ob uporabi ustreznih komercialnih programov seznaniti z osnovami računalništva in informatike.”*

Zanima nas koliko je računalniške znanosti in programiranja v splošnoizobraževalnem področju v učnih načrtih za **OŠ** in **SŠ**. Novi trendi in potrebe industrije kažejo na potrebo povečanja po znanjih programiranja. Ali zaradi razširjenosti računalniške tehnologije, ki je na vsakem koraku bi morda bilo potrebno definicijo računalniške znanosti, ki jo je povzel Gerlič posodobiti in ji dodati nazaj osnove znanje računalništva in programiranja ter uporabe odprto kodne programske opreme. Predvsem pri mlajših generacij obstaja potreba in trend, da se programiranje ponudi v širšem kontekstu tudi na splošnem izobraževalnem nivoju.

Ne glede na to, kako bo v prihodnje definirana računalniška pismenost, v diplomskem delu želimo pokazati, da spletni portali za učenje programiranja zaradi tehnološkega napredka omogočajo lažjo pot k učenju programiranja. Zato nas po pozneje zanimalo kje je umeščeno v učnem načrtu programiranje v **OŠ** in **SŠ**.

## 2.2 Strokovno izobraževalno področje

Sem prištevamo vse tiste aktivnosti, s katerimi želimo udeležence izobraževanja usposobiti na različnih ravneh tako, da se bodo ti z računalnikom in informacijskimi sistemi ukvarjali na poklicni ravni Lahko povemo, da sem spadajo vse ozko usmerjene računalniške in informacijske smeri srednjih šol, visokih, višjih ter univerzitetnih študijev. Zanima nas področje programiranja, kar je sicer predmet strokovnega izobraževanja, vendar nas bo zanimalo programiranje na splošnem izobraževalnem področju. Spletne portale za učenje programiranja bomo predstavili predvsem kot vstopno točko za začetnike in novice, katere seveda najdemo prav tako na vseh stopnjah.

## 2.3 Programiranje v OŠ

Pouk računalništva v OŠ poteka kot **izbirni predmet** ali kot **neobvezni izbirni predmet**. Za začetek bomo navedli kako je definirano Računalništvo v učnem načrtu za izbirne predmete v osnovni šoli [8]: “*Računalništvo je naravoslovno-tehnični izbirni predmet, pri katerem se spoznavanje in razumevanje osnovnih zakonitosti računalništva prepleta z metodami neposrednega dela z računalniki, kar odpira učencem in učenkam možnost, da pridobijo tista temeljna znanja računalniške pismenosti, ki so potrebna pri nadalnjem izobraževanju in vsakdanjem življenju.* ”.

### 2.3.1 Izbirni predmet računalništva

Izbirni predmeti računalništva so sestavljenih z treh predmetov in jih učenke in učenci lahko izberejo v tretjem trilerju, v 7., 8. in/ali 9. razredu. Prvi od teh predmetov je **Računalništvo - urejanje besedil**, kjer si učenci pridobijo osnovna znanja, ki so potrebna za razumevanje in temeljno uporabo računalnika. Naslednja dva predmeta sta **Računalniška omrežja** in **Multimedija**, kjer se ta znanja spiralno nadgradijo.

Zanima nas kje se pri teh predmetih računalništva pojavlja programiranje. Vsak izmed teh predmetov ima operativne učne cilje tako razdeljeno, da programiranje najdemo v tretji enoti, ki spada med dodatne vsebine. Posamezne operativne cilje, dejavnosti in vsebino prikazuje tabela 1.

Tabela 1: Operativni cilji, dejavnosti in vsebine izbirnega predmeta računalništvo za III. dodatno enoto. [8]

OPERATIVNI CILJI	DEJAVNOSTI	VSEBINE
<ul style="list-style-type: none"><li>• napisati algoritem z odločitvijo, ki reši preprost vsakdanji problem;</li><li>• izdelati in spremeniti računalniški program z odločitvijo.</li></ul>	<ul style="list-style-type: none"><li>• analizirati preprost problem;</li><li>• uporabljati osnovne korake programiranja.</li></ul>	<ul style="list-style-type: none"><li>• risanje diagrama poteka za problem z odločitvijo;</li><li>• izdelava računalniškega programa.</li></ul>

Programiranje se pojavlja le kot dodatna vsebine, kar se kaže v uresničitvi smeri računalništva, ki zagovarja **splošno usposobljenost**, kot smo jo opredelili v poglavju 2.1. Vendar se tu učiteljem računalništva ponuja izjemna priložnost, da z učenci in učenkami naredijo korak v osnove računalništva in programiranja pri vseh treh predmetih.

### 2.3.2 Neobvezno izbirni predmet računalništva

Opredelitev predmeta v učnem načrtu [9], pravi, da neobvezni izbirni predmet računalništva učence seznanja z različnimi področji računalništva, računalniškimi koncepti ter procesi in jih ne učijo dela z posameznimi programi. Učenci se seznanjajo tehniko in metodami reševanja problemov, razvijajo algoritmičen način razmišljanja. Opredelitev zadostuje prvi smeri računalniške pismenosti, ki zagovarja **sposobnost računalniškega programiranja..** Ker se v tej diplomski nalogi še posebej posvečamo programiranju nam učni načrt tega predmeta dosti bolj ustreza in si ga bomo podrobneje pogledali, saj nam bo pregled operativnih ciljev služil kot vodilo pri nadalnjem delu. Najprej preglejmo splošne cilje, katere povzemamo po učnem načrtu [9] in jih morajo doseči učenci:

- spoznavajo temeljne koncepte računalništva,
- razvijajo algoritmični način razmišljanja in spoznavajo strategije reševanja problemov,
- razvijajo sposobnost in odgovornost za sodelovanje v skupini ter si krepijo pozitivno samopodobo,
- pridobivajo sposobnost izbiranja najustreznejše poti za rešitev problema,
- spoznavajo omejitve človeških sposobnosti in umetne inteligence,
- se zavedajo omejitev računalniških tehnologij,
- pridobivajo zmožnost razdelitve problema na manjše probleme,
- se seznanjajo z abstrakcijo oz. poenostavljanjem,
- spoznavajo in razvijajo zmožnost modeliranja, strokovno terminologijo.
- razvijajo ustvarjalnost, natančnost in logično razmišljanje,
- razvijajo in bogatijo svoj jezikovni zaklad ter skrbijo za pravilno slovensko izražanje in strokovno terminologijo.

Neobvezni izbirni predmet je namenjen učencem 4., 5. in 6. razreda. V učnem načrtu so operativni cilji predstavljeni tako, da so temeljni označeni **krepko** in izbirni *poševno*. Navedli bomo tiste, ki so navezujejo na programiranje in strategije reševanja problemov. Operativni cilji so razdeljeni na vsebinske sklope.

#### Vsebina/sklop: Algoritmi

- **razumejojo pojmom algoritom,**
- **znajo vsakdanji problem opisati kot zaporedje korakov,**
- **znajo z algoritmom predstaviti preprosto opravilo,**
- **algoritom predstavijo simbolno (z diagramom poteka) ali s pomočjo navodil v preprostem jeziku,**
- **sledijo algoritmu, ki ga pripravi nekdo drug,**
- **znajo v algoritmu vključiti vejitev (če) in ponavljanje (zanke),**
- **znajo algoritom razgraditi na gradnike (podprograme),**

- **znajo povezati več algoritmov v celoto, ki reši neki problem,**
- *razumejo vlogo testiranja algoritma in vedo, da je testiranje orodje za iskanje napak in ne za potrjevanje pravilnosti,*
- *primerjajo več algoritmov za rešitev problema in znajo poiskati najustreznejšega glede na dana merila,*
- *znajo uporabiti nekatere ključne algoritme za sortiranje in iskanje,*
- *poznajo osnovne algoritme za iskanje podatkov.*

### Vsebina/sklop: Programi

- **znajo slediti izvajanju tujega programa,**
- **znajo algoritem zapisati s programom,**
- **znajo v program vključiti konstante in spremenljivke,**
- *razumejo različne podatkovne tipe in jih znajo uporabiti v programu,*
- *znajo spremenljivkam spremeniti vrednost s prireditvenim stavkom,*
- **znajo v programu prebrati vhodne podatke in jih vključiti v program,**
- **znajo izpisovati vrednosti spremenljivk med izvajanjem programa in izpisati končni rezultat,**
- **v program vključijo logične operatorje,**
- **znajo uporabiti pogojni stavek in izvesti vejitev,**
- **razumejo pojem zanke in ga znajo uporabiti za rešitev problema,**
- *razumejo kompleksnejše tipe podatkov (nizi, seznamy/tabele) in jih znajo uporabiti v programu,*
- **prepoznajo in znajo odpraviti napake v svojem programu,**
- *znajo popraviti napako v tujem programu,*
- *znajo spremeniti program, da dosežejo nov način delovanja programa,*
- *znajo rezultate naloge zapisati v datoteko,*
- **se seznanijo z dogodkovnim programiranjem,**
- **so zmožni grafične predstavitve scene (velikost objektov, ozadje, pozicioniranje),**
- **so zmožni sinhronizacije dialogov/zvokov,**
- **so zmožni razumeti in realizirati interakcije med liki in objekti,**
- **so zmožni ustvarjanja animacij.**

### Vsebina/sklop: Podatki

- **razlikujejo podatek in informacijo,**
- **razumejo dvojiški sistem zapisovanja različnih podatkov,**
- **razumejo kodiranje podatkov,**
- **razumejo, da obstajajo podatki v različnih pojavnih oblikah (besedilo, zvok, slike, video),**
- *poznajo načine predstavitev določenih podatkov in odnose med njimi (dvojiška drevesa*

*in grafi),*

- *vedo za stiskanje podatkov in vedo, da je stiskanje lahko brez izgub ali z izgubami,*
- **pojasnijo razliko med konstantami in spremenljivkami v programu,**
- *poznajo osnovne algoritme za iskanje podatkov,*

#### **Vsebina/sklop: Reševanje problemov**

- *znajo uporabiti različne strategije za reševanje problema,*
- **znajo našteti faze procesa reševanja problema,**
- *znajo postavljati vprašanja in ugotoviti, kateri podatki so znani,*
- *znajo za podano nalogu izluščiti bistvo problema,*
- **znajo najti ustrezno orodje, s katerim rešijo problem,**
- **znajo problem razdeliti na več manjših problemov,**
- **znajo načrtovati in realizirati rešitev,**
- *za podano rešitev znajo oceniti posledice in vpliv na "okolje",*
- *znajo uporabiti znano strategijo v novih okoliščinah,*
- *znajo ustvariti nov algoritem za bolj kompleksne probleme,*
- **znajo učinkovito sodelovati v skupini in rešiti problem z uporabo informacijsko-komunikacijske tehnologije znajo ceniti neuspešne poskuse reševanja problema kot del poti do rešitve,**
- *znajo kritično ovrednotiti rešitev in ugotoviti ali rešitev uspešno reši dani problem,*
- *znajo kritično ovrednotiti strategijo reševanja problema,*
- *zavedajo se omejitve informacijsko-komunikacijske tehnologije pri reševanju problemov.*

#### **Vsebina/sklop: Komunikacija in storitve**

- **znajo uporabiti ustrezna orodja in metode za iskanje po spletu,**
- **znajo uporabiti različne iskalne strategije v iskalnikih,**
- **poznajo omejitve pri rabi na spletu najdenih informacij (zavedajo se pojma intelektualna lastnina),**

Kot smo že povedali smo nekatere cilje izvzeli, čeprav je znanje računalniških omrežij pomembno, ga z vidika programiranja lahko izpustimo. Lahko povemo, da nam je večina operativnih ciljev ostala, saj smo jih lahko izključili le malo. Pridemo do spoznanja, da se v računalniški znanosti pretežen del znanja vrti okoli programiranja in reševanja problemov, kar je zajeto v tem učnem načrtu. Z samih ciljev lahko spoznamo tudi samo zahtevnost snovi. Lahko si upamo napovedati, da jih bomo s pravimi orodji kot so spletni portali za učenje programiranj tudi lažje dosegli.

## 2.4 Programiranje v SŠ

Pregledali bomo predmet **informatike**, splošnega Gimnazijskega programa in **Računalništvo Tehniške Gimnazije**.

### 2.4.1 Informatika - Splošni gimnazijski program

Predmet **informatike** se poučuje v prvem letniku splošne, klasične in strokovne Gimnazije. če povzamemo opredelitev učnega načrta [10], ta pravi naslednje. *"Informatika je splošnoizobraževalni predmet, pri katerem se teorija poznavanja in razumevanja osnovnih zakonitosti informatike prepleta z metodami neposrednega iskanja, zbiranja, hranjenja, vrednotenja, obdelave in uporabe podatkov z digitalno tehnologijo z namenom oblikovanja relevantnih informacij za dogajevanje lastnega znanja in za njegovo predstavitev oziroma posredovanje drugim."*

Poučevanje Informatike obsega *70ur* v 1.letniku, izbirni predmet obsega *210ur* in maturitetni predmet ima *70 + 210ur*, pri čemer je *70ur* namenjenih projektnemu delu.

Cilji vsebine predmeta so razporejeni na dve ravni:

- **splošna znanja**, v katerih dijaki razvijajo temeljnje digitalne kompetence, ki so potrebne za učinkovito uporabo digitalne tehnologije pri razvijanju lastnega znanja in za njegovo predstavitev oziroma posredovanje drugim;
- **posebna znanja**, s katerimi dijaki znanje, veštine, osebnostne in vedenjske značilnosti, prepričanja, motive in druge zmožnosti splošnega znanja spiralno nadgradijo.

Za nas pomembne enote najdemo na drugi ravni, **posebnih znanj** v tematskem sklopu **obdelava podatkov**. Zbrali bomo cilje, kateri izpostavljajo programiranje. Dijaki:

- **Računalniška obdelava podatkov**
  - poznajo vlogo računalniškega programa in razložijo pomen programiranja;
- **Algoritmi**
  - opredelijo algoritem in poznajo temeljne zahteve za algoritmom,
  - poznajo temeljne gradnike algoritma, razvijejo algoritom za problem z vejiščem in zanko (do 15 gradnikov),
  - uporabijo diagram poteka in uporabljeno rešitev utemeljijo,
  - analizirajo algoritmom, ki reši zahtevnejši problem, in ga ovrednotijo;
- **Programski jezik**
  - opredelijo programski jezik in razložijo njegovo funkcijo,
  - poznajo temeljne gradnike izbranega programskega jezika,
  - razložijo njihovo funkcijo in razlagajo ponazorijo s primeri,

- opredelijo strukturirano, objektno in dogodkovno programiranje,
  - ločijo med prevajalnikom in tolmačem in razliko razložijo;
- **Programiranje**
    - za dani algoritem izdelajo računalniški program,
    - opredelijo dokumentiranje programa in razložijo njegov pomen,
    - analizirajo program in ovrednotijo rezultate, dobljene s programsko rešitvijo;

#### 2.4.2 Računalništvo - Tehniška gimnazija

Cilje učenja programiranja pri tem maturitetnem predmetu najdemo pod poglavjem **Programski jeziki in programiranje**. Cilji so naslednji, dijaki [11]:

- poznajo pomen načrtovanja in sistematične gradnje programa,
  - poznajo pojem algoritma in opredelijo njegove lastnosti (razumljivost, končnost, enoumnost, razčlenjenost),
  - opišejo načine zapisa algoritma in jih prikažejo na primeru, izdelajo algoritem za lažji problem,
  - opredelijo pojem programskega jezika in razložijo njegovo funkcijo,
  - poznajo različne vrste programskih jezikov in jih razvrstijo po namenu in uporabi,
  - poznajo temeljne gradnike postopkovnega programskega jezika, razložijo njihove funkcije in razlago ponazorijo s primeri (zaporedje, vejitev, iteracija),
  - opredelijo strukturirano in objektno programiranje,
  - ločijo med prevajanjem in tolmačenjem in razliko razložijo,
  - poznajo osnovno razvojno okolje,
  - poznajo pojme deklaracija, inicializacija, postopek, konstanta, spremenljivka, rezervirana beseda, operator, prioriteta,...
  - ločijo med pojmi izvorna koda, izvršljiva koda in vmesna (byte) koda,
  - pridobljeno znanje o tipih in algoritmih prenesejo v programski jezik in samostojno rešujejo probleme s pomočjo višjega programskega jezika Java,
  - poznajo in uporabijo osnovne in sestavljeni tipe podatkov,
  - uporabijo pogojna stavka (if, switch),
  - iteracijo realizirajo z zankami (do, for, while),
  - uporabijo tokove podatkov,
  - pripravijo testne podatke, testirajo delovanje programa in beležijo rezultate testiranj,
  - uporabijo razhroščevalnik,
  - napišejo sled programa,
  - izdelajo dokumentacijo programa,
  - uporabijo metode za delo z objekti razredov Math, String, StringBuffer, Integer, Double
- ...

- napišejo definicijo razreda(lastnosti, metode ...),
- deklarirajo in uporabijo objekte,
- poznajo vlogo in način izvajanja konstruktorja,
- uporabijo enkapsulacijo, dedovanje in polimorfizem,
- poznajo načine za prestrezanje in obravnavo izjem,
- napišejo programe za enostavne probleme iz okolja,
- analizirajo program in ovrednotijo rezultate, dobljene s programsko rešitvijo (različni
- algoritmi urejanja podatkov, različni načini zapisovanja podatkov v datoteke ...),
- predstavijo delovanje programa,
- analizirajo in kritično vrednotijo rešitve,
- *poznajo zahtevnejše tehnike programiranja,*
- *napišejo program z uporabo zahtevnejših tehnik programiranja.*

Cilji določajo zelo podrobno obravnavo snovi programskih jezikov in programiranja, ki na prvi pogled presega uporabo spletnih portalov za učenja programiranja.

### 3 Računalniška znanost in programiranje

Računalniška znanost ima številne različice definicij, v grobem jo lahko definicijo strnemo v naslednjih trditvah [5].

- Ukvarja z značilnostmi tisktega kar je izračunljivo.
- Je znanost, ki izhaja iz več področij in ime korenine v matematiki, znanosti in inženirstvu.
- Ima mnoga podpodročja in je interdisiplinarna z biologijo, ekonomijo, medicino, zavavo.
- Ime računalništvo ali računalniška znanost nas lahko tudi zavede in jo zamenjamo z področjem uporabe računalnika.

#### 3.1 Zgodovina programskih jezikov

Uporaba računalništva v izobraževanju je bila deležna številnih sprememb. Sama uporaba računalnika v izobraževanju je tesno povezana z razvojem računalnikov. Začetno obdobje, 1960 letih prejšnjega stoletja so računalniki bili zelo dragi in veliki glavni računalniki (*ang. mainframe*), na njih se je učilo programiranja, a so se uporabljali tudi za druga področja. V tem obdobju se je za učenje programiranja uporabljal **FORTRAN** ali **asembler**. Programi so bili majhi in enostavi, zaradi fizičnih omejitev takratnega delovnega pomnilnika. Temu začetnemu obdobju pravimo **terminalsko obdobje** [1].

V 1970 so na trg prišli manjši računalniki, ki so bili tudi cenejši in zmogljivejši. V tem času pride v ospredje strukturirano programiranje. Najpopularnejši programski jezik je bil **PASCAL**. Predstavnimi obdobja **mikrorračunalnikov** so bili *Commodore 64, Sinclair ZX-81, Apple II*.

V 1980 so se prvič pojavili **samostojni osebni računalniki**. Programski jeziki v tem obdobju so bili strukturirani in močnega tipa (*ang. strong type*). Med te spada **Ada, Modul 2, ML** in **naj omenimo še Prolog**.

V naslednjem desetletji, 1990 so v ospredje prišli objektno orijentirani programski jeziki, kot sta **C++** in **JAVA** [7].

Metode poučevanja računalništva so se prav tako spreminali. 1960 so računalnike uporabljali samo za poučevanje programiranja. Povdarek pri predmetih programiranja je bil predvsem na detaljnih zmožnosti programskega jezika. Programiranje je bilo omejeno le na reševanje enostavnih primerov in poudarka na splošnem reševanju problemov ni bilo.

V 1970 je reševanje problemov in abstrakcija podatkov postala glavni in najpomembnejši del vseh programerskih predmetov, kar velja še danes. Programi so postali večji, bolj interaktivni

in spremenil se je vnos podatkov z tekstovnega v grafičnega. Vsebina predmetov računalništva se je hitro razširjala, kakor so se množili številni programski jeziki [7].

Posebno mesto v izobraževanju ima programski jezik **LOGO**. Z ekipo ga je Seymour Papert. Značilnost programskega jezika je ta, da z programsko kodo rišemo grafiko, ki jo lahko predstavimo na zaslonu ali z "z želvo", ki se premika po tleh z ukazi in riše sliko na podlago. Programski jezik je bil prvič zasnovan v namene učenja programiranja.

## 3.2 Osnovni pojmi

Preden nadaljujemo moramo razjasniti nekaj osnovnih pojmov, ki se pojavljajo računalniški znanosti.

### 3.2.1 Program

Računalniški program je zbirka navodil, ki opravlja točno določeno nalogu in jo izvajamo na računalniku. **Centralno procesna enota** je tista, ki izvajanje programa omogoča. Računalniški program navadno napiše **programer** v nekem **programskem jeziku**, postopku pisanja programa pravimo **programiranje**. Programska jezikom omogoča, da je program zapisan v takšni obliki, da je berljiv za ljudi in je zapisan v **izvorni kodi**. Da računalnik razume napisan program, ga prevede **prevajalnik** (*ang. compiler*) v **strojno kodo** ali ga **tolmači** tolmač. [12].

### 3.2.2 Algoritem

Z besedo algoritmom ponazarjamо postopek, ki je zgrajen iz posameznih operacij, ki se izvajajo po posameznih korakih. Algoritmom daje rešitev za izračune, procesiranje podatkov, avtomatizacijo postopkov. Ime besed "algoritmom" prihaja z imena **Al-Khwārizmī**, Perzijskega matematika, astronoma, geografa in učenjaka [13].

Pri pregedu učnih načrtov v poglavju 2.3.2 smo lahko zasledili cilje kot so:

- **znajo vsakdanji problem opisati kot zaporedje korakov,**
- **znajo z algoritmom predstaviti preprosto opravilo,**
- **algoritmom predstavijo simbolno (z diagramom poteka) ali s pomočjo navodil v preprostem jeziku.**

Prikazali bomo več različnih predstavitev algoritma, najprej bomo rešitev zapisali v besedilni obliki, zatem bomo rešitev podali z diagramom poteka, na koncu bomo rešitev podali še z programskim jezikom Scratch in Python. Za primer bomo prikazali primer algoritma, ki reši

naslednjo nalogu.

### Primer 1: Algoritem - Najdi največje število

Algoritem med podanimi števili poišče največje število.

Postopek zapisan v **besedilu**:

1. Zapišemo si začetno število, naj bo 0.
2. Števila v seznamu preglejujemo po vrsti.
3. Vsakič, ko najdemo večje število, ko je naše začasno število,
4. To začasno število prečrtamo in napišemo tisto z seznamoa, ki je večje.
5. Postopek ponavljamo, dokler ne pridemo na konec seznama.

Podprogram v **Scratchu**:



Podprogram v **Pythonu**:

```
1 def najdiNajvecjeStevilo(seznam):
2     """Funkcija poišče najvecje stevilo v seznamu."""
3     i = 0
4     najvecje_stevilo = 0
5     for i in range(len(seznam)):
6         if seznam[i] > najvecje_stevilo:
7             najvecje_stevilo = seznam[i]
8     return najvecje_stevilo
```

### 3.2.3 Programiranje in kodiranje

Iraz programiranja smo že spoznali. Veliko krat slišimo tudi izraz, "**kodiranje**" (*ang. coding*). Povedali bi lahko da izraz pomeni, pisanje programske kode, konček izdelek, tisto kar na koncu damo **prevajalniku**, kar je **izvorna koda**, da prevede v **strojno kodo**, katero lahko potem zaganjam.

Za vsakim programiranjem stoji seveda nek programer, lahko bi rekli, da je za vsakim kodira-

njem nekdo, ki mu pravimo **koder**. Programer in koder stav veliko krat, dana v isti koš, vendar si to čisto ne zaslužita, saj je programer, nekdo ki načrtuje rešitve, na različne načine in z različnimi orodji preden sploh zapiše kaj programske kode. Koder po drugi strani je tista oseba, ki se dobro spozna na programske jezike vendar, dela veliko krat po načrtu programera, je tisti ki na kocu zapiše rešitve in je pri tem zelo unčikovita. Čeprav se ta dva poklica zelo povezana in so meje med njima tudi veliko krat zbrisane sploh, če je programer in koder ena in ista oseba [14].

Pri samem poučevanju računalništva, lahko menimo, da je v prvi vrsti pomembno to, da se učimo strategije reševanja problemov, mora obstajati težnja kako poučevati, da bo čim več učencev postalo dobrih programerjev in samo koderjev.

### 3.2.4 Urejevalnik besedil

Dober urejevalnik besedi lahko programerju lajša delo s številnimi zmožnostmi. Opisali smo nekatere, saj nam bodo te pomagale lažje prepoznati dober urejevalnik besedil.

- **Barvanje rezerviranih besed prog. kode.** je značilno za skoraj vsak novo dobni urejevalnik besedil. Z različnimi barvami je olajšano branje programske kode.
- **Samodejno zamikanje vrstic programske kode.** Urejevalnik, ki ima to zmožnost zna prepoznati, ko sledi nov del programske kode, ki je zamknjen npr. za stavkom `if/else`.
- **Ponujanje predlog za samo dokončevanje rezerviranih besed in funkcij prog. jezika.** Ob pisanju programske kode urejevalnik ponuja dokončevanje programske kode.
- **Izpis opisa funkcij z atributi.**
- **Samodejno zaključevanje oklepajev.** Odprtji oklepaj se samodejno zaključi z zaprtim in smernik za pisanje se postavi v sredino med oba oklepaja.

### 3.2.5 Integrirano razvojno okolje

Del **integriranega razvojnega okolja (IRO)** (*ang. Integrated development environment IDE*) je dober urejevalnik besedil, kot smo ga opisali v prejšnjem poglavju. Za IRO je značilno, da omogoča **pisanje, testiranje, razhroščevanje in prevajanje končnega programa**. Omočajo povezavo med datotekami projekta in knjižnicami. Svoje zmožnosti nadgrajujejo s številnimi zunanjimi orodji, ki jih navadno vklopimo preko vtičnikov. Slabost IRO je ta da so veliki programski paketi, ki včasih znajo biti okorni in počasni. Pri nekaterih obsežnih IRO je tudi čas, ki ga potrebujemo, da se ga naučimo uporabljalni dolgotrajen.

### 3.3 Programske paradigmme

Paradigma je način kako obravnavamo in gledamo na stvari, je okvir v katerem leži naša interpretacija realnosti sveta. Paradigma najpogosteje pomeni vzorec delovanja v znanstvenem ali drugem raziskovanju. Izraz programske paradigmme je več pomenka, ki povzema mentalne procese, strategije reševanja problemov, povezave med različnimi paradigmami, programske jezike, stil programiranja in še več [17] [5].

Programske paradigmme so hevristike, ki se uporablajo za reševanje problemov. Programska paradigma analizira problem, čez specifičen pogled in na ta način formulira rešitev za dani problem, ki ga razdeli na manjše dele med katerimi definira razmerja.

Programske paradigmme so na primer proceduralno, objektno orientirano, funkcionalno, logično in istočasno programiranje. V nadaljevanju spoznamo značilnosti programske paradigmme objektnega programiranja, saj je ta v zadnjih dveh desetletjih najbolj razširjena.

#### 3.3.1 Objektno orientirano programiranje

V objetno orientirano **OO** programiranje je način kako programerji razmišljajo o svojem delu. Princip OO model realnosti sveta predstavlja v **razredih ang. class**. Z takim načino zapisa programske kode je ramišljjanje o programu dosti bolj naravno [15].

Objekt je predstavnik različnih stvari, ki jih želimo predstavljati v programske razredu. Te stvari so lahko kar koli, od realnih objektov in vse do konceptov. Podajmo primer objekta mačke. Mačka ima številne karakteristike, kot so barva, ime, teža ..., tem lastnostim pravimo da so lastnosti objekta. Mačka je živo bitje, zato njenemu početju, kot je mjavkanje, spanje, igranje ..., pravimo, da so metode objekta. Pri objetih lahko uporabimo analogijo in objekte poimenujemo z samostalniki, metode so glagoli in vrednosti lastnosti objekta so pridevniki. V nadaljevanju si bomo ogledali nekatere značilnosti, ki definirajo programski jezi kot OO [16].

**Razred (ang. Class):** V resničnem življenju lahko objekte združujemo po nekih določenih kriterijih. Orel in sinička sta oba ptiča, zato jih lahko damo skupaj v razred katerega poimenujemo Ptiči. Razredi so načrti ali recepti za objekte, tako lahko ustvarimo več objektov iz istega razreda, saj je razred le shema.

**Enkapsulacij (ang. Encapsulation):** je koncept, ki predstavlja, da so podatki, torej *lastnosti* objektov in opravila, ki jih lahko opravljajo ali *metode* objektov, združeni.

**Združevanje (ang. Aggregation):** pomeni, da lahko več objektov združimo v en objekt. To predstavlja močno orodje pri razčlenjevanju problemov na manjše pod probleme.

**Dedovanje (ang. Inheritance):** je eleganten način kako porabimo eno kodo večkrat. Podajmo primer, imamo splošen razred Oseba, ta ima lastnosti kot je ime, datum rojstva in

ima napisane metode, ki prestavlja funkcionalnost kot je, da Oseba lahko govori, hodi, je, spi. Zatem bi želeli bolj specifičen razred ko je Programer. Lahko bi vso kodo ponovno napisali in ji dodali specifično za programerja. Dedovanje omogoča, da povemo da Programer deduje od razreda Osebe in si tako prihranimo velik del dela.

**Polimorfizem (ang. polymorphism):** je način kako lahko isto ime metode uporablja več različnih razredov in posledično objektov ne glede nato, da je najverjetnejše koda v njem različna.

Programsko paradigma OO programiranja smo povzeli na kratko, da bi lažje razumeli zakaj je ta programska paradigmata tako popularna. V nadaljevanju bomo spoznali, da je večina programskih jezikov, ki se uporabljajo danes OO ali vsaj vsebijejo nekaj lastnosti OO programskih jezikov.

## 3.4 Programske jeziki

V tem poglavju bomo povzeli osnovne značilnosti posameznih programskih jezikov. Če na spletu v spletnem iskalniku podamo zahtevo po najpopularnejših programskih jezik, dobimo podobne rezultate večih spletnih strani<sup>1</sup> <sup>23</sup> in sicer: **JAVA, C, C++, Python, C#** v top 10 za nas pomembne najdemo še **Java Script**.

Programski jeziki v izobraževanju so se skozi zgodovino menjavali, tako kot se je rezvijala računalniška znanost, kar smo že povzeli v poglavju 3.1. Zanima nas, kateri so programski jeziki, ki so najbolj primerni za uporabo učenja programiranja in se uporabljajo danes.

Vsak programski jezik bomo z kratim primerom tudi predstavili z primerom programske, kode tako bomo dobili lažjo predstavo kakšna je osnovna razlika v sintaksi.

Večina od zgoraj naštetih programskih jezikov je **OO** razen izjeme **C**-ja, ki je predhodnik **C++**. Za nas bodo z izobraževalnega vidika zanimivi predvsme tisti, ki se uporabljajo pri splošnem izobraževanju pri nas.

### 3.4.1 Java

Java je več namenski programski jezik, njegova osnov so *razredi* in je OO. Njegova glavna prednost je, da napisano kodo lahko zaganjamо ne glede na platformo na kateri teče, torej so napisani programi neodvisni od operacijskega sistema, ki ga poganja računalnik. Zato je za vsako platformo prilagojen **virtualni stroj za Java** (ang. *Java Virtual Machine (JVM)*), ki

---

<sup>1</sup>Pridobljeno 27.04.2016 iz, [http://www.tiobe.com/tiobe\\_index](http://www.tiobe.com/tiobe_index).

<sup>2</sup>Pridobljeno 27.04.2016 iz, <http://github.info/>.

<sup>3</sup>Pridobljeno 27.04.2016 iz, <http://pypl.github.io/PYPL.html>.

prevedeno kodo poganja. Sintaksa programskega jezika je zelo podobna **C++**. Popularnost programskega jezika je še izboljšal zaradi operacijskega sistema za tablice in telefone **Android**, ki prav tako teče na različici (JVM) oz so programi napisani v Javi [19].

V izobraževanju je Java postala zelo priljubljena, prav zaradi zmožnosti, poganjanja programov na različnih platformah. Uporablja se na primarnem področju izobraževanja, predvsem na srednjem in visokem šolskem področju. V sekundarnem področju izobraževanj se je uporabila predvsem za pisanje programske opreme, ki dopolnjuje izobraževanje, kot so fizične simulacije (*fizleti*) in podobno. Eden od razlogov, da se je Java na tem področju dobro uveljavila je tudi ta, da omogoča zagon aplikacije s spletnega brskalnika, vendar je za to potrebna instalacija posebnega vtičnika, ki to omogoča. Primer 2 prikazuje sintakso "*Dobrodošel svet*" napisanega v Javi.

#### Primer 2: Program napisan v javi

```
1 class First {  
2     public static void main(String[] arguments) {  
3         System.out.println("Dobrodosel svet!");  
4     }  
5 }
```

#### 3.4.2 C++

C++ je vse namenski programski jezik, ki je OO in je bil zasnovan kot sistemski programski jezik. Večina operacijskih sistemov je danes napisana v kodi C++ in predhodniku C. Kodo programskega mora prevesti prevajalnik preden jo lahko zaganjam, za vsako platformo moramo prevajati posebej. C++ velja za najhitrejši programski jezik, z njim lahko opravljamo tako naloge, kot je neposreden nadzor nad polnilnikom, kot tudi vse višje funkcije, ki jih omogoča. Zato velja za enega težje učljivih programskih jezikov. Programiranje v C++ se uči predvsem na višjem in univerzitetnem izobraževalnem nivoju [20].

### Primer 3: Program napisan v C++

```
1 // 'Hello World!' program
2 #include <iostream>
3
4 int main()
5 {
6     std::cout << "Hello World!" << std::endl;
7     return 0;
8 }
```

#### 3.4.3 Java Script

**Java Script (JS)** se je razvil z potrebe po bogatejših in dinamičnih spletnih straneh. Začetek spletja so predstavljali statični dokumenti, ki so bile povezane z hiper povezavami. Skriptni jezik z imenom Java script se je prvič pojavil z spletnim brskalnikom *Netscape 2.0*. Takrat je bilo možno vstavljanje kratkih odsekov kode, ki so spletne strani naredile dinamične. Težnja po standardizaciji skriptnega jezika se je pojavila ko se je na trgu pojavil *Internet explorer 3.0*, saj je ta imel svojo različico skriptnega jezika *JScript*. Sedaj se standardni jezik imenuje **ECMA Script** oz. točneje **ECMA-262**, ki opisuje glavne dele programskega jezika JavaScript brez specifikacij, spletnega brskalnika [16].

Če smo v prejšnjih poglavjih govorili, da sta bila **Java** in **C++** večnamenska jezika, je JS bil eden tistih, ki so tekli znotraj vgrajenega gostiteljskega okolja, kot je spletni brskalnik. Danes imamo tudi okolja, ki omogočajo, da JS teče na strežnihih, na namizju in mobilnih napravah. Torej, kljub zgoraj omenjeni omejitvi, postaja prav tako večnamenski skriptni programski jezik. V spodnjem primeru 4 imamo primer programa „*Dobrodošel svet!*”, z **HTML** ogrodjem. Tako programska kodo odpremo v spletnem brskalniku. Del skriptnega jezika se začne z značko `<script type="text/javascript">` JS koda `</script>`. Povejmo še to, da se koda programskega jezika ne prevaja, temveč jo poganja **tolmač**.

#### Primer 4: Program napisan v JavaScriptu + HTML ogrodje

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <title>Some Page</title>
5     <script type="text/javascript">
6       alert("Hello World!");
7     </script>
8   </head>
9   <body>
10    <p>The content of the web page.</p>
11  </body>
12 </html>
```

#### 3.4.4 Python

**Python** je zelo pogosto uporabljen večnamenski programski jezik. Njegovo kodo podobno kot JS poganja tolmač. Zasnovan je tako, da je koda čim bolj berljiva in njegova sintaksa omogoča, da programske koncepte zapišemo v čim manj vrsticah, kakor bi jih lahko v Javi ali C++. Če so posamezni odseki ali bloki programske kode pri Javi in C++ označeni z zavitimi oklepaji (""), jih v Pythonu označimo z tabulatorskim zamikom. V Pythonu lahko uresničimo več programskih paradigem, kot je OO ali proceduralno programiranje. Omogočen je dinamičen tip spremenljivk, ima urejeno avtomatsko upravljanje z pomnilnikom in ima veliko standardno knjižnico [21].

Pyon se veliko uporablja tudi v izobraževalne namen. Pri nas se priporoča za začetke učenja programskega jezika na srednjem šolskem izobraževalnem nivoju. Zaredi tega, ga bomo v diplomskem delu uporabljali, ko glavni demonstracijski programski jezik.

#### Primer 5: Program napisan v Pythonu HTML ogrodje

```
1 print ('Hello world!')
```

### 3.5 Osnovni koncepti programiranja

V naslednjem odstavku se bomo vprašali kako lahko formuliramo sintakso programskega jezika? In kaj je npr. definicija *kopice*. V ta namen definiramo mehko idejo po avtorju Hazzan [5], ki je naslednja. Mehka ideja je koncept, ki mu ne moremo pripisati toge, niti formalne

definicije. Mehke ideje ni niti možno opisati z točno določeno aplikacijo. Na tem mestu se postavlja vprašanje kako lahko definiramo nekaj kar se odvija po korakih.

Da odgovorimo na zgornji dve vprašanji, lahko povemo, da so pravila sintakse togih orisov pri pisanju programske kode in da so semantična pravila mehke ideje. Opozorimo še na to, da koncepti v računalniški znanosti niso le toga pravila ali samo mehke ideje, temeč skupek obojega. V spodnji tabeli 2 prikazuje primer spremenljivke.

Tabela 2: Prikaz dvojnih, togih in mehkih orisov idej na primeru spremenljivke [5].

	<b>togi orisi</b>	<b>mehki orisi</b>
ime spremenljivke	Pravilo sintakse.	Potreba po imenu spremenljivke. Katero ime spremenljivke je pomembno in zakaj ga je potrebno določiti.
vrednost spremenljivke	Pravila tipa spremenljivke. Rezervacija pomnilnika.	Spremenljivka ima eno vrednost, ki se lahko spreminja s časom.
dodelitev začetne vrednosti	Pravila sintakse.	Pomen dodelitve začetne vrednosti

V nadaljevanju bomo pregledali in skušali razložiti osnovne koncepte pri programiranju. Z primerom bomo pokazali enega izmed načinov, kako jih prestavimo. Za vodilo bomo uporabili učni načrt za OŠ, ki smo ga pregledali v poglavju 2.3.2 in SŠ, ki je v poglavju 2.4. Primeri programov, ki jih bomo uporabljali in prilagodilo so povzeti s knjige in spletnih strani *“Learning python the hard way”* [22].

### 3.5.1 Izpis in računske operacije

Osnovno interakcijo z računalnikom lahko opišemo na naslednji način. Računalniku damo neke vhodne podatke, ta podatke po navodilu programa obdelava in nam poda rezultate na neko izhodno napravo. Ta izhodna naprava je na primer zaslon in na njem se izpisujejo obdelani podatki. Izpišimo nekaj stakov, pri tem uporabljammo ukaz `print`.

### Primer 6: Izpis besedila na zaslon | 01\_izpis\_na\_zaslon.py [22]

**Navodilo naloge:** Sledi navodilu, ki je zapisano v programske kodi.

```
1 #Del programa, ki nocemo, da ga uposteva tolmac,  
2 #oznacimo z # in ga imenujemo komentar.  
3 print "Pozdravljeni, to je nas prvi izpis na zaslonu."  
4 print "Izpisemo lahko tudi pravzno vrstico. \n"  
5 #Za izpis prazne vrstive uporabimo "\n"  
6 print "Pred to vrstico je prazna! in za njo.\n"
```

```
$ python 01_izpis_na_zaslon.py  
Pozdravljeni, to je nas prvi izpis na zaslonu.  
Izpisemo lahko tudi pravzno vrstico.
```

Pred to vrstico je prazna! in za njo.

Ena izmed glavnih nalog računalnikov so računske operacije, zato si poglejmo dva primera izračunov v programskej jeziku Python.

### Primer 7: Računske operacije | 02\_racunske\_operacije.py [22]

**Navodilo naloge:** Sledi navodilu, ki je zapisano v programske kodi.

```
1 #Izračunajmo nasednje izraze in izpišimo njihovo rezultat.  
2 #Preden poženemo program izračunajmo vrednost sami.  
3 print 100 - 5%2 + 3*4 - 22/3  
4 print 4+7 > 13
```

```
$ python 02_racunske_operacije.py  
104  
False
```

## 3.5.2 Spremenljivke

V tem poglavju bomo uresničili naslednje cilje vendar smo jim nekoliko spremenili vrstni red:

- **znajo izpisovati vrednosti spremenljivk med izvajanjem programa in izpisati končni rezultat,**
- **znajo spremenljivkam spremeniti vrednost s prireditvenim stavkom,**
- **znajo v program vključiti konstante in spremenljivke,**
- **razumejo različne podatkovne tipe in jih znajo uporabiti v programu,**

- **znajo v programu prebrati vhodne podatke in jih vključiti v program,**

Spremenljivke so način kako shranujemo podatke v računalniku. Ime spremenljivke ima podobno vlogo kot imena ljudi ali stvari v vsakdanjem življenju. Ljudje in stvari imajo imena zato, da si jih lažje zapomnimo in se z njimi in o njih lažje pogovarjamo. Podobno je to v programiranju, izbrati si moramo dobra imena spremenljivk, saj bomo tako lažje brali napisano kodo. Poglejmo primer 8.

### Primer 8: Izpis besedila na zaslon | 03\_uporaba\_spremenljivk.py [22]

#### Navodilo naloge:

Na parkirišcu je 100 avtomobilov, vsak izmed avtomobilov ima 5 sedežev. Z temi avtomobili želimo pripeljati 90 potnikov od tega jih ima 30 vozniško dovoljenje. Izračunaj in izpiši naslednje podatke.

- Koliko avtomobilov je navoljo. Koliko šoferjev je navoljo?
- Koliko avtomobilov bo ostalo na parkirišcu, če bodo vozili vsi šoferji?
- Koliko ljudi lahko prepeljemo z vsemi avtomobili?
- Kakšno je povprečno število potnikov, če vozijo vsi vozniki?

---

```

1 #1. Določimo spremenljivke:
2 avtomobili = 100
3 prostor_v_avto = 5.0
4 potniki = 90
5 soferji = 30
6 #2. Izračunajmo vrednosti in jih shranimo v spremenljivke:
7 avtomob_na_park = avtomobili - soferji
8 kapac_avtomob = avtomobili * prostor_v_avto
9 avtomob_na_potnikov = potniki/prostор_v_avto
10 povpr_st_potnikov = potniki/soferji
11 #3. Izpišimo vse zahtevane podatke.
12 print ("Na voljo je", avtomobili, "avtomobilov.")
13 print ("Na voljo je", soferji, "šoferjev.")
14 print ("Na parkiriscu bo ostalo", avtomob_na_park, "avtomobilov.")
15 print ("Z vsemi avtomobili prepeljemo", kapac_avtomob, "potnikov.")
16 print ("Povpr. st. potnikov je", povpr_st_potnikov, ", ce vozijo vsi
      soferji.")

```

---

```

$ python 03_uporaba_spremanljivk.py
Na voljo je 100 avtomobilov.
Na voljo je 30 šoferjev.
Na parkiriscu bo ostalo 70 avtomobilov.
Z vsemi avtomobili lahko prepeljemo 500.0 potnikov.
Povprecno stevilo potnikov je 3, ce vozijo vsi soferji.

```

### 3.5.3 Pogojni stavki in vejitve

V tem poglavju bomo uresničili naslednje cilje: **znajo uporabiti pogojni stavek in izvesti vejitev.**

#### Primer 9: Program odločitve | 05\_pogojni\_stavki.py [22]

**Navodilo naloge:** Kodo programa spremeni tako, da bo potniki potovali z avtomobili.

```
1 potniki = 40
2 avtomobili = 40
3 if avtomobili > potniki:
4     print ("Pojdite z avtomobili.")
5 elif avtomobili < potniki:
6     print ("Ne morete z avtomobili.")
7 else:
8     print ("Ne morem se odločiti.")
```

```
$ python 05_pogojni_stavki.py
Ne morem se odločiti.
```

### 3.5.4 Zanke

V tem poglavju bomo uresničili naslednje cilje: **razumejo pojem zanke in ga znajo uporabiti za rešitev problema.**

#### Primer 10: Zanka | 06\_zanka.py [22]

**Navodilo naloge:** Napišite zanko, ki pregleda seznam in prešteje in izpiše vsa liha števila.

```
1 #Podan seznam celih stevil.
2 cela_st = [11, 41, 2, 32, 12]
3 vsota_lihih = 0
4 for st in cela_st:
5     if st%2 == 1:
6         print (st)
7         vsota_lihih += 1
8 print ("Vseh lihih stevil:", vsota_lihih)
```

```
$ python 06_zanka.py
11
41
Vseh lihih stevil: 2
```

## 4 Pristopi in strategije poučevanja

V naslednjem poglavju raziščemo, kaj so sodobni pristopi in značilne strategije pri poučevanju računalniške znanosti in programiranja. Izpostavili bomo **model aktivnega učenja** in **strategijo reševanja problemov**. Z razumevanjem tega bomo v nadaljevanju lažje ocenili ali uporaba spletnih portalov vzpodbuja oba pristopa.

### 4.1 Model aktivnega učenja

Vsak pouk računalništva naj bi imeti modelno zgradbo in bi upošteval naslednja načela:

- naj vzpodbuja študente s pozitivno naravnanim poukom in jim naj omogoča okolje kjer najdejo pomoč.
- Pouk računalništva je naj grajen na konstruktivnih metodah poučevanja in aktivnem učenju.

**Konstruktivizem** je kognitivna teorija, ki preučuje naravo procesov učenja. Po tem principu naj bi učenci konstruirali novo znanje na osnovi preurejanja in izpopolnjevanja že obstoječega znanja. Znanje se gradi na obstoječih mentalnih strukturah in na odzivu, ki ga dobi učenec iz učnega okolja. Mentalne strukture so grajene korak za korakom, ena za drugo, seveda s to metodo lahko pride tudi do sestopanja ali slepih koncev. Proses je povezan z Piagetovim mehanizmom asimilacije [5].

Pri **aktivnem učenju** je najpomembnejše to, da učenci z lastno aktivnostjo ugotovijo, sami za sebe kako nekaj deluje. Sami si morajo izmisliti primere, preiskusiti lastne veščine in reševati naloge, ki so jih že ali jih še podo spoznali. Učenje je aktivno usvajanje, je gradnja idej in znanja. Za učenje mora biti posameznik aktivno vključen v gradnjo svojih lastnih mentalnih modelov. Model aktivnega učenja je sestavljen s štirih korakov [5].

- **Sprožilec** Je je naloga, ki predstavlja iziv za uvod v novo tematiko.
- **aktivnost** Študenti izvajajo aktivnost, ki jim je bila predstavljena v sprožilcu. Ta kora je lahko kratek ali lahko zavzame večju del učne ure. To je odvisno od vrste sprožilca in izobraževalnih ciljev.
- **diskusija** sledi po koncu aktivnosti, kjer se zbere zeloten razred, neglede na obliko dela. V temo koraku študenti izpopolnijo koncepte in ideje, kod del konstruktivnega učnega procesa.
- **povzetek** je lahko izračen v različnih oblikah, kot so zaokrožene definicije, lahko so miselnici vzorci ali povezav med temami, ki so jih obravnavali študenti in med drugimi temami, ki se navezujejo nanje.

Ko se ta model izkaže za primernega, ga lahko uporabimo v številnih učnih urah v različnih variacijah. Zanima nas ali znajo pristop aktivnega učenja spletni portali upoštevati?

## 4.2 Strategije reševanja problemov

Programiranje je preces pri katerem rešujemo probleme, zato je reševanje problemov v središču poučevanja računalniške znanosti. Reševanje problemov je zahteven mentalni proces. Če na spletu pobrskamo za strategije reševanja problemov lahko hitro ugotovimo na obstajajo različne strategije. Kot so recimo naštete na strani *Wikipedia:Reševanje problemov* (ang. *Problem solving*) [6], **abstrakcija, analogija, brainstorming, deli in vladaj** in mnoge druge. Proces in tehnike reševanje problemov se uporablja v mnogih tehničnih in znanstvenih disciplinah [5].

V nekaterih primerih učenci sami razvijejo strategijo s katero rešijo nek problem. Otroci si na primer sami izmislijo enostreno seštevanje in odštevanje, dolgo pred tem kadar se to učijo pri pouku matematike. Toda brez formalne podpore za učinkovito strategijo reševanja problemov, spodleti še tako inovativnemu učencu tudi pri enostavnih strategijah kot je **preizkus in napaka**. Zato je pomembno, da se uči strategij za reševanje problemov.

Vsak osnoven proces, ki se ukvarja z reševanjem problemov, ne glede na znanstveno disciplino, se začne z opisom problema. Vsak problem se navadno zaključi z neko rešitvijo, ki je v nekaterih primerih izražena z **zaporedjem korakov** ali **algoritmom**. V računalnik algoritmom zapišemo z kodo nekega programskega jezika. Zapisan algoritmom testiramo tako, da kodo prevedemo v strojni jezik in jo izvedemo. Za pravilno delovanje programa primerjamo vhodne in željene izhodne podatke. Preden pridemo od opisa problema do podane rešitve moramo prehoditi kar nekaj težkih korakov. Na te vmesne korake lahko gledamo kot na procese odkrivanja, zato lahko na reševanje problemov gledamo tudi kot na kreativen, umetniški proces. Splošno priznani koraki reševanja procesov so naslednji [5]:

1. *Analiza problema.* Najprej je pomembno da razuemo kaj je problem in ga znamo identificirati. Če tega ne znamo, ne moremo priti do nobene rešitve.
2. *Alternativne rešitve.* Razmišljamo o alternativnih rešitvah kako bi lahko rešili nek problem.
3. *Izbira pristopa.* Izberemo primeren pristop, kako rešiti problem.
4. *Razgradnja problema.* Problem razgradimo na manjše podprobleme.
5. *Razvoj algoritma.* Algoritmom razvijamo po korakih, ki smo jih določili v podproblemih.
6. *Pravilnost algoritma.* Preverjanje pravilnosti algoritma.
7. *Učinkovitost algoritma.* Izračunamo učinkovitost algoritma.
8. *Refleksija.* Naredimo refleksijo in analizo na pot, ki smo jo naredili pri reševanju problema in naredimo zaključek z tem kar lahko izboljšamo za naslednji problem, ki ga

bomo reševali.

Točen recept kako se lotiti reševanja ne obstaja. Učencem lahko le pokažemo nekatere metode in strategije, ki jim lahko pomagajo pri reševanju problemov. Da bi bolje znali oceniti, kako izrazito spletni portali upoštevajo korake strategije reševanja problemov, poglejmo še nekatere pomembne korake podrobneje in koko se z njimi spopadajo novinci.

#### 4.2.1 Razumevaje problema

Razumevanje problemov je prva stopnja v procesu reševanja problemov. Pri reševanju algoritemskih nalog najprej moramo prepoznati, kaj so vhodni podatki in kateri podatki naj bi bili izhodni. Če znamo povedati kaj bodo vhodni podatki, razumemo tudi bistvo samega problema.

#### 4.2.2 Načrtovanje rešitve

Novinci se spopadajo z največjimi težavami na začetni stopnji načrtovanja rešitve za nek problem. V nadaljevanju so predstavljene tri strategije, ki jih lahko uporabimo na tem koraku reševanja problema.

**Definicija spremenljivk problema:** Pri rešitvi problema si pomagamo tako, da ugotovimo kaj morajo biti vhodni in kateri bojo izhodni podatki. S tem razjasnimo problem. V naslednjem koraku definiramo **spremenljivke**, ki so potrebne za rešitev problema.

**Postopno izboljševanje (ang. *Stepwise Refinement*):** Po tej metodi nas najprej zanima celoten pregled strukture problema in odnosi med posameznimi deli. Zatem se šele poglobimo specifični in kompleksni implementaciji posameznih pod problemov. Postopno izboljševanje je metodologija, ki poteka od **zgoraj-navzdol**, torej od splošnega k specifičnemu. Drugačen pristop je od **spodaj-navzgor**. Za oba pristopa velja da eden drugega dopolnjujeta. V obeh primerih je problem razdeljen na manjše pod probleme ali naloge. Glavna razlika med obema je mentalni proces, ki je potreben za en ali drugi pristop. V nadaljevanju se posvetimo samo pristopu od **zgoraj-navzdol**. Rešitev, ki jo poda **postopno izboljševanje** ima modularno obliko, ki jo:

1. jo lažje razvijamo in preverjamo,
2. jo lažje beremo in
3. nam omogoča, da uporabljam posamezne pod rešitve tudi za reševanje drugih problemov.

**Algoritemski vzorci:** Algoritemski vzorci združujejo matematični pogled in elemente načrtovanja. Vzorec podaja načrt na rešitev, s katero se srečamo mnogokrat. Algoritemski vzorci so primeri elegantnih in učinkovitih rešitev problemov in predstavljajo abstraktni

model algoritemskega procesa, katerega lahko prilagodimo in ga integriramo v rešitve drugim problemom.

Pri tem procesu lahko nastopi težava prepozname vzorca algoritma pri novincih, saj ti niso sposobni prepoznati podobnosti med posameznimi algoritmi ali ne znajo prepozнатi bistvo problema, njihove posamezne komponente in razmerja med njimi, da bi lahko rešili nove probleme. V takih primerih novinci radi ponovno izumijo že njim pozname rešitve, ki bi jih lahko uporabili. Te težave navadno nastanejo zaradi slabe organizacije sistematike znanja o algoritmih.

Proces reševanja problemov z algoritemskim vzorcem se navadno začne z prepoznamenjem komponent, ki vodijo k rešitvi in iskanjem podobnih problemov, na katere še imamo znane rešitve. Zatem prilagodimo vzorec prilagodimo za rešitev problema in ga vstavimo v celotno rešitev. V večini primerov je potrebno vstaviti več različnih vzorcev, da dobimo neko novo rešitev.

#### 4.2.3 Preverjanje rešitve

Ko imamo pripravljeno rešitev moramo preveriti ali je ta pravilna. Pogled na preverjanje pravilnosti rešitve je lahko teoretične in praktične narave. Razhroščevanje (*ang. debugging*) spada me vrsto aktivnosti, ki nam pomaga pri ugotavljanju pravilnosti rešitve. Splošno velja da proces razhroščevanja, z programom, ki nam pomaga razhroščevati (*ang. debugger*) ali brez njega, poglablja razumevanje računalniške znanosti. Z tem ko učenci razmišljajo, kako bodo preverjali ali njihov program deluje pravilno, hkrati v njih poteka miselni proces refleksije o tem kako so implementirali določen program in kako ga bojo morebiti morali spremeniti.

Na nivoju do srednje šole uporabljam praktične metode ugotavljanja pravilnosti programa, kot je razgloščevanje. Ko želimo znanje pravilnosti delovanja poglobiti se lahko lotimo tudi teoretične analize.

#### 4.2.4 Refleksija

Refleksija je mentalni proces ali obnašanje, ki nam omogoča da neko delovanje analiziramo in o njem tudi premislimo. Refleksija je pomembno orodje v splošnem učnem procesu, prav tako spadam med kognitivne procese višjega reda. Z refleksijo učenec dobi priložnost, da stopi korak nižje in premisli o svojem razmišljanju in tako izboljša večino reševanja problemov. Refleksivno razmišljanje je proces, ki zahteva veliko časa in vaje. Med procesom reševanja problemov, lahko refleksijo uporabimo na različnih stopnjah.

- *Pred* reševanjem problemom. Ko problem preberemo, in že načrtujemo rešitev, se splača uporabiti refleksijo in razmisiliti o tem ali smo morda že reševali podoben problem in

temu primeren vzorec algoritma.

- *Med* reševanjem problema. Ko rešujemo problem refleksija služi, kot pregled, kontrola in nadzor. Na primer, ko nastopijo težave pri načrtovanju rešitve ali morda zaznamo težavo ali napako. Temo procesu lahko pravimo **refleksija v akciji**.
- *Po* reševanju problema. Ko že najdemo rešitev, ki deluje, nam refleksija služi kot orodje z katerim pregledamo učinkovitost delovanja. Pregledamo strateške odločitve, ki so bile sprejete med samim načrtovanjem rešitve.

Refleksija je kreativni proces in je pomemben za učenca tako kot za učitelj.

## 5 Spletni portali za učenje programiranja

Spletne portale za učenje programiranja (**SPUP**) bomo predstavili in spoznali tako, da bomo najprej pregledali kaj so bili glavni razlogi, da so se pojavili. Spletni portali za učenje programiranja, v nadaljevanju krajše **SPUP** so nastali takoj po

razmahu interneta v začetku novega tisočletja. Najprej so nastali na univerzah, zanima nas, kaj so glavni razlogi za nastanek SPUP. Razen tehnoloških zmožnosti IKT za nastanek spletnega portala nas zanimajo predvsem težave, ki so jih skušali premostiti z uporabo SPUP.

Spletni portali so nastali na različnih univerzah, ogledali si bomo spletni portal, ki je nastal na na *odprtih univerzah v Hong Kong-u* (**OUHK**), na *Univerzi Strathclyde iz Velike Britanije* (**USVB**) in *Queensland University of Technology, Australia* (**QUTA**).

Zanimali nas bodo predmeti, ki veljajo za začetne pri poučevanju računalniške znanosti in programiranja. **Novinci**, kot jih bomo imenovali so študenti, ki se šele začnejo učiti programiranja. V diplomskem delu nas zanimajo le učenci osnovnih šol in dijaki srednjih, vendar se oni prav tako šele srečujejo s programiranjem, podobno kot študentje in jih bomo zato lahko vse poimenovali kot **novince**.

Kot je razvidno iz literature bomo lahko sklepali na nekatere skupne značilnosti vseh novincev, ne glede na težavnostno stopnjo na kateri se nahajajo, saj je programiranje večina, ki ni dana naravno in se je moramo vsak priučiti.

### 5.1 Razlogi za nastanek spletnih portalov

Na Odprti univerzi v Hong Kong-u (**OUHK**) ponujajo tri računalniške sklope različnih težavnosti, za dodiplomske programe. Imajo zelo veliko populacijo študentov, ki se učijo programiranja. Avtorji članka [3] ugotavljajo, da je proces učenja programiranja kompleksen in zahteva veliko vaje programiranja. Izkaže se, da praktični del igra poglavitno vlogo v učnem procesu.

Glavna težava s katero se srečujejo na **OUHK**, je ta, da se s številom študentov, ki se vpisujejo v smeri računalništva povečuje. Povečanje študentov pomeni manj časa za mentorstvo za posameznega študenta.

Da bi študentje, lahko normalno sledili pouku na daljavo, si morajo doma urediti delovno okolje, kjer lahko programirajo. Študenti, dobijo vso potrebno učno literaturo in tudi programsko opremo, ki predstavlja **prevajalnik** in **razvojno okolje**. Izkaže se, da imajo številni težavo nastaviti in se spoznati z integriranim razvojnimi okoljem (ang. *Integrated Development Environment (IDE)*) [3].

Težave pri izobraževanju na daljavi, se pojavijo tudi v komunikaciji. Študent, ki se izobražuje od doma in naleti na neko težavo, ki je ne ve sam rešiti, nima dostopa do svojih kolegov, ali mentorjev. Do mentorjev dostopa lahko le preko telefonskih klicev ali elektronske pošte. Če pogledamo še s strani mentorjev, imajo ti težavo z spremeljanjem napredka velikega števila študentov.

Naslednji članek, ki so ga sestavili avtorji z *Univerze Strathclyde iz Valike britanije (USVB)*, se ukvarja z raziskovanjem vpliva nove strategije kognitivnega pristopa k poučevanju programiranja, ki spreminja mentalni model študentom tako, da v njih ustvari konflikt. V ta namen je bilo razvito tudi spletno okolje, ki implementira uporabo nove kognitivne strategije [4].

Kot pravijo avtorji v članku, z hitrim razvojem IKT, narašča tudi potreba po sposobnih programerjih in učenje programiranja postaja globalna skrb. V prvem letu pri predmetih programiranja, študenti obvladajo naloge programiranja dosti slabše kot bi to pričakovali. Slaba uspešnost s pozna predvsem na tem, da se mnogi izpišejo z smeri računalništva, takih je kar od 30% do 50%. Kot avtorji poudarjajo in povzemajo po drugih študijah so za to v glavnem krive težave pri reševanju problemskih nalog, ki nastopajo v programiranju. Nekatere druge študija vidijo krivca za neuspeh tudi v napačnem razumevanju ključnih konceptov pri programiranju, ki so lahko posledično krivi za težave pri reševanju problemov. Tradicionalni učni pristop je za učenje programiranja manj zanesljiv, da bi zagotovil pravilnost v razvoju mentalnih modelov o konceptih programiranja. Študije kažejo da študenti po enem letu predmeta programiranja še vedno nimajo pravih mentalnih modelov o osnovnih programskeih konceptih.

Na univerzi QUTA se z začetnimi predmeti programiranja srečujejo s podobnimi težavami, kot na OUHK in USVB.

1. Inštalacija in nastavitev okolja za programiranje.
2. Uporaba urejevalnika besedil.
3. Razumevanje programskih vprašanj in uporabe sintakse jezika pri pisanju programske kode.
4. Razumevanje napak prevajalnika.
5. Razhroščevanje.

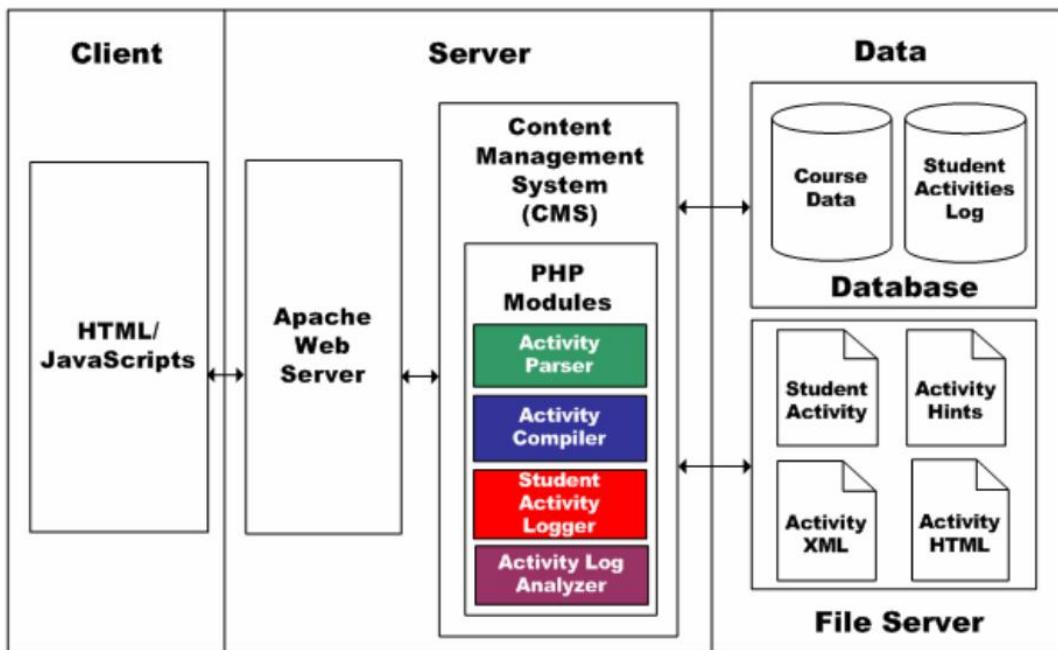
Ugotavljam, da je pri tistem vajah programiranja pomembno, da ob težavah, novinci dobijo čimprajšen odziv mentorja. V velikih razredih se to izkaže za zelo zahtevno. Tudi začetniki, ki uspešno premagajo začetne ovire in se lotijo takojšnjega programiranja, imajo zelo slabo napisano in konstruirano programsko kodo. Pomagati študentom, pistati kvalitetno programsko kodo je prav tako časovno zelo zahtevno. Težave programiranja se stopnjujejo ko se za učenje programiranja uporabljajo OO programski jeziki, saj ti zahtevajo visoko stopnjo stopnje abstraktnega razumevanja programskih konceptov. Za izdelavo spletnega portala za učenje programiranja so na QUTA bili pomembni naslednji cilji [7].

- Omogočiti lažji začetek pri učenju programiranja, z pogostim odzivom mentorjev na težave novincev. S pomočjo ob pravem času spremenimo odnos novincev do programiranja.
- Izboljšati uspeh začetnih predmetov programiranja.
- Da pomagamo mentorjem pri učenju in administraciji predmetov programiranja.

## 5.2 Primeri implementacije in sistemska arhitektura

Zanimalo nas tudi kakšna je morebitna sistemska arhitektura takega spletnega portala, zato si pomagamo z primerom, arhitekture, ki so ga izdelali na **OUHK**. V nadaljevanju govorimo o *aktivnostih*, ki jih mora študent opraviti, to so naloge, programske rešitve na zastavljene probleme. Študenti na **OUHK** se učijo programiranja v programskega jeziku **JAVA**.

Kot prikazuje slika 1 je sistem urejanja vsebine (*ang. Content Management System (CMS)*), teče na spletnem strežniku *Apache* z *MySQL* podatkovno bazo. Sistem je narejen iz štirih pod modulov, ki so napisani v skriptnem jeziku *PHP*.



Slika 1: Sistemska arhitektura spletnega portala za učenje programiranja, kot so jo naredili na OUHK [3].

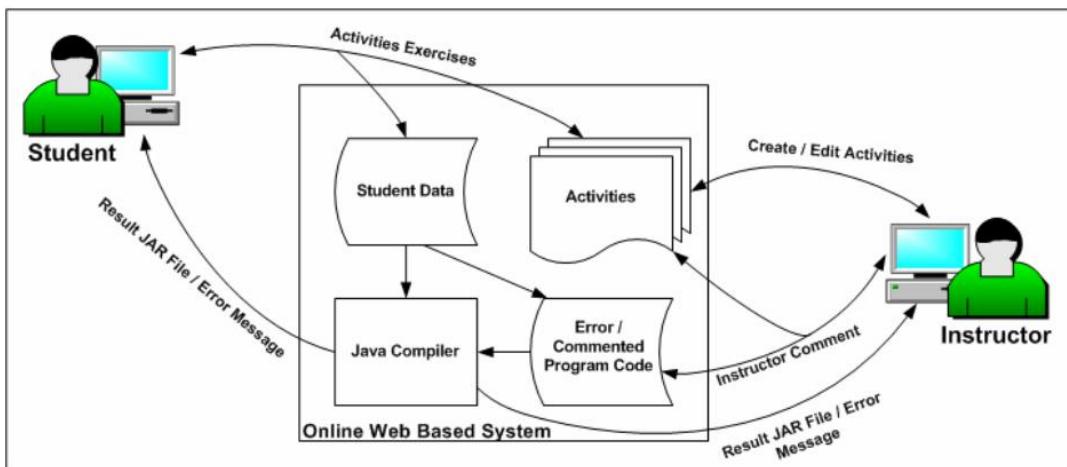
Ti moduli so naslednji, zajem aktivnosti, prevajalnik aktivnosti, dnevnik študentove aktivnosti in analizator dnevnikov aktivnosti. Samo delovanje je naslednje, ko odjemalec pošlje zahtevo za neko aktivnost, se ta naslovi strežniku, ki poišče programsko aktivnost. Z modulom *zajema aktivnosti*, strežnik zajame aktivnost, ki je zapisana v obliki **XML** in naloži vse potrebne datoteke. Zajem aktivnosti, prav tako naloži študentovo predhodno delo, ki je shranjeno

datoteki aktivnosti. Ko se vse zajame in naloži se vsebina pošlje v obliki **HTML** nazaj k klientu.

Strežnik omogoča tudi prevajanje aktivnosti. Ko strežnik dobi prošnjo za prevajanje programske kode, se ta prevede, če seveda v njej ni sintaktičnih napak in se ustvari datoteka **JAR**, ki jo študent lahko prenese s strežnika. Če so v programu napake, se ustvari dnevnik napake, v trenutni aktivnosti, prav tako se napaka izpiše na zaslonu študenta. Vsako aktivnost zajame dnevnik študentove aktivnosti in jo shrani v podatkovno bazo. Z analizatorjem dnevnika študentove aktivnosti, mentorji dobijo vpogled v delo študenta in njegovega napredka.

### 5.3 Pregled delovanja in interakcija s SPUP

Opišimo, kako so si zamislili interakcijo med študentom in mentorjem s spletnim sistemom na **OUHK**. Diagram prikazuje slika 2. Spletni sistem omogoča študentom in mentorjem spletno okolje za učenje programiran. Mentorji na spletni portal naložijo snov preko spletnega brskalnika. Mentor lahko naloži datoteke opisom aktivnosti. Ta datoteka vsebuje osnovne opise in informacije o aktivnostih. Posebej naloži še datoteko v kateri je predloga za aktivnost. V to predlogo študent rešuje zadano nalogu. V posebno datoteko je naložen tudi namig, ta je študentu v pomoč in ponuja primer izpisa programa.



Slika 2: Prikaz med interakcijo študenta in mentorja s spletnim portalom [3].

Študent lahko pregleduje vse aktivnosti in si naloži katero koli izmed njih. Omogočeno ima, da program prevede na strežniku, ko prevajalnik naleti na napake strežnih vrne napako na spletno stran. Če študent naleti na težavo, ki je povezana z reševanjem aktivnosti, lahko pošlje prošnjo za pomoč svojemu mentorju. Ko se mentor prijavi v sistem ima vpogled v napako in na začasno delovno datoteko študenta, mentor lahko zaganja prevajalnik na tem začasnem projektu študenta. Ko mentor popravi programsko napako, odgovori študentu in poda komentar na programsko kodo študenta. Študent ima vpogled v komentarje in predloge, ki jih je posredoval mentor [3].

Na univerzi v US [4], je okrog strategije kognitivnega konflikta nastalo spletno okolje, ki naj bi izboljšalo mentalne modele ključnih programskega konceptov. Učni model je sestavljen iz štirih korakov:

- **Predhodni korak:** Mentor razišče kakšni so predhodni mentalni modeli študentov in identificira neprimerne.
- **Korak kognitivnega konflikta:** V študentovi predstavi mora sprožiti tak dogodek, ki v študentu izzove neskladje z njegovo predhodno predstavo in s tem se študenta potisne v konfliktno situacijo.
- **Korak konstruiranja modela:** Vizualizacija študentu pomaga ustvariti pravo mentalno predstavo.
- **Korak aplikacije:** Študent mora rešiti programsko nalogu z na novo ustvarjeno mentalno predstavo.

Spletno učno okolje podpira programski jezik **JAVA**. Za učenje programskih konceptov je na spletni strani vsak posamezen koncept povezan z potjo, ki predstavlja načrt potovanja. Poti koncepte se povezujejo tako, da se ti nadgrajujejo, saj znanje določenega koncepta potrebuje neko predznanje prejšnjega. Tako za razumevanje določevanja reference najprej potrebujemo predznanje o spremenljivkah ali npr. preden se študenti učijo kako s podajajo parametri v podprograme najprej morajo razumeti kaj je obseg nekega pod programa. Torej je vrstni red spoznavanja programskih konceptov pomemben. Med potmi so gumbi, ki predstavljajo vsak koncept. Na vsakem gumbu je označen rdeč križ kar pomeni, da študent še ni spoznal koncepta. Ko študent opravi naloge povezane s posameznim konceptom se rdeč križ spremeni v zeleno kljukico. Ko študent vstopi v koncept se izpiše študentova zgodovina z nalogami tega koncepta. Vsaka naloga vsebuje tako vprašanje, ki sproži konfliktno situacijo v mentalnem modelu študenta. Nato študenti dobijo učni material v visualni obliki. Za vizualizacijo uporabljajo orodje **Jeliot**, ki dinamično upodablja izvajanje javaskih programov. Za pravilnost razumevanje mentalnega modela mora študent odgovoriti na dodatna vprašanja. Če študentovi odgovori niso v skladu z podanim mentalnim modelom, dobi študent povratno informacijo o nepravilnem odgovoru. Naslednji korak je ta, da študent mora zagnati vizualizacijo dela programske kode, ki si ga je prej moral predstavljati. Tako ima možnost, da zazna nepravilnost v svojem mišljenju in tako lahko gradi na pravilnem konceptu [4].

V preteklosti je bilo razvitih mnogo orodij, ki so nastala ravno z raziskovanja učenja programiranja, vendar mnoga od teh zahtevajo, da študenti pišejo celotne programe od začetka do konca. Spletni portal v primeru QUTA uči programiranja v programskega jeziku Java in ima naslednjo naslednje zmožnosti [7].

1. Spletni portal za programiranja, ki omogoča naloge tipa "Zapolni prazna mesta".
2. Ogrodje za analizo, ki preverja kvaliteto in pravilnost, nalog, tipa Žapolni prazna mesta".
3. Avtomatski sistem za dajanje povratnih informacij, ki sporoča prilagojena sporočila

prevajalnika in formalni odziv študentom in njihovim mentorjem. Poročilo vsebuje kvaliteto napisanega programa, strukturo in pravilnost glede na programsko analizo.

## 5.4 Rezultati izvedenih rešitev SPUP

Večina študentov smeri računalništva na OUHK nima predhodnih izkušenj v programiranju z programskim jezikom **JAVA**. Sistem se uporablja kot spletno okolje za učenje programiranja. Študentom je s tem, dana množica aktivnosti oz. nalog, katere morajo sami uspešno opraviti. To lahko počnejo kadarkoli in od koderkoli. Študentom ni potrebno nastavljati programskega okolja, študenti vse programe, ki jih napišejo, lahko takoj prevedejo in jih zaganjajo na svojih računalnikih. Uporaba spletnega portala je pokazala da so študentje oddali 100% programskih nalog, napisanih v javi. To kaže na to, da so študentje samozavestno reševali naloge in jih oddajali. Pred uporabo spletnega portala je oddaja nalog bila 80%.

Kot pravijo avtorji članka in portala [3], je to šele začetek uporabe spletnega portala, ki nudi osnovno funkcionalnost. V nadaljevanju nameravajo dodati še inteligentni sistem, ki po nadzoroval napredok študentov.

Za izboljšanje mentalnih modelov so avtorji predlagajo konstruktivno naravna učni model, ki vključuje strategijo kognitivnega konflikta in vizualiacijo programov. Zgodnje preizkušanje strategije kognitivnega konflikta pokažejo da so študenti bolj zavzeti za učni material in jih motivira tako, da si prej ustvarijo pravilno mentalno predstavo [4].

Tudi začetniki, ki uspešno premagajo začetne ovire in se lotijo takojšnjega programiranja, imajo zelo slabo napisano in konstruirano programsko kodo. Pomagati novincem, pistati kvalitetno programsko kodo je časovno zelo zahtevno opravilo.

Težave programiranja se stopnjujejo ko se za učenje programiranja uporablajo Objektno-orientirani programski jeziki, sej ti zahtevajo visoko stopnjo abstraktnega razumevanja programskih konceptov in so načrtovani predvsem za zahtevne programerje.

Rezultat dela avtorjev spletnega portala QUTA, gre še nekoliko naprej od OUHK in v njihov spletni portal vgradijo, odziv spletnega portala, ki mora o pravilnosti programa in o kvaliteti. Ogrodje (*ang. framework*) za analizo programske kode vsebuje naslednje komponente [7].

- Sintaktično ali semantično opozarjanje na napake ali napake prevajalnika.
- odziv na kvalitetu in pravilnost programske kode.
- Formalni odzin učitelja oz. komunikacija med učiteljem in učencem.

## 5.5 Značilnosti SPUP

Ena od osnovnih in glavnih komponent pri vseh SPUP je orodje, ki omogoča pisanje in preizkušanje programske kode. Poimenujemo jo lahko kot **Spletna aplikacija za programiranje (SAZP)**. Njene glavne značilnosti so naslednje:

- **urejevalnik besedil**, ki ima lahko osnovne funkcije ali tudi zahtevne, ki so značilne za IRO;
- omogočen je zagon napisanega progama z vhodnimi in izhodnimi podatki;
- omogočena je **povratna informacija**:
  - **sintaktičnih napak**, ki ju vrne prevajalnik ali tolmač;
  - **semantičnih napak**, ki preverjajo željen rezultat napisanega programa oz. pravilno rešitev;

Iz pregleda SPUP, ki so nastali na univerzah smo se lahko poučili, kaj so nekatere značilnosti spletnih portalov. Strnimo te značilnosti, saj jih bomo pozneje uporabili pri iskanji, kategorizirjanju in vrednotenju. Spletni portal vsebuje naslednje elemente:

- razdelano vsebino z nalogami oz. aktivnostmi;
- **Spletno aplikacijo za pisanje programske kode**;
- omogočena je komunikacija med mentorjem in novincem;
- omogočen je pregled nad napredkom novincem oz. tako imenovan *nadzor nad razredom*.

## 6 Kriteriji za klasifikacijo spletnih portalov

Spletni portali za učenje programiranja imajo različne značilnosti in zmožnost. V poglavju bomo razdelili posamezne kriterije s katerimi bomo klasificirali in ovrednotili posamezne spletne portale.

### 6.1 Vrsta vsebine

Po prvem pregledu in iskanju spletnih portalov lahko ugotovimo, da spletni portalu za učenje programiranja ponujajo najrazličnejše vrste vsebin in njihove kombinacije kot je na primer, **tekstovni vodič in spletno aplikacija za programiranje**. V posebno kategorijo bomo uvrstili tudi spletnne portale, ki ponujajo **spletne igre**, ki učijo programiranje. Različne vrste spletnih portalov, ki jih lahko obravnavamo so naslednje:

- **tekstovni vodiči**,
- **video vodič**,
- **spletna aplikacija za programiranje**, kot smo jo definirali v poglavju 5.5,
- **spletne igre**,
- **kombinacija** vrst vsebin, ki jih lahko še razdelimo na:
  - **najosnovnejša kombinacija** (*tekstovni vodič + preizkus kode*);
  - **napredna kombinacija** (*različne vrste vodičev + spletna aplikacija za programiranje*), ki tvorijo **vadnice**.

V tej diplomi se ne bomo podrobno ukvarjali s tem katera izmed vrst vsebin predstavlja boljše zmožnosti za prenos znanja. Vsaka ima svoje prednosti in slabosti, zato bomo za vsako izpostavili le bistvene pozitivne značilnosti in tudi slabosti. Zanimale nas bodo predvsem tise **kombinirane** vrste vsebin, ki bodo predstavljale čim bolj celovit spletni portal za učenje programiranja, kot smo ga definirali v poglavju 5.5.

#### 6.1.1 Tekstovni vodiči

Spletni vodiči veljajo med starejšimi metodami za podajanje znanja na spletu. Za njih je značilno, da uporabnika vodijo **po korakih** do nekega določenega cilja, kako nekaj narediti ali specifičnega znanja. Besedilo, ki podaja znanje je opremljeno z **primeri**. [23]

Značilni predstavniki takih vodičev je spletna stran <https://docs.python.org> na kateri najdemo vso dokumentacijo programskega jezika **Python**. Na strani najdemo tudi vodiča z naslovom *The Python Tutorial* [24].

### 3.1.1. Numbers

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators +, -, \* and / work just like in most other languages (for example, Pascal or C); parentheses (()) can be used for grouping. For example:

A screenshot of a computer screen displaying the Python Tutorial. The code examples are as follows:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5 # division always returns a floating point number
1.6
```

Annotations with arrows:

- An arrow points from the text "primer" to the second line of code: `>>> 50 - 5*6`.
- An arrow points from the text "razlaga" to the third line of code: `>>> (50 - 5*6) / 4`.

Slika 3: Zaslonski posnetke poglavja z vodiča *The Python Tutorial* s primerom [24].

Spletne vodiče pri pouku uporabljamo na podoben način kot bi vodiča napisanega na učnem listu z **metodo dela z tekstom**. Sicer je pomembno da učenci usvojijo uporabo spletnih vodičev, vendar so spletni vodiči marsikdaj prezahtevni za uporabo sploh na osnovno šolskem nivoju, s vso tehnično dokumentacijo. Negativna stran spletnih vodičev je še ta, da direktno s spletne strani ne moramo preizkušati primerov programske kode, kar je slabo tudi z motivacijskega vidika.

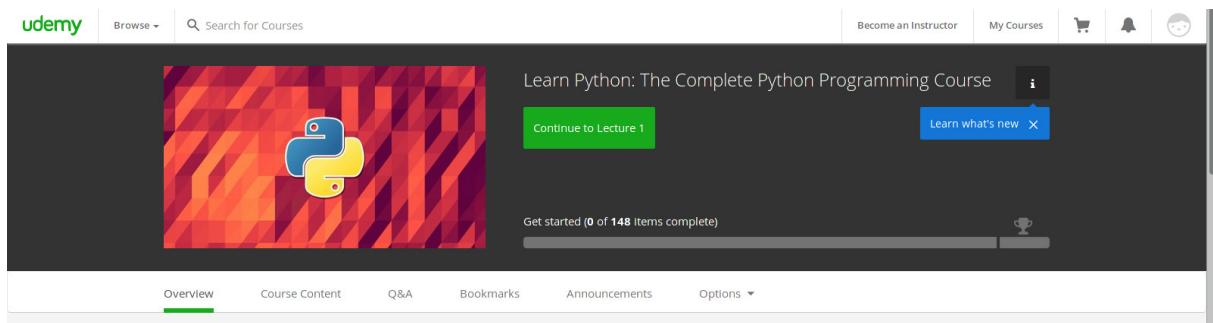
### 6.1.2 Video vodiči

Z razmahom video vsebin na spletu, so marsikateri spletni vodič dopolnili vsebine oz. jih zamenjali video vodiči. Popularno je postalo zajemanje oz. **snemanje lastnega namizja**. Vido vodiče najdemo za številna področja, od uporabe določene programske opreme in vse do programiranja. Ena izmed prednosti video vodičev naprem tekstovnim je ta, da ti omogočajo nazornejši prikaz nekega postopka, po korakih. Preden sami opravimo nek postopek, lahko v video posnetku opazujemo vsak korak, potek miške in poleg teka poslušamo razlago, če je ta vključena. Številne študije kažejo da je učenje s multimedijo, torej kombinacijo zvoka in slike bolj učinkovito, samo poslušanje ali branje teksta [25]. V razredu je uporaba video vodičev lahko koristna pri samostojnjem delu in domačem delu. Uporaba video vodičev ima tudi slabe strani, v njih je lahko predstavimo dosti manj vsebine in iskanje po vsebini v video ni preprosto, kot je to pri tekstu.

Eden izmed spletnih portalov, ki je specializiran za podajanje znanja s video vodiči je *Udemy* [26]. Na njem vsak kdo lahko postane učitelj in pripravi učne ure z različnih področij, ne samo z računalniške znanosti. Nekateri sklopi učnih ur so v celoti brezplačni, kljub temu je večin plačljivih.

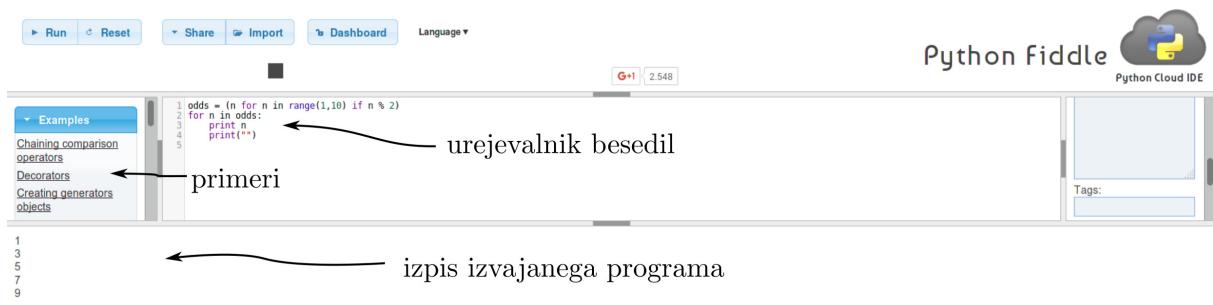
### 6.1.3 Spletna aplikacija za programiranje.

Nekatere spletne strani ponujajo le spletno aplikacijo za programiranje, kot smo jo povzeli v poglavju 5.5. Taki spletni portali ne ponujajo vsebine, ponujajo le aplikacijo, ki jo uporabimo kot **orodje**. Ali pa ponujajo le toliko vsebine, kot je potrebno, da se uporabnik nauči uporabljati



Slika 4: Uvodna stran enega izmed video vodičev za učenje pythona [26].

spletno aplikacijo. Kljub temu, da nas zanimajo celoviti spletni portali, ki ponujajo tudi vsebino, nas bodo podrobneje zanimala tudi orodja, saj so ta ključna za nekatere prednosti, ki jih ponujajo spletni portali za učenje programiranja. Predstavnik takega orodja je *Python Fiddle* [27]. Omogoča osnovni urejevalnik besedila (slika), z barvanjem programske kode, s predlogami za samo dokončevanja izpisa vgrajenih funkcij. Uvozimo lahko datoteke in jih delimo. Zaganjam napisane programe, izhodni podatki se izpišejo v konzoli.



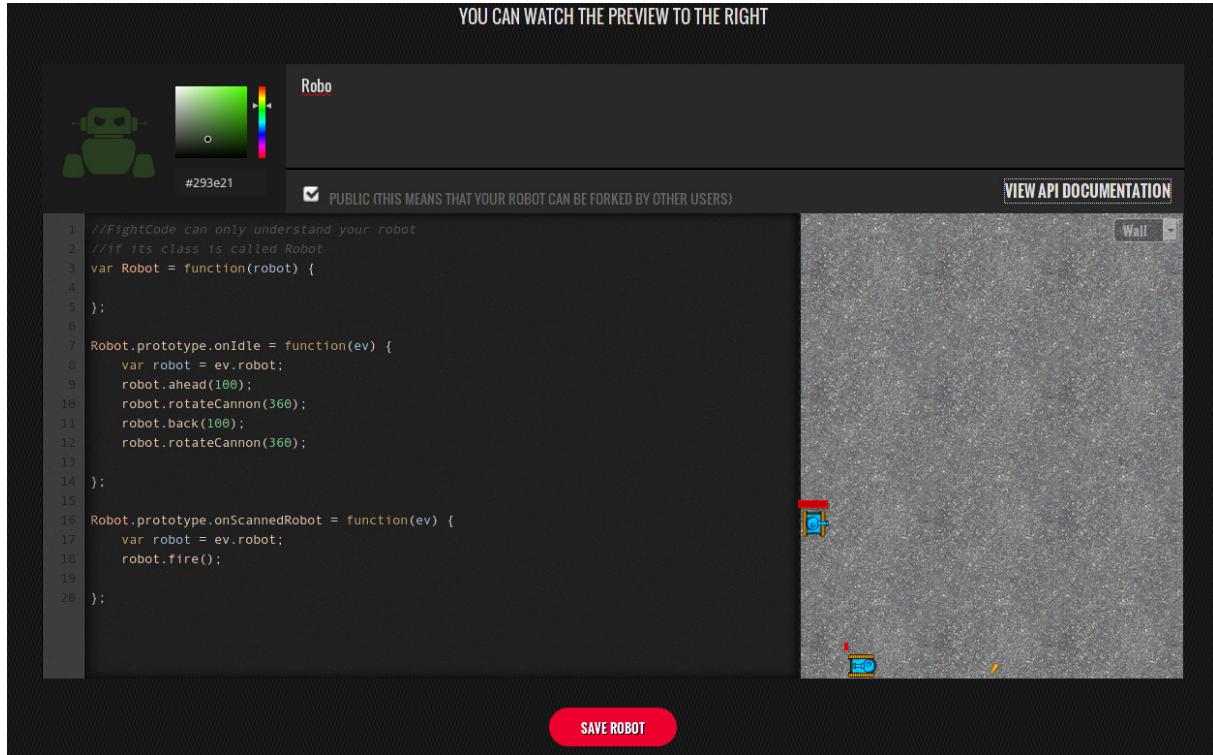
Slika 5: Zaslonska slika spletnne aplikacije za programiranje *Python Fiddle* [27].

Spletne tehnologije so danes zalo napredovale in že nekaj let se aplikacije in podatki selijo v **oblak**. Te aplikacije in podatki so dostopni od koder koli. V **oblaku** so razvijajo številna profesionalna okolja **IRO**, ki omogočajo delo na večjih projektih in ponujajo napredne funkcije IDE, ki smo jih drugače lahko imeli le z namiznimi aplikacijami. Navedimo dva primera: *Codenavy* [29] in *Cloud9* [28]. Oba ponujata profesionalen IRO v oblaku. Za uporabo v šoli sta ti dve okolji preveč zahtevni, in jih ni smiselno uporabljati pri poučevanju novincev. S tem jim otežimo učenje programiranja, saj prej potrebujejo čas, da spoznajo in se naučijo uporabljati IRO.

#### 6.1.4 Spletne igre

Na spletu obstajajo številni spletni portali, ki učijo in spodbujajo k učenju programiranja s igrami podobnimi vsebinami. Vsebina je razdeljena na stopnje. Igralci napreduje iz stopnje v stopnjo in pri tem nabirajo izkušnje, nove veščine in dosežke. Za igranje igre ne upravljamo

gibanje lika v igri z tipkovnico in miško temveč pišemo programsko kodo, ki upravlja njihovo početje. Takšni spletni portali dajejo zelo dobro motivacijsko osnovo, saj se novinci spoznajo na osnovne principe igranja iger. Primer spletnega portala je *Fightcode* [31] (slika 6). Igralci programirajo robota v programskem jeziku JavaScript. Vsak izmed igralcev lahko izzove drugega igralca v boj med roboti. Za vsako zmago se igralec pomika navzgor po lestvici najboljših robotov.



Slika 6: Zaslonska slika spletnne strani Fightcode [31].

V podrobnem pregledu si bomo še podrobnejše ogledali značilnosti nekaterih drugih spletnih iger, ki učijo programirati.

#### 6.1.5 Kombinirane vrste vsebin

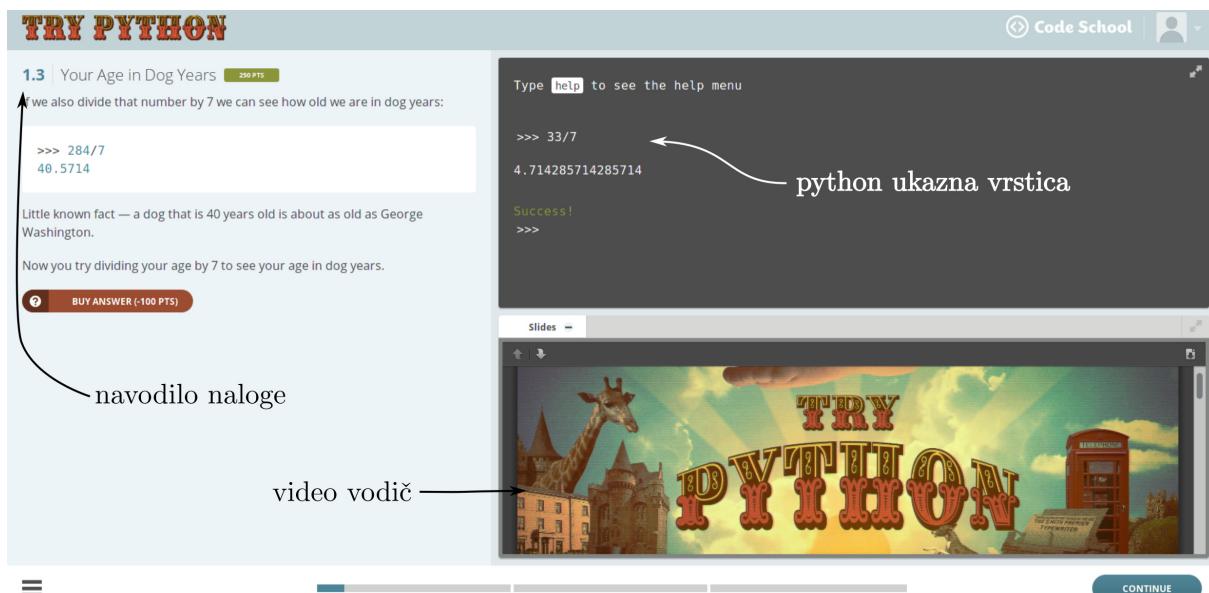
V prejšnjih poglavjih smo opisali **osnove vrste** spletnih portalov. Zanimale nas bodo predvsem **kombinirane vrste**, ki so sestavljene iz osnovnih. Na spletu najdemo številne kombinacije spletnih portalov, **Osnovno kombinirano vsebino** predstavljajo spletni portali, kot je *w3School* (slika 7) [30]. Sestavljeni so iz **tekstovnih vodičev** in **najosnovnejšega preizkusa programske kode**. Vsak primer v vodiču je opremljen s primerom, katerega lahko zaženemo in preizkusimo. Za izvajanje primera pritisnemo na gumb **Preizkus!** (*ang. Try it!*) Programsko kodo primera lahko tudi spremojamo in jo ponovno izvajamo.

**Napredno kombinirano vsebino** predstavljajo spletni portali, ki so sestavljeni **tekstovnega vodiča in/ali video vodičev ter spletnne aplikacije za programiranje**. Omenimo naslednjega



Slika 7: Zaslonska slika spletnne strani *w3School* [30] v poglavju HTML.

predstavnika *Codeschool* [32]. Taki napredni kombinirani vsebini pravimo, da je **vadnica**. V njej je navadno podana razlaga, zastavljen problem oz. naloga, ki ga rešujemo v urejevalniku besedil. V vadnici navadno imamo še preverjanje rešitev, torej sintaktično preverjanje pravilnosti napisane programske kode, ter pravilnost rešitve naloge.



Slika 8: Zaslonska slika spletnne strani *Codeschool* [32].

Različne kombinirane vsebine imajo različne zmožnosti, ki jih bomo spoznali na konkretnem primeru v podrobne pregledu.

## 6.2 Jezik spletnega portala

Ugotovimo lahko, da večina spletnih portalov uporablja **angleščino** kot primarni jezik. Nekateri ponujajo tudi druge jezike, vendar je **slovenščina** zaradi majhnosti le malo krat zajeta, razen v redkih primerih. Angleščina je glavni jezik spleta in računalniške znanosti, zato je pomembno, da učenci oz. dijaki spoznajo tudi angleške izraze in jih seveda povežejo s pravilnimi slovenskimi. Kljub temu, da je učenje v slovenskem jeziku predpisano, lahko vsako učno uro s uporabo spletnih portalov za učenje programiranja povežemo med predmetno z angleščino. Zanimalo nas bo kateri jezik uporablja spletni portali: **angleščina (da/ne)**, **slovenščina (da/ne)**, **drugi (da/ne)**.

## 6.3 Ponujena znanja

Spletne strani za učenje programiranje navadno ponujajo znanja oz veščine programiranja z določenim programskim jezikom. Nekatera svojo ponudbo širijo tako, da ponujajo številne druge projekte, ki združujejo prej naučeno znanje. Na primer izdelava **interaktiven spletne strani**. Za posamezen spletni portal nas bo zanimalo ali ponuja samo:

- **znanja/veščine programiranja** oz. učenje določenega programskega jezika,
- **znanje algoritmov** ali tudi,
- **druga projektna znanja/veščine** (npr. izdelava spletne strani).

## 6.4 Programske jeziki

Zanimalo nas bo katere programske jezike ponuja nek spletni portal. Najbolj pogoste programske jezike smo opisali v poglavju 3.4, v prvi vrsti nas bodo zanimali spletni portali, ki ponujajo naj popularnejše programske jezike in tiste, ki se v izobraževanju uporabljajo pri nas. Prednost bodo imeli tisti, ki ponujajo Python. Večina takšnih spletnih portalov ponuja več programskih jezikov.

## 6.5 Težavnostna stopnja

Vsak spletni portal je namenjen svojemu občinstvu, zato se razlikujejo se tudi po težavnostni stopnji, čeprav govorimo o novincih. Glavna težavnostna razdelitev bo na **osnovo in srednjo šolo**. Po potrebi bomo podrobneje razdelili že osnovno šolo, ki je razdeljena na triletja. Večina spletnih portalov izhaja iz Združenih držav amerike, zato smo povzeli njihove stopnje šolanja (3), saj nekatere strani uporabljajo **K-12** formulacijo za definicijo težavnosti oz. prilagoditev

učnemu načrtu. V podrobнем pregledu bomo ocenili, kateri težavnostni stopnji ustreza spletni portal.

Tabela 3: Primerjava starosti in stopnje šolanja šolskega sistema v ZDA in Slovenskega [33].

Leta	ZDA K-12 naziv	SI Primerjava
6 - 10	<i>Elementary school</i>	<b>1. in 2. triletje OŠ</b>
10 - 14	<i>Middle school</i>	<b>3. triletje OŠ</b>
10 - 14	<i>High school</i>	<b>Srednja šola</b>

## 6.6 Upoštevanj učnih načel

Za uspešno delo in uporabo SPUP v razredu je dobro, da vsebine, ki jih najdemo na spletu sledijo že v samem jedru nekaterim **načelom**, ki jih upoštevamo tudi drugače pri pouku.

**Problemski pristop (Da/Ne)** , zanima nas ali spletni portal, ki ponujajo vsebino uporablja problemsko zasnovane naloge, torej ima v začetku obravnavanja snovi podano oz predstavljeno kateri problem bomo znali na koncu neke vadnice rešiti. Čeprav je vsebina računalniške znanosti in programiranja že po naravi problemsko zasnovana, marsikdaj ni najbolje predstavljeno za kaj je neka stvar dobra.

**Načelo sistematičnosti (Da/Ne)** , lahko potrdimo za tiste spletne portale, kjer so posamezni vsebinski sklopi povezani v nekem logičnem zaporedju. Kot smo že ugotovili pri pregledu spletnih portalov na univerzah je pomembno, da novinci spoznajo nekatere koncepte prej kot druge. Tisti spletni portali, ki bodo imeli neko rdečo nit v povezavi vsebine jih bomo potrdili kot take.

**Načelo postopnosti (Da/Ne)** , pripšemo lahko spletному portalu, ki po podaj snov v posameznem vsebinskem sklopu tako, da bo razlago in program nadgrajeval postopoma in se težavnost stopnjuje.

## 6.7 Uporaba ocenjevanja dosežkov značilnih za igre

V izobraževanju se uveljavlja trend ocenjevanja napredka in dosežkov, ki je tipičen za video igre. To metodo ocenjevanje so poimenovali *ang. Gamification*. Vsako snov ali naloga, ki je v osnovi toga, popestrimo z načinom ocenjevanja tako, da vsako nalogo predstavimo z različnimi izzivi. Vsaki nalogi oz. izzivu sledijo različne nagrade, ki jih učenci zbirajo in jim pravimo dosežki [40].

Dosežki v video igrah so prisotni že vrsto let. Dosežki se razlikujejo po kompleksnosti, vse od zmagoslavne glasbe ob končani stopnji ali igri pa vse do kompleksnega sistema dosežkov, z zbiranjem značk. Značko igralec dobi, ko na primer zbere dovolj predmetov ali razišče

določen procent ozemlja. Poznamo več načinov nagrajevanja dosežkov. Kot smo že omenili lahko dosežke predstavimo kot **značke** ali za posamezne izzive pripravimo sistem točkovanja, ki se jih zbira. Z zbranih točk se lahko sestavijo **lestvice ali uvrstitev**. V razredu slednje morajo biti skrbno načrtovane, da ne pride do prevelikih razlik med učenci in bi, kjer bi se eni lahko počutili nadrejeni in drugi podrejene.

Zanimalo nas bo ali spletni portali, ki učijo programiranja uporabljajo kakršen koli sistem ocenjevanja dosežkov, saj za tiste, ki ga uporabljajo lahko rečemo, da imajo dodaten motivacijski faktor. Zapisali bomo **uporaba dosežkov (da/ne)** in kateri tipi, **značke, lestvice, zbiranje točk za izkušnje, napredovanja** ....

## 6.8 Dodajanje lastnih vsebin

Nekateri spletni portali omogočajo, da pripravimo lastne vsebine, ki jih potem delimo. Navadno je **spletna aplikacija za programiranje** ali **vadnica** razširjena tako, da omogoča sestavljanje programskega nalog. Večina teh je tako, da pripravimo **spremno besedilo, začetni program ali ogrodje programa, končno različico in pomoč ali namig**. Lahko so dodani tudi **testni vhodni in izhodni podatki**. Z podatki, ki smo jih vnesli imamo avtomatizirano nalogo, ki jo lahko posredujemo oz. delimo z novincem. Zanimalo nas bo ali kateri spletni portal omogoča to zmožnost, torej **Dodajanje lastnih vsebin (da/ne)**.

## 6.9 Upravljanje razreda

Zmožnost upravljanja razreda, je velika prednost in lajšanje dela pri administraciji razreda za mentorja. Osnovni način delovanja je naslednji. Mentor ustvari razred ali predmet, podobno kot je to možno pri sistemih spletnih učilnic kot je *moodle* [34] in v učilnico povabi učence. Učitelj s spletno učilnico **spremlja napredek in dosežke posameznega učenca**. Pri nekaterih portalih je **omogočena komunikacija** med mentorjem in novincem. Spletni portali spremeljanje učencev navadno ponujajo kot plačljivo storitve za šole, kar navadno ni najbolj poceni. Zanimalo nas bo ali spletni portal ponuja **upravljanje razreda (da/ne)** in ali je ta storitev **plačljiva ali brezplačna**.

## 6.10 Dostop do gradiv

Veliko vsebin na spletu je brezplačnih in jih pri pouku lahko uporabimo. Mnogo vsebin je tudi plačljivih. Spletni portali, ki imajo plačljive vsebine uporabljajo navadno model **plačevanja naročnine** za dostop do vsebin. Uporabnik more na **letni ali mesečni** ravni odšteti različne

zneske. Nekateri izmed portalov, kot je *Codeacademy* imajo plačljive le nekatere zahtevnejše vsebine in storitve. Drugi portali imajo vso vsebino dostopno za proti plačilo. Obstaja tudi vrsta portalov, kot je *Udemy*, kjer je potrebno plačati za posamezen učni sklop ali temo. Plačljivost dostopa do gradiv lahko razvrstimo na naslednji način, tako da je dostop:

- brezplačen,
- pol plačljiv (*nekatere so brezplačne za druge je potrebno plačati*),
- popolnoma plačljive vsebine.

## 7 Pregled spletnih portalov

V prejšnjem poglavju smo nastavili kriterije, po katerih bomo lažje vrednotili spletnne portale. Preden se lotimo tega opravila, določimo še omejitve, katere spletnne portale bomo dali v ožji izbor in jih pregledali. Te določitve bodo nastavili v mislih uporabe pri pouku v srednji in osnovni šoli. Omejitve za izbor spletnega portala so naslednje:

- spletni portal vsebuje **spletno aplikacijo za programiranje**, katero lahko nastopa tudi samostojno brez vsebine in jo uporabljamo kot **orodje**,
- **vrsta vsebine** naj bo sestavljena z osnovnih vrst oz. naj bo **kombinirana** vrsta vsebine, ki je lahko **osnovna ali napredna** kombiniranih vsebin oz. vsebuje **vadnice**,
- spletni portal ima dosegljivo vsebino **brezplačno ali pol plačljivo**.

### 7.1 Code.org

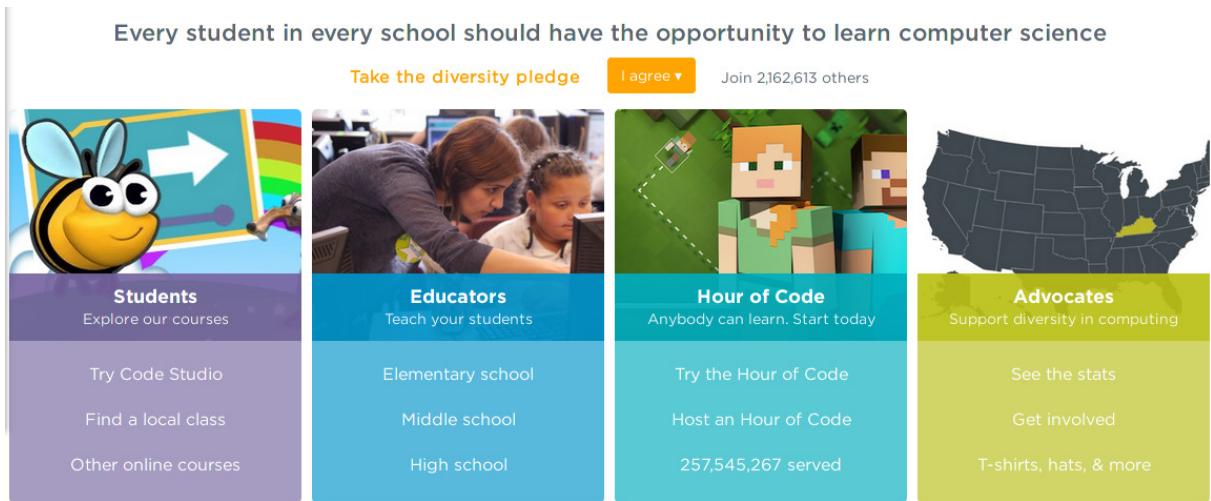
V predstavitvi spletnne strani je predstavljena statistika, ki je povzeta iz raziskave v ZDA. Z nje je razvidno, da bo v prihodnje primanjkovalo kadra za računalniško in informacijsko tehnologijo ter da premalo šol poučuje računalniško znanost [35]. Na *code.org* [37] pravijo, da so neprofitna organizacija, ki se je posvetila širjenju dostopa do poučevanja računalniške znanosti, s poudarkom, ki temelji na ne rasni diskriminaciji in povečanju ženskega spola pri učenju računalništva [36].

Spletni portal je v osnovni sestavljen iz štirih glavnih pod vsebin (slika 9), ena je namenjen **študentom**, druga **učiteljem** in tretja **Uri kode**. Četrти vsebinski sklop je namenjen **promociji drugih spletnih portalov, aplikacijam in strojni opremi**, ki pripomorejo k učenju računalniške znanosti in programiranja ter so del projekta **Ura kode**. Najprej si bomo ogledali slednjega.

#### 7.1.1 Ura kode (*ang. Hour of code*)

**Ura kode** so krajši projekti, ki jih lahko organizacije kot so šole izvedejo v času ene do dveh ur. Celoten projekt je namenjen promociji računalniške znanosti in programiranju. Vsebine, ki so v okviru tega projekta so navadno uvodne vsebine.

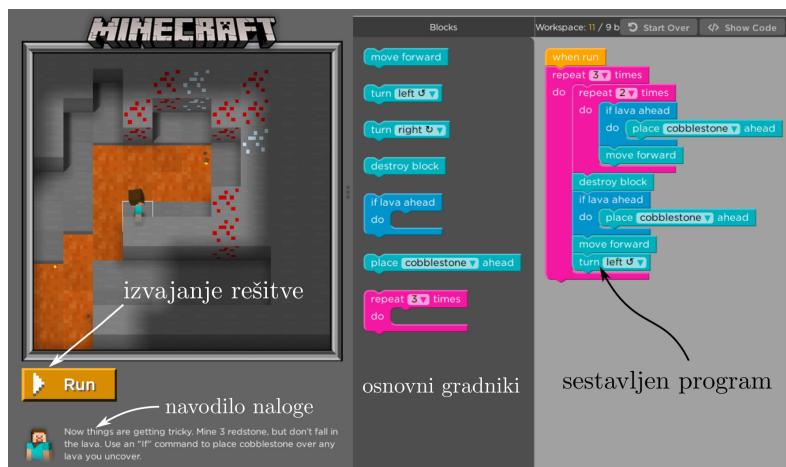
Povezava na portalu **Ura kode** *code.org* nas vodi do zbirke vadnic, ki so del spletnega portala **code**, prav tako najdemo povezave do številnih drugih spletnih portalov in vsebin, ki so vključile v projekt, nekatere smo opisali tudi v nadaljevanju. Vadnice, ki so del portala, ponujajo številne začetne vsebine, ki so grajene na neki znani temi iz sveta računalniških iger ali animiranih filmov.



Slika 9: Zaslonska slika dela začetne spletnne strani *code.org* [37], iz katerega je razvidna razdeljenost vsebin.

Primer vadnice, ki smo si jo izbrali je tematsko povzeta s znane video igre **Minecraft**. V uvodu vsake vadnice najprej sledi video uvod znane osebe, v tem primeru je to glavni razvijalec omenjene igre. V tem uvodnem delu zvemo, zakaj je on postal programer in kaj bomo počeli v tej uri ter kaj se bomo naučili. V predstavitvi je predstavljen tudi uporabniški vmesnik vadnice. Uvodni del predstavitve lahko gledamo kot video ali izberemo zapisan povzetek predstavitve. Snov, ki je razložena v vadnici je **uporaba ukazov, zank in vejitev**. Vadnica je sestavljena iz več enot. Med uvedbo nove snovi sledi spet video predstavitev ali različica v besedilu.

Vadnice so sestavljene z aplikacijo za programiranje **Code studio** (slika 10). Pisanje programske kode v njej je omogočeno z **zlaganje gradnikov** ali načinom *ang. Blockly*, kjer z metodo **vleči in spusti** sestavljamo oz zlepljamo, programsko kodo. Podoben programski jezik je tudi **Scratch**, ki smo ga opisali v poglavju 7.3.



Slika 10: Spletna aplikacija Code studio v kateri so zgrajene vadnice [37].

Pod temi zlepiljenimi gradniki, se ustvarja programska koda v jeziku **JavaScript**. Posamezne

enote gradijo na znanju sistematično in postopoma z večanjem težavnostne stopnje. V Vadnicah so omogočeni samo tisi gradniki, ki jih potrebujemo za rešitev naloge. Večino vsebinskih sklopov je narejeno tako, da v zadnji vadnici vsebinskega sklopa lahko prosto uporabljamo vse gradniki, ki smo se jih naučili uporabljati ter nam ni več potrebno izpolniti konkretnega cilja naloge, da jo opravimo.

### 7.1.2 Code studio

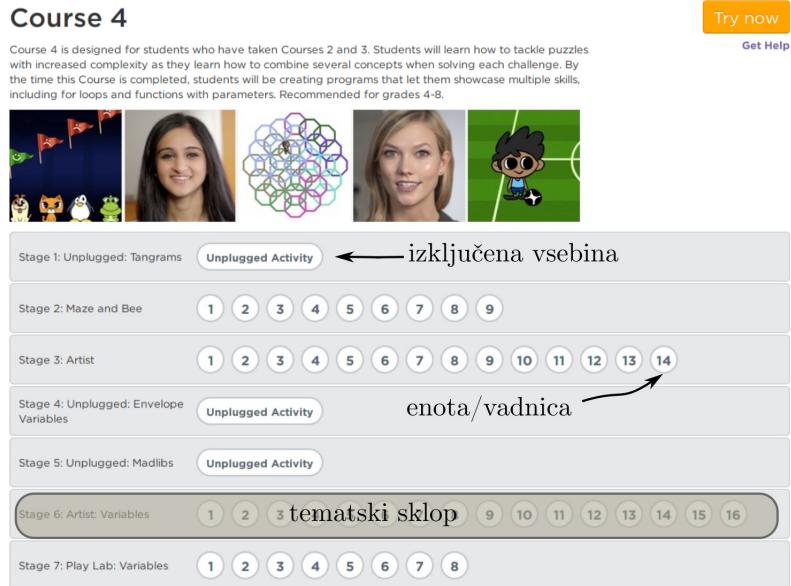
**Code studio** ni samo spletna aplikacija, ki omogoča vadnice in pisanje programske kode temveč je pod stran na kateri novinci najdejo zbrane vsebine sklope (slika 11), ki so razdeljeni po težavnosti, kateri se spreminja spodnja meja starosti, zgornja ostaja ista pri 18 letih. Posebna značilnost spletne strani je tudi ta, da ponuja računalniške vsebine, pri katerih ne potrebujemo računalnika, tako imenovano **računalništvo brez računalnika** ali **izključene vsebine** (*ang. Unplugged lessons*). Te vsebine so navadno predstavljene s predstavitvenimi videi in gradivi, ki jih lahko natisnemo.



Slika 11: Pod stran Code studio za katere lahko nadaljujemo na različne vsebinske sklope [38].

Z klikom na posamezni vsebinski sklop, preidemo na stran s povzetkom. Na tej strani sledimo tudi lastnemu napredku. Vsebine so razdeljene na manjše tematske sklope, ki so spet sestavljeni iz posameznih enot oz. vadnic (slika 12). Med posameznimi tematskimi sklopi najdemo tudi **izključene vaje**, ki jih predelujemo brez računalnika.

Reševanje naloge oz. pisanje programa poteka na podoben način kot smo ga že predstavili pri enotah **Ure kode**, vendar je teh enot več in na vsaki naslednji enoti ponujajo več gradnikov.



Slika 12: Pod stran vsebinskega sklopa [38].

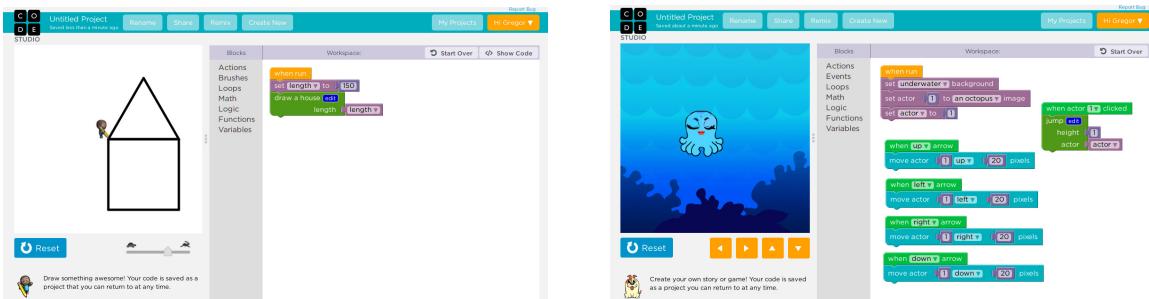
### 7.1.3 Samostojni projekti

Čeprav nekoliko skrito na dnu pod strani **Code studio** najdemo gumb na katerem piše **Moji projekti** (ang. *My projects*). S pritiskom na gumb preidemo na stran na kateri lahko ustvarimo nove projekte, samodejno shranjene projekte ponovno odpremo ali jih izbrišemo. Ustvarimo lahko naslednje nove projekte, **nariši, ustvari igro in ustvari aplikacije**. Ti projekti, so novi in niso omejeni z nobeno vsebino, zato imajo na voljo vse gradnike.

**Nariši** (slika 13a) deluje na podobnem principu kot programski jezik **Logo**. Imamo pisalo, ki pušča sled, mi mu s programsko kodo določamo potek, barvo itd. Lahko ustvarjamo lastne funkcije in uporabljamo že obstoječe, ki jih lahko spremojamo. Takšna je npr. funkcija **nariši hišo**, katere sprejme argument **velikost**. Vsi gradniki, ki jih lahko uporabimo so omejeni izključno na uporabo pisala.

Podobno kot nariši uporabljamo **ustvari igro** (slika 13b). Vendar tu ne uporabljamo pisala, temveč lahko gradimo igre in zgodbe. Na del kjer teče aplikacija lahko vstavljam številne pred nastavljeni figure, ki se odzivajo na dogodke. Sklopi ukazov pri obeh oblikah projektov, risanja in iger so naslednji **akcija, dogodki, zanke, matematike, logika, funkcije in spremenljivke**. Kljub številnim možnostim programske logike so omejitve še vedno prisotne, kot je na primer izbira figur, ki je omejena, saj lastnih figur ne moremo vstavljati. Uporaba teh dveh možnosti je namenjena predvsem osnovni šoli.

Naprednejše možnosti za programiranje smo našli v primeru **ustvarjanja aplikacij** (slika 14). V tej spletni aplikaciji za programiranje se lahko odločamo ali programsko kodo sestavljamo z gradniki ali jo pišemo tekstovno. Glavna značilnost je ta, da poleg pisanja programske kode

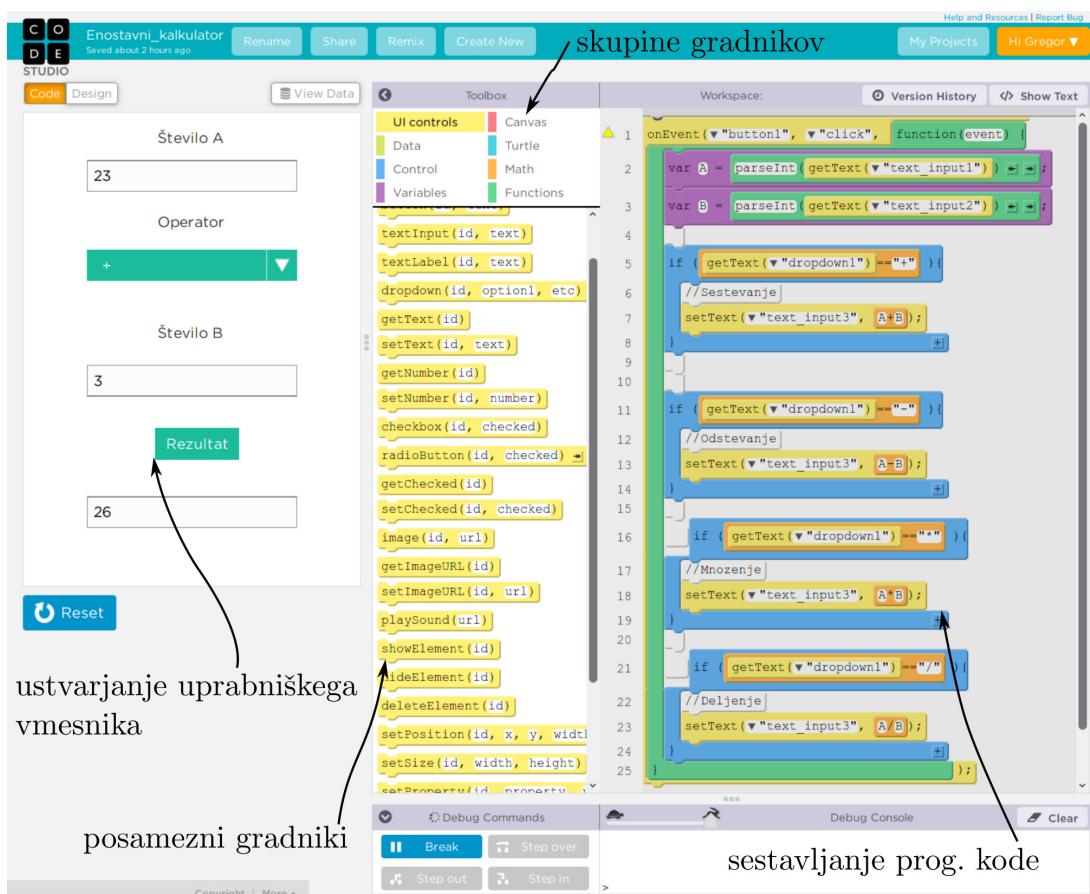


(a) Aplikacija za risanje.

(b) Aplikacija za ustvarjanje iger.

Slika 13: Samostojni aplikaciji za programiranje primerni za osnovno šolo [38].

moramo ustvariti še **uporabniški vmesnik**, ki je omejene velikosti. Sestavljanje uporabniškega vmesnika je podobno sestavljanju mobilnih aplikacij. Z programskim jezikom nismo več tako omejeni in lahko uporabljamo vse vgrajene funkcije JavaScripta. Vgrajena je tudi dodatna možnost ukaznega izpisa in osnovni pripomočki za razhroščevanje, **prekini**, **preskoči**, **vstopi** in **izstopi**.



Slika 14: Spletna aplikacija za programiranje - Ustvarjanje aplikacij. Primer, enostavno računalo [38].

#### 7.1.4 Uporaba za učitelje

Celotna vsebina spletnega portala je zasnovana na ameriškem učnem načrtu. Učitelji na strani najdejo podrobne učne načrte z cilji in idejami za njihovo realizacijo.

#### 7.1.5 Povzetek

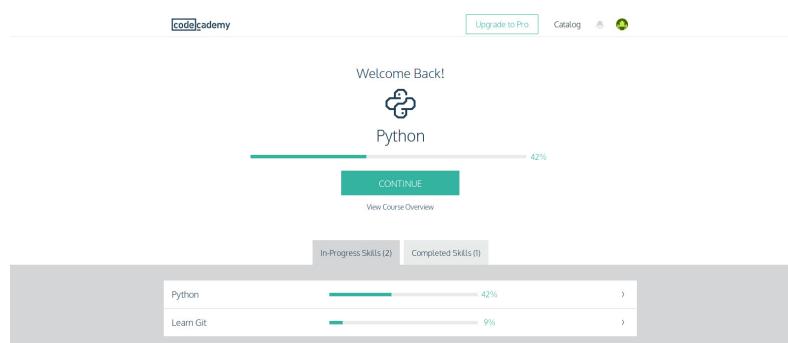
Spletni portal velja za glavnega začetnika pobude **Ure kode** in s tem popularizaciji učenja računalniške znanosti in programiranja. Na portalu najdemo ogromno pripravljenih in razdelanih vsebin. Primerna je predvsem za čiste začetnike OŠ in jo lahko uporabljamo pri najmlajših učencih. Za SŠ so pripravljene vsebine morda prelahke in nekoliko otročje, zato jih v ta namen nebi priporočali, čeprav zgornje omejitve vadnic ni. Vsebinski sklopi so dobro razdelani na podrobne enote in ti postopoma stopnjujejo težavnost. Postopnost je včasih celo pretirana in bi kakšna naloga lahko zajela dve enoti, saj so posamezni koraki ne zahtevni oz. prelahki. Problemska zasnova je tako, da naloge, ki je potrebno rešiti, izvajajo in rešujejo junaki z filmskega oz. sveta video iger. To je pomembno predvsem za motivacijo najmlajših. Nekatere vadnice so prevedene v slovenski jezik, vendar je delež prevedenih zanemarljiv. V splošnem lahko rečemo, da večina spletnega portala ni prevedena. Jezik uporabe spletnega portala pri pouku nebi smel ovirati, saj so naloge razdelane na majhne dele in so navodila enostavna, tako jih lahko učitelj podaja sproti ustno ali jih ima pripravljene na učnih listih po posameznih korakih. Celotna razporeditev na splettem portalu včasih deluje ne strukturirana in neurejena.

Kot samostojne aplikacije v primeru **risanja in ustvarjanja iger** so uporabne, vendar so po zmožnostih dokaj omejene, zato lahko v ta name z večjo svobodo in izbiro uporabljamo Scratch. **Ustvarjanje aplikacij** je primerna predvsem za srednje šole, omogoča pisanje programske kode, prav tako lahko posamezne dele kode nazorno predstavimo grafično z gradniki. Uporabna je predvsem v uvod gradnje aplikacij, saj ustvarjamо uporabniški vmesnik in programsko kodo sami od začetka.

<b>Vrsta vsebine</b>	Napredna kombinirana vsebina: Vadnica (video vodič + navodila(naloga) + spletna aplikacija za programiranje).
<b>Jezik spletne strani</b>	Angleščina: da, slovenščina: da (nekatere vadnice), drugi: ne.
<b>Ponujena znanja</b>	Osnove rač. znanosti in programiranja.
<b>Programski jeziki</b>	Blocky (podobno kot Scratch), JavaScript
<b>Težavnostna stopnja</b>	Osnovna šola (Vadnice, Risanje, Ustvarjanje iger), Srednja šola (Ustvarjanje aplikacij)
<b>Upoštevanje načel</b>	Upošteva načelo sistematičnosti: da, postopnosti: da, problemski pristop: da.
<b>Dosežki/Gamification</b>	Ne.
<b>Dodajanje lastnih vsebin</b>	Ne.
<b>Upravljanje razreda</b>	Ne.
<b>Dostop vsebin</b>	Brezplačno.

## 7.2 Codeacademy

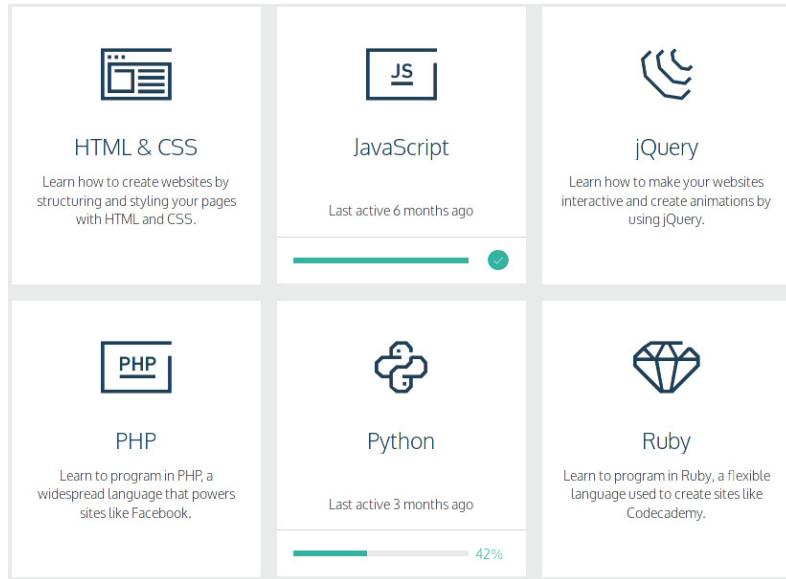
Spletni portal je tipični predstavnik novo nastalih portalov za učenje programiranja. Sami o sebi pravijo, da so ameriško podjetje, ki se ukvarja z izobraževanjem. Njihov tim se z ustvarjanjem spletne strani *Codeacademy* ter se uči in poučuje, saj želijo ustvariti najboljšo spletno izobraževalno izkušnjo za prihodnost, ki domuje na spletu [39]. Po registraciji in prijavi nas čaka naslednja spletna stran (slika 15). Z začetne, nadzorne strani lahko izbiramo nov vsebinski sklop ali nadaljujemo z že začetimi.



Slika 15: Zaslonska slika spletne strani *Codeacademy* [39]. Začetna, nadzorna stran po prijavi, od tu nadaljujemo na vsebinske sklope, ki smo jih že začeli.

**Jezik** spletne strani je **angleščina**, drugih jezikov ni mogoče izbrati. Če še podrsamo po spletni

strani navzdol, najdemo vsebinske sklope, ki učijo programske jezike in so naslednji: **HTML + CSS**, **JavaScript**, **JQuery**, **PHP**, **Python**, **Ruby** in že v prejšnjem odseku so bile na voljo osnove **Java**.



Slika 16: Zaslonska slika spletnne strani *Codecademy* [39]. Seznam znanj/veščin programskih jezikov, ki jih ponuja spletni portal.

Nad zbirkom osnovnih programskih jezikov najdemo druga **ponujenih znanj**. V tem delu najdemo nekatere vsebine kot je na primer, učenje **SQL**, uporaba **ukazne vrstice** ali uporabo spletnega orodja za kontrolo verzije **GIT**. Nekatere vsebine so sestavljene kot projekti, taki sta na primer **Naredi spletno stran** ali **Naredi spletno stran interaktivno**. Strnemo lahko, da spletni portal ne ponujajo le znanja in veščine **programiranja in programskih jezikov**, temveč tudi druga znanja. S klikom na želeno vsebino pridemo na stran (slika 17), s katere lahko nadaljujemo tam kjer smo ostali ali pregledujemo posamezne teme, ki smo jih že opravili ali tiste, ki nas še čakajo.

Razvidno je, da so teme sistematično razporejene, zato lahko ugotovimo da je **načelo sistemičnosti upoštevano**. Z pritiskom na gumb za nadavaljevanje (*ang. Continue*) odpremo urejevalnik (slika 18) na temi in pod enoti na kateri smo ostali.

Vsaka tema vsebinskega sklopa je razdeljena na več enot oz. vadnic, ki so sestavljene tako, da postopoma dograjujejo program. Delo, ki smo ga opravili v predhodni vadnici se samodejno prenese naprej, ko je to potrebno. Lahko povemo, da je **načelo postopnosti upoštevano**. Pri nekaterih tematskih sklopih sledi najprej ponovitev že naučenega. Na primer pri Temi *Seznamy in funkcije* najprej sledi pregled osnovnega upravljanja z seznamami in pisanjem funkcij. Uporabniški vmesnik (slika 18) je urejen tako, da na desni strani imamo podano snov, ki je sestavljena s **primerom določene programske strukture in navodili**, kaj moramo dograditi v programu. Na levi strani imamo **urejevalnik besedil** v katerega pišemo programsko kodo.

Want more practice and review? Upgrade for the complete experience.

8 Projects    9 Quizzes    1 Final Project

**UNIT 1: PYTHON SYNTAX**

- Lesson: Python Syntax
- Lesson: Tip Calculator
- Quiz: Python Syntax

**UNIT 2: STRINGS AND CONSOLE OUTPUT**

Slika 17: Zaslonska slika pod strani spletne strani *Codecademy* [39] na kateri lahko pregledujemo posamezne teme in nadaljujemo tam, kjer smo ostali.

**For the Record**

Excellent. Now you need a hard copy document with all of your students' grades.

```
animal_sounds = {
    "cat": ["meow", "purr"],
    "dog": ["woof", "bark"],
    "fox": [],
}
print(animal_sounds["cat"])
```

The example above is just to remind you how to create a dictionary and then to access the item stored by the "cat" key.

**Instructions**

for each student in your students list, print out that student's data, as follows:

- print the student's name
- print the student's homework
- print the student's quizzes
- print the student's tests

**script.py** Urejevalnik besedila

```
lloyd = {
    "name": "Lloyd",
    "homework": [90.0, 97.0, 75.0, 92.0],
    "quizzes": [88.0, 40.0, 94.0],
    "tests": [75.0, 90.0]
}
alice = {
    "name": "Alice",
    "homework": [100.0, 92.0, 98.0, 100.0],
    "quizzes": [82.0, 83.0, 91.0],
    "tests": [80.0, 77.0]
}
tyler = {
    "name": "Tyler",
    "homework": [87.0, 93.0, 75.0, 82.0],
    "quizzes": [98.0, 92.0, 85.0, 91.0],
    "tests": [100.0, 100.0]
}
students = [lloyd, alice, tyler]
for i in students:
    print("Name: " + i["name"])
    print("Homework: " + str(i["homework"]))
    print("Quizzes: " + str(i["quizzes"]))
    print("Tests: " + str(i["tests"]))
```

Izpis programa.

Programska koda prenešena s prejšnje enote.

Dodan del programa, zahtevan v navodilih enote.

Shrani in oddaj programsko kodo.

Slika 18: Zaslonska slika *Codecademy* [39] vadnice, ki jo sestavlja urejevalnika z navodili in oknom za izpis v programu.

Urejevalnik zna barvati programsko kodo in samodejno predviditi zamike besedila. V zgornjem desnem kotu je **okno za izpis** v katerem se izpisujejo **izhodni podatki** s programa in napake sintakse **Python tolmača**. Spodaj je gumb za **Shrani in oddaj programsko kode** (*ang. Save and Submit Code*).

Vsaka v uvodnem delu predstavi novo problematiko, ki jo potem čez posamezne enote rešujemo. Vsebina je predstavljena **problemško**. Samo delo z **vadnico** poteka tako, da napišemo program v **spletno aplikacijo za programiranje**, ki je zahtevan v navodilih. Ko menimo, da imamo pravilno rešitev pritisnemo na gumb **Shrani in oddaj programsko kode**. Program najprej preverja **sintaktično** pravilnost. Napako program vrne nad gumbom za oddajo programske kode, podrobna napaka **Python-ovega tolmača** se izpiše v **oknu za izpis**. Zatem sledi **semantično** preverjanje pravilnosti rešitve naloge. V posamezni enoti program samodejno vrši osnovno preverjanje programa, z točno določenim rezultatom. Dokler test napisanega programa ne da pravega rezultata me moremo nadaljevati na naslednjo enoto. Ko se nam zatakne, spletna stran ponuja **forum** na katerem najdemo odgovor ali lahko postavimo vprašanje. Forum je razdeljen na posamezne teme in enote, tako da lahko hitro najdemo zahtevano vprašanje.

Za uspešno premagovanje enot je uporabnik nagrajen z **značkami**, ki so vidne na strani njegovega profila. Na tej strani se beležijo tudi predelane vsebine. Spletni portal torej uporablja nagrajevanje z **dosežki**.

Spletni portal ponuja tudi plačljive vsebine, čeprav je osnova vsebinskih sklopov brezplačna. Za dostop do plačljivih storitev, za ponujajo model zakupa z naročnino za ceno **19\$/mesec**. Za naročnino uporabnik pridobi dostop do **naslednjih dodatnih storitev**:

- personaliziran učni načrt;
- dostop do kvizov;
- dostop do realnih projektov;
- dostop pomoči v živo preko sporočil.

Ugotovimo, da je spletni portal **pol plačljivi** in da dodatne storitve, ki jih ponuja spletni portal niso potrebne za uporabo pri pouku, saj vse omenjene dodatne storitve lahko zagotovi učitelj.

### 7.2.1 Uporaba za učitelje

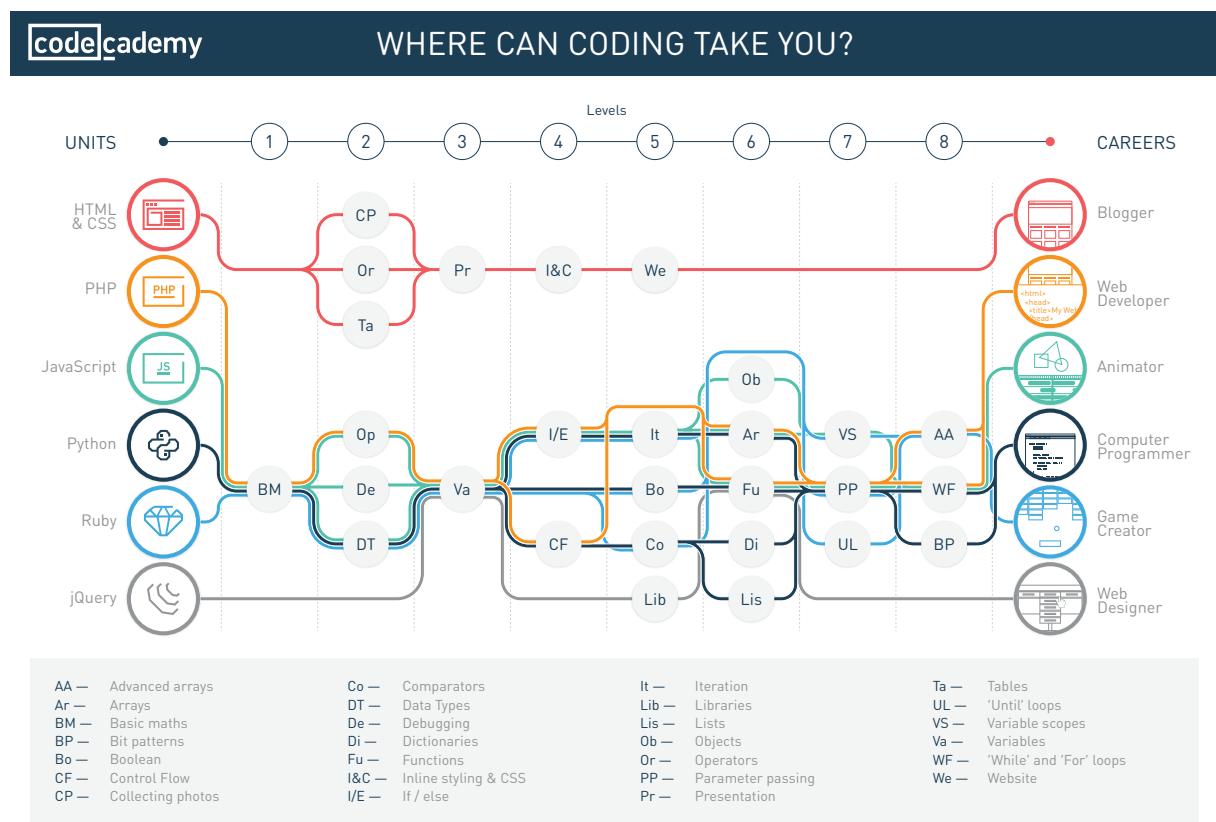
Spletni portal nudi vsebine, ki so prilagojene za šole in uporabo v šolah. V pregledu lahko ugotovimo, da za srednješolske profesorje ponujajo naslednje:

- **trening za učitelje**, ki je možen le v ZDA;
- **gradiva**;

- sledenje napredku učencem;
- ura kode (ang. *Hour of code*).

Pod stran z **gradivi** profesorju ponuja razdelane posamezne teme na enote podobno kot so razdelane na glavni strani. Pod vsako enoto profesor lahko preizkusi, rešuje vaje. Večina enot ima pripravljene kvize, ki jih profesor lahko prav tako preizkusi in se pripravi na učno uro.

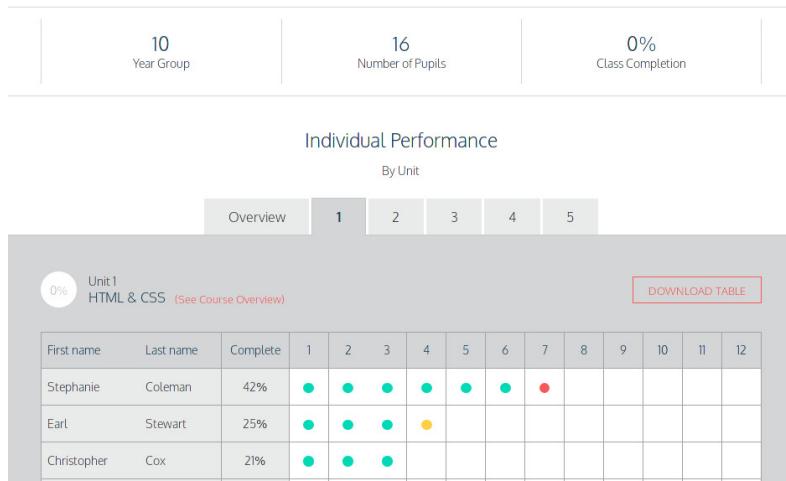
Kot so si zamislili avtorji spletnega portala so tematski sklopi in posamezne enote med seboj prepletene in vodijo do posameznega cilja. Zemljevid (slika 19) prikazuje povezavo enot na posamezni stopnji in različne cilje. Za vsak tematski sklop učitelj lahko prenese **pregled posamezne teme**, v kateri so podrobnejše zapisani učni cilji in so označene stopnje ki sovpadajo z zemljevidom.



Slika 19: Plakat z povezavami enot različnih tematskih sklopov po stopnjah, ki vodijo do posameznih ciljev (ang. *Level prograsion mapa*) [39].

Ena izmed naprednih zmožnosti spletnega portala je **sledenje napredku učencem** (ang. *Students tracking*). Ta omogoča, da profesor dijakom ustvari račune in jih povabi na spletnem portalu v razred. Določi tematske sklope katerim bo sledil. V pregledu (slika 20) profesor lahko opazuje napredek posameznega dijaka po enotah ali v povzetku za celotni napredek. Napredek je prikazan z pikami različnih barv, ki predstavljajo procente napredka pri posameznem tematskem sklopu. V tem primeru **ne moremo** govoriti o **upravljanju razreda**, saj omogočeno le sledenje napredku in ne tudi komunikacija med profesorjem in dijakom, prav

tako ni mogoče dodajati lasnih enot oz. nalog. Pri samem ustvarjanju razreda, je profesorju delo zelo olajšano, saj portal omogoča, da informacije o dijakih ustvarjalec razreda kopira direktno s programa podobnega kot je **Excel**. V tabeli so podani podatki **ime**, **priimek**, **skupina**, **uporabniško ime**. Dijaki za dostop do spletnega portala uporabijo uporabniško ime, ki si ga izmisli učitelj ali oni sami, geslo je skupno za celotno učilnico. Z prejetim uporabniškim imenom in gesлом se dijaki na spletni strani registrirajo z svojim email naslovom. Če so na portalu že registrirani profesorju posredujejo uporabniško ime in jih ta doda v ustvarjen razred. Dijak mora povabilo potrditi.



Slika 20: Zaslonska slika *Codeacademy* [39]. Prikazuje tabelo za sledneje napredku učencem.

## 7.2.2 Povzetek

*Codeacademy* [39] ima dobro razdelano vsebino, ki je na nekaterih delih dokaj poglobljena. Sistematičnost, postopnost in problemski pristop sta prav tako dobro zastavljeni. Portal ponuja številne projektne vsebine in učenje programskih jezikov. Izbor teh je tak, da ustreza današnjim spletnim tehnologijam in zahtevam. Znanje se omejuje predvsem na učenje programiranja ter da se to znanje zna uporabiti v praktične namene. Spletni portal ima nekatere vsebine in zmožnosti plačljive, ampak ima zadostno število brezplačnih vsebin, ki omogočajo normalno učenje.

Zanemarjeno je znanje **Računalniške znanosti**, saj se med spoznavanjem programskih jezikov in podatkovnih struktur ne uči različnih algoritmov. Uči se bolj uporabo posameznih funkcij, ki so vgrajene v programske jezike. Slaba stran spletnega portala je ta, da je ves v *angleškem jeziku*. Poleg tujega jezika so nekatera navodila napisana dokaj kompleksno in zahteva že dobro poznavanje razumevanja sporočil tolmača in semantičnih napak, ki se zgodijo v programu.

Spletni portal je v primeru učenja spletnih tehnologij, kot je **HTML/CSS** je primeren za učence

zadnje triade osnovne šole, saj se ti omenjeno snov učijo pri izbirnem predmetu **Računalniška omrežja**. V primeru učenja programskega jezika **Python** spletni portal ponuja zahtevna znanja in je primeren predvsem za srednje in višje šole.

Da bi mentor lahko spletni portal uporabljal pri pouku bi moral imeti prevode navodil za posamezno temo in enoto. Vsekakor je možno izvesti kot *praktično voden delo*. Spletni portal se lahko uporablja za domače delo in smo uporabo kot takega opisali v poglavju 8.2. Mentor lahko, spletni portal priporoča v uporabo, kot za neobvezno dopolnilno dejavnost tistim, dijakom, ki želijo razširiti znanje programiranja. Opozorimo, da jim lahko priporoča le brezplačne vsebine. Čeprav je možno, pa vseeno morda ni primerno uporabljati spletni portal tako, da bi z njim predelali celoten vsebinski sklop in bi ga pri pouku uporabljali kot edino orodje, saj je poučevanj računalniške znanosti na njem omejeno. Lahko ga pa s pridom uporabimo kot dodatek pri učenju programskega jezika, kot je na primer **Python**, ki v nadaljevanju služi kot orodje za poučevanje računalniške znanosti. Pri pouku smo omejeni tudi s spremjanjem vsebine, ki je ni mogoče prilagoditi k učnemu načrtu tako kot bi to žeeli, zato moramo učno pripravo prilagoditi uporabi spletnega portala. Poleg prednosti in slabosti lahko zaključimo, da ima spletni portal še dodatno motivacijsko vrednost, saj njegova sistematičnost in postopnost ter nagrajevanje z dosežki motivira uporabnike, da imajo željo po dokončanju vsebinskega sklopa, ki ga obravnavajo.

Codeacademy | [www.codeacademy.com](http://www.codeacademy.com)

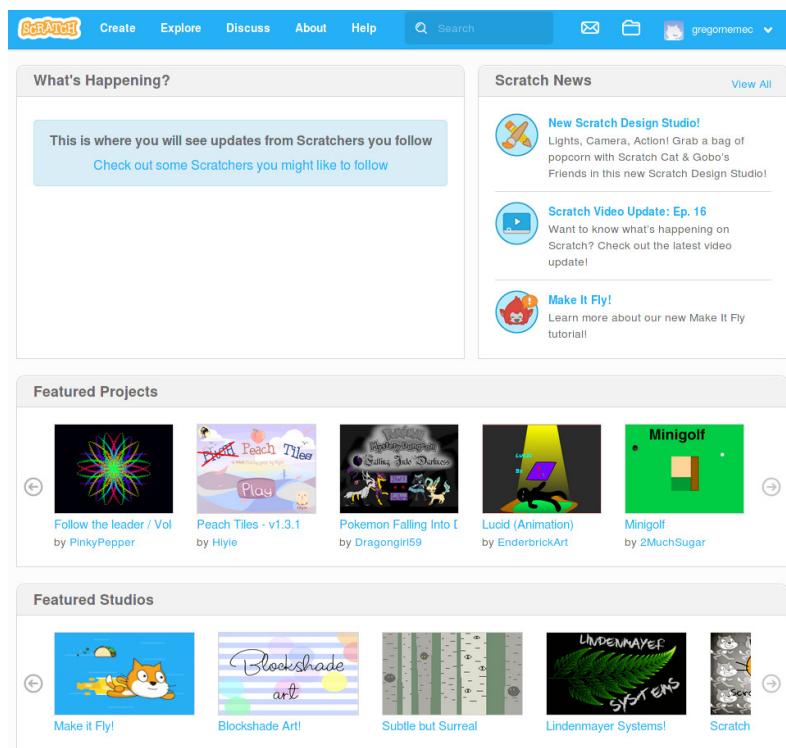
<b>Vrsta vsebine</b>	Napredna kombinirana vsebina: Vadnica (primer + navodilo(vodič) + spletna aplikacija za programiranje).
<b>Jezik spletnne strani</b>	Angleščina: da, slovenščina: ne, drugi: ne.
<b>Ponujena znanja</b>	Znanje prog. jezikov, druge vsebine.
<b>Programski jeziki</b>	<b>HTML+ CSS, Java JavaScript, Jquery, PHP, Python, Ruby</b>
<b>Težavnostna stopnja</b>	3/3 Osnovna šola, Srednja šola.
<b>Upoštevanje načel</b>	Upošteva načelo sistematičnosti: da, postopnosti: da, problemski pristop: da.
<b>Dosežki/Gamification</b>	Da (značke).
<b>Dodajanje lastnih vsebin</b>	Ne.
<b>Upravljanje razreda</b>	Da, ustvarjanje razreda in sledenje napredku učencem.
<b>Dostop vsebin</b>	Pol plačljiv (plačljivi so projekti, kvizi, podpora v živo).

## 7.3 Scratch

Spletni portal *Scratch* [41] je spletna različica zelo popularnega programskega jezika **Scratch**. Scratch je pri nas popularen predvsem v **osnovnih šolah** in je zamenjal dolgo uporabljen **Logo**.

Razvoj samostojne namizne različice Scratcha se je končala pri verzi 1.4, od tu naprej je razvoj Scratcha potekal za spletno različico. Spletna različica je narejena na osnovi zaprttega **Adobe Flash**. Za poganganje Scratch v spletnem brskalniku potrebujemo vtičnik **Flash**. Ko prvič naložimo spletni portal *Scratch* (slika 21) lahko ugotovimo, da ponuja naslednje funkcionalnosti:

- ustvarjanje programov (Orodje Scratch);
- deljenje ustvarjenih programov;
- raziskovanje narejenih programov, drugih uporabnikov;
- forum za diskusije;
- pomoč pri uporabi



Slika 21: Zaslonski posnetek glavne strani *Scratch* [41].

Scratch smo po vrsti vsebine umestili med **spletne aplikacije za programiranje** oz. smo ga predstavili kot samostojno **orodje**. Lastne vsebine, ki bi v širšem smislu poučevala računalniško znanje ne najdemo. Vse vsebin, ki so na strani, so namenjene učenju uporabe orodje in spoznavanjem zmožnosti programskega jezika. Govorimo lahko vseeno o spletnem portalu,

saj ta ima vse za uspešno uporabo orodja in omogoča vso funkcionalnost, ki jo potrebuje neka spletna skupnost.

### 7.3.1 Uporaba Scratcha

Scratch omogoča ustvarjanje animacij, predstavitev in iger. Namenjen je 8 do 16 let starim, vendar ne predstavlja nobene omejitve na zgornji meji starosti. Preveden je v številne jezike med njimi je tudi **slovenščino** [42].

Če smo v preteklosti že uporabljali namizno različico Scratcha, nam uporaba spletne različice (slika 22) nebo predstavljal nobenih težav, saj je postavitev uporabniškega vmesnika zelo podobna kot je bilo to v namizni verziji. Osnovni princip delovanja je tak, da na *oder* (slika 22) postavljamo različne *like*. Vsak lik, ki ga dodamo z knjižnice ali ga naložimo sami, je predstavljen kot svoj objekt in vsakemu posebej dodajamo programsko kodo, ki jo sestavljamo iz različnih *gradnikov*. V samem orodju lahko dorisujemo k že obstoječim likom, rišemo nove, ali jih naložimo neposredno iz računalnika. Dodajamo lahko tudi zvok, ki ga posnemamo sami ali ga izberemo iz knjižnice zvokov. Programsko kodo lepimo skupaj oz. sestavljamo podobno kot bi sestavliali kocke. Kot smo že spoznali takemu načinu sestavljanja programske kode pravimo tudi "*Blocky*". Gradniki so oblikovani tako, da se sklopijo samo tisti, ki se med sabo lahko povežejo.



Slika 22: Zaslonska slika Scratch [41].

**Vodiči** v Scratchu najdemo na desnem robu (slika+ 22). Vodiči so sestavljeni tako, da uporab-

nika postopoma vodijo skozi gradnjo programa. S tem je zagotovljena **postopnost**. Posamezen korak v vodiču je sestavljen iz besedila in animiranega poteka dela.

### 7.3.2 Deljenje in raziskovanje projektov

Vsak uporabnik, ki se registrira na spletnem portalu ima dostop do svojega profila. Na svoji strani se samodejno shranjujejo projekti, ki smo jih izdelovali in jih od tu lahko ponovno naložimo za urejanje. Vse projekte lahko delimo z drugimi. Vsak projekt ima pod stran na kateri določamo nastavitev za *deljenje* (slika 23). Na strani lahko dodamo *navodila za program* in *zapiske in zasluge*, prav tako določamo, če želimo projekt deliti ali ne.



Slika 23: Pod stran za deljenje projekta na *Scratch* [41].

Če je omogočeno da lahko delimo vsebine je na spletnem portalu tudi dobro poskrbljeno za **raziskovanje projektov** drugih uporabnikov. Pod zavihkom *razišči* (*ang. Explor*) najdemo številne projekte, ki so razporejeni v kategorije. Tu lahko črpamo številne ideje in si najljubše projekte shranimo tudi na svojem profilu.

### 7.3.3 Povzetek

**Učni načrt** v slovenski osnovni šoli ne obveznega izbirnega predmeta **Računalništvo** je prilagojen prav za uporabo Scratcha, kot je razvidno iz ciljev. Čeprav z dobro pripravljenimi vodiči spoznavamo tudi na primer vejitve, si učitelj mora pripraviti in prilagoditi vsebino za uresničevanja učnega načrta sam. Kot smo že lahko ugotovili, na spletu in na sploh ne zmanjka literature in idej, ki učitelju pomagajo pri uresničevanju učnega načrta s Scratchem. Tudi na spletni strani Scratcha lahko črpamo številne ideje iz projektov drugih uporabnikov.

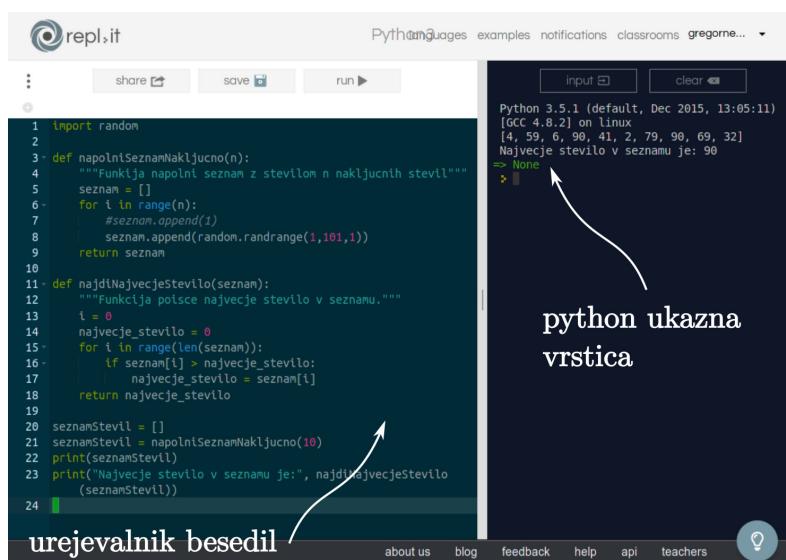
Za slabost spletnih različic štejemo, da je narejen z zaprto tehnologijo **Adobe Flash**, kar pomeni, da si uporabnik mora naložiti vtičnik Flash na spletni brskalnik. Vtičnik Flash uradno podpira le nekatere komercialne operacijske sisteme in njegovo vlogo zamenjuje vedno boljše zmožnosti samih spletnih brskalnikov z tehnologijo **HTML5 + CSS + JS**.

## Scratch | <https://scratch.mit.edu>

Vrsta vsebine	Spletna aplikacija za prog.
Jezik spletne strani	Scratch: angleščina: da, slovenščina: da, drugi: da. Spletni portal: angleščina: da, drugi: ne
Ponujena znanja	Vodič za spoznavanje uporabe Scratch in pomoč
Programski jeziki	Scratch
Težavnostna stopnja	Osnovna šola
Upoštevanje načel	Problemski pristop: ne, sistematičnost: ne, postopnost: da (Vodič).
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Da, vendar v smislu ustvarjanja lastnih projektov in ne celovitih vadnic.
Upravljanje razreda	Ne.
Dostop vsebin	Brezplačen.

## 7.4 Repl.it

**Repl.it** [43] je še ena **spletna aplikacija za programiranje**. Podjetje, ki spletno stran ustvarja je tržno osredotočeno na ponujanje **aplikacijski programski vmesnik - APV** ali (*ang. application programming interface - API*). Njihov APV uporabljajo številni spletni portali kot je na primer <https://www.freecodecamp.com> [44] in nekateri drugi plačljivi spletni portali. Na svoji strani ponujajo prost dostop do spletnne aplikacije za programiranje (slika 24).



The screenshot shows the repl.it web interface. On the left, there is a code editor window containing Python code. The code defines two functions: `napolniSeznamNaključno` which generates a list of random integers, and `najdiNajvecjeStevilo` which finds the maximum value in a list. The code editor has line numbers from 1 to 24. On the right, there is a terminal window showing the output of running the script. The terminal window has tabs for 'Input' and 'clear'. The output shows the Python version (3.5.1), the command used to run it (GCC 4.8.2 on linux), the generated list of numbers [4, 59, 6, 90, 41, 2, 79, 90, 69, 32], and the result of calling `najdiNajvecjeStevilo` on that list, which is 90. A callout arrow points from the text "python ukazna vrstica" to the terminal window.

```

1 import random
2
3 def napolniSeznamNaključno(n):
4     """Funkcija napolni seznam z stevilom n naključnih stevilk"""
5     seznam = []
6     for i in range(n):
7         #seznam.append(i)
8         seznam.append(random.randrange(1,101,1))
9     return seznam
10
11 def najdiNajvecjeStevilo(seznam):
12     """Funkcija pošče največje stevilo v seznamu."""
13     i = 0
14     najvecje_stevilo = 0
15     for i in range(len(seznam)):
16         if seznam[i] > najvecje_stevilo:
17             najvecje_stevilo = seznam[i]
18     return najvecje_stevilo
19
20 seznamStevil = []
21 seznamStevil = napolniSeznamNaključno(10)
22 print(seznamStevil)
23 print("Največje stevilo v seznamu je:", najdiNajvecjeStevilo
      (seznamStevil))
24

```

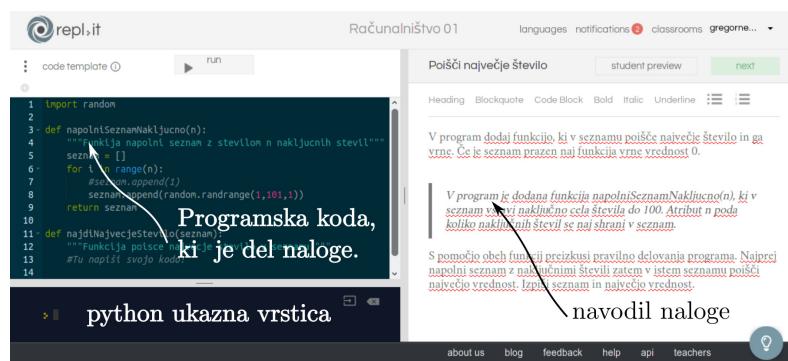
Slika 24: Zaslonska slika spletnne aplikacije za programiranje *repl.it* [43].

Aplikacija ponuja številne programske jezike kot je **Python3**, **Ruby**, **JavaScript**, **HTML**, **CSS**, **C#**, **JAVA** in še mnoge druge. Vsako nova pojava spletne aplikacije predstavlja novo sejo. Po registraciji na spletnem portalu lahko shranujemo posamezne seje. Posamezno sejo lahko **tudi delimo** preko spletne povezave. **Urejevalnik besedil** omogoča barvanje rezerviranih besed programske kode in ima napredno **možnost ponujanja predlogov** za samo dokončevanje izpisa rezerviranih besed in funkcij programskega jezika.

#### 7.4.1 Ustvarjanje razredov in nalog

Spletni portal skorja, da ni vreden omembe z **vsebinskega vidika** tudi kot spletna aplikacija me predstavlja nekih posebnih funkcij, ki jih nebi imele tudi nekatere druge spletnne aplikacije. Ima pa spletni portal veliko zmožnost, saj omogoča **ustvarjanje razredov**. Mentor lahko ustvarja razred in v njega povabi dijake ter samodejno dodaja oz. **ustvarja lastne naloge** oz. **vadnice** (slika 25). Dijke v razred povabi z dodajanjem email-ov v seznam. Dijakom ni potrebna registracija. S povezavo, ki so jo dobili na email se prijavijo v razred. Mentor v načinu ustvarjanja naloge, doda lastna navodila in začetno programsko kodo. V naslednjem koraku se mentor lahko odloči ali bo pravilnost naloge preverjal sam ali po dodal avtomatski preizkus programske kode. V avtomatskem načinu mora podati primer vhodnih podatkov in rezultat izhoda.

V razredu imajo dijaki vpogled v seznam nalog. Naloge se dijakom prikazane z navodili. Dijke rešujejo nalogo in ko so zadovoljni s svojo rešitvijo, nalogo oddajo. Mentorju se status naloge pri dijaku spremeni na *oddano* in sedaj mentor nalogo lahko pregleda, poda komentar in jo označi kot *opravljeno* ali jih s sporočilom tudi *zavrne*.



Slika 25: Zaslonska slika pogleda mentorja v načinu priprave naloge [43].

#### 7.4.2 Povzetek

Spletna aplikacija za programiranje, ponuja mentorjem računalniških vsebin osnovno orodje za ustvarjanje razredov in nalog ter komunikacijo z dijaki. Pri tem uporablja osnovne zmožnosti

brez pretirane kompleksnosti. Sam spletni portal ne ponuja nobene vsebine, kar je svoje vrstna prednost, saj lahko mentor prilagodi naloge učnemu načrtu in to v svojem jeziku.

#### Repl.it | <https://repl.it/>

Vrsta vsebine	Spletna aplikacija za prog.
Jezik spletne strani	angleščina: da, slovenščina: ne, drugi: ne.
Ponujena znanja	Ne ponuja nobene vsebine
Programski jeziki	Python3, Ruby, JavaScript, HTML, CSS, C#, JAVA in drugi
Težavnostna stopnja	Srednja šola.
Upoštevanje načel	Problemski pristop: ne, sistematičnost: ne, postopnost: ne (Vodič).
Dosežki/Gamification	Ne.
Dodajanje lastnih vsebin	Da. Ustvarjanje razreda in nalog ter komunikacija z dijaki.
Upravljanje razreda	Da.
Dostop vsebin	Brezplačen.

## 7.5 Tutorials point

*Tutorials point* [45] je eden izmed velikih portalov, ki ponujajo obsežne in zahtevne vodiče tehničnih in ne tehničnih vsebin. Na spletu je vodičev veliko, tega smo izpostavili iz razloga, ker ponuja ogromno **vsebin programiranja in računalniške znanosti**, vsi primeri v vodičih imajo **možnost preizkusa** in razvito imajo lastno **spletno aplikacijo za programiranje**, ki ima veliko zmožnosti, nekatere značilne za namizne **IDE**.

Spletni portal ponuja knjižnico vodičev (26) različnih vsebinskih sklopov. Z slike je razvidno, da je zares obsežna.

Vsebinsko si smo pregledali vodič za **Python3** (slika 27). Vodiči so oblikovani tako, da na desni strani najdemo kazalo vsebine, v sredinskem delu je razložena snov z primeri.

Nekatere od primerov lahko tudi preizkusimo, z klikom na gumb **Preizkusi (ang. Try it)**, se nam na isti strani odpre podokno (slika 28). Kodo v urejevalniku lahko spremojamo in ponovno zaženemo.

Tutorials Library			
Free Online Tutorials & Courses			
JAVA TECHNOLOGIES	PROGRAMMING	WEB DEVELOPMENT	SCRIPTS
<ul style="list-style-type: none"> <li>• Learn Apache Ant</li> <li>• Learn Apache POI (Powerpoint)</li> <li>• Learn Apache POI (Word)</li> <li>• Learn Apache POI</li> <li>• Learn AWT</li> <li>• Learn Design Patterns</li> <li>• Learn EasyMock</li> <li>• Learn Eclipse</li> <li>• Learn EJB</li> <li>• Learn Guava</li> <li>• Learn Hibernate</li> <li>• Learn IBATIS</li> <li>• Learn Jackson</li> <li>• Learn JasperReports</li> <li>• Learn Java XML</li> <li>• Learn Java</li> <li>• Learn JBossM5</li> </ul>	<ul style="list-style-type: none"> <li>• Learn Apex</li> <li>• Learn Assembly</li> <li>• Learn Awk</li> <li>• C Standard Library</li> <li>• Learn COBOL</li> <li>• Learn C++</li> <li>• Learn C</li> <li>• Learn Computer Programming</li> <li>• Learn C by Examples</li> <li>• Learn C#</li> <li>• Learn Clojure</li> <li>• Data Structure &amp; Algorithms</li> <li>• Learn D</li> <li>• Learn Erlang</li> <li>• Learn Euphoria</li> <li>• Learn F#</li> <li>• Learn Fortran</li> </ul>	<ul style="list-style-type: none"> <li>• Learn Ajax</li> <li>• Learn AngularJS</li> <li>• Learn Angular Material</li> <li>• Learn ASP.NET</li> <li>• Learn Aurelia</li> <li>• Learn BackboneJS</li> <li>• Learn Bootstrap</li> <li>• Learn CSS</li> <li>• Learn Codeigniter</li> <li>• Learn CoffeeScript</li> <li>• Learn CPanel</li> <li>• Learn Drupal</li> <li>• Learn Django</li> <li>• Learn EmberJS</li> <li>• Learn ExtJS</li> <li>• Learn Flask</li> <li>• Learn Flex</li> </ul>	<ul style="list-style-type: none"> <li>• Learn JavaScript</li> <li>• Learn jQuery</li> <li>• Learn jQueryUI</li> <li>• Learn Lua</li> <li>• Learn Perl</li> <li>• Learn PHP</li> <li>• Learn PHP-7</li> <li>• Learn Python</li> <li>• Learn Python-3</li> <li>• Learn PyQt</li> <li>• Learn WxPython</li> <li>• Learn Ruby</li> <li>• Learn RSpec</li> <li>• Learn Sed</li> <li>• Learn Tcl/Tk</li> <li>• Learn Unix</li> <li>• Learn VBScript</li> </ul>

Slika 26: Del seznama oz knjižnica vodičev, ki ga ponuja spletna stran *Tutorials point* [45].

The screenshot shows a section titled "Single Statement Suites" from a Python 3 tutorial. It includes a code snippet and its execution result. The code is:

```
#!/usr/bin/python3
var = 100
if ( var == 100 ) : print ("Value of expression is 100")
print ("Good bye!")
```

The execution result shows the output: "Value of expression is 100" followed by "Good bye!". Navigation links at the bottom include "Previous Page", "Print", "PDF", and "Next Page".

Slika 27: Zaslonski izrez vodiča za Python3. Slike je razvidno kazalo in gumb za Preizkus! [45].

The screenshot shows a coding environment with a toolbar labeled "SIMPLY EASY LEARNING Python 3.x". The code in the editor is:

```
1 #!/usr/bin/python3
2
3 var = 100
4
5 if ( var == 100 ) : print ("Value of expression is 100")
6
7 print ("Good bye!")
```

The "Result" panel shows the output: "Value of expression is 100" and "Good bye!".

Slika 28: Pod okno za preizkus primera programske kode [45].

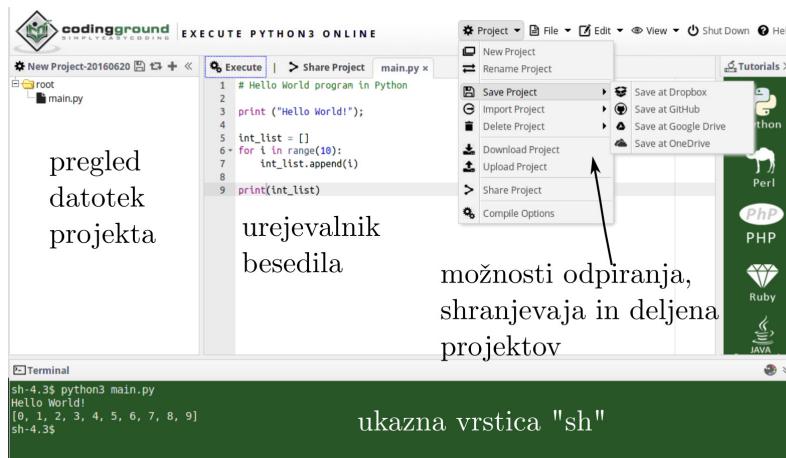
### 7.5.1 Coding ground

Kot smo že omenili spletni portal kot orodje ponuja lastno spletno aplikacijo za programiranje. Spletna aplikacija se imenuje *Codingground* [46]. Na uvodni strani orodja (slika 29) smo odkrili, številnost ponujenih **programskih jezikov**, ki sovpadajo s številnimi vodiči, ki jih ponuja spletni portal.

 Java 8	 Java MySQL	 JS javascript	 JSP JSP	 jQuery	 Julia
 Ksh Shell	 LATEX Latex	 Lisp	 LOLCODE	 Lua	 Matlab/Octave
 Malbolge	 Markdown	 MathML	 Mozart-OZ	 Nimrod	 NodeJS
 Objective-C	 OCaml	 Pascal	 PARi/GP	 Pawn	 Perl
 Perl MySQL	 PHP	 PHP MySQL	 Web View	 Pike	 Processing.js
 p5.js	 Prolog	 Python	 Python-3	 Python MySQL	 Rexx

Slika 29: Del seznama različnih programskih jezikov katere lahko uporabljamo z spletno aplikacijo za programiranje *Codingground* [46].

Razporeditev spletne aplikacije (slika 30) je tako, da na levem robu imamo seznam datotek v korenskem imeniku, v sredinskem delu je urejevalnik besedil, nad urejevalnikom najdemo menijsko vrstico in na dnu strani je **ukazna vrstica**, v kateri lahko zaganjamo napisano programsko kodo. V njej se izpisujejo tudi povratne informacije tolmača in izhod programske kode. **Urejevalnik besedil** omogoča barvanje kode rezerviranih besed in nastavljanje barvne sheme urejevalnika. Urejevalnik omogoča še samodejno zamikanje programske kode, ko je to potrebno. **Shranjevanje in uvažanje projektov v oblak**, lahko smatramo kot eno izmed večjih zmožnosti te spletne aplikacije. *Codingground* lahko nastavimo, da se poveže s oblačnimi shrambami kot so **Dropbox**, **Google Drive**, **Onedrive** in s sistemom za objavljanje, upravljanje verzij in kolaboracijo **Git**. Seveda lahko projekt naložimo neposredno z računalnika in ga seveda tja tudi shranimo. Prednost oblačnega shranjevanja je ta, da na programiramo lahko od koder koli in z katerim orodjem želimo. če je to spletna aplikacija ali namizna. Spletna aplikacija omogoča tudi upravljanje z datotekami. Lahko ustvarimo, preimenujemo in brišemo datoteke ali imenike. To lahko počnemo z **menija (file)** ali **ukazne vrstice**. Vsak projekt lahko delimo preko neposredne kratke **url povezave**, kot smo to že videli pri drugih spletnih portalih.



Slika 30: Spletna aplikacija za programiranje - *Codingground* [46].

### 7.5.2 Povzetek

Vsebina vodičev je zelo tehnična in deluje kot okrnjen povzetek uradne reference za določen programski jezik. Kot tako je predvsem primerna za programerje začetnike, ki se želijo poučiti o določenem programskem jeziku, vendar že poznajo osnovne koncepte programiranja. Velik plus je vsekakor preizkus programske kode. Vodiče lahko priporočimo kot skrajšano verzijo reference programskemu jeziku.

S pravo nastavitevijo, spletna aplikacija omogoča, da dijaki imajo programsko kodo in snov, ki jo v nekem trenutku predelujejo povsod na voljo. S pomočjo shranjevanja in deljenja projektov lahko mentor uporabi spletno aplikacijo kot glavno orodje za učenje računalništva in programskega jezika. Mentor mora pripraviti sistem za izmenjavo navodil, programske kode, in rešitev dijakov. To lahko storiti uporabo kratkih url povezav. Urejevalnik besedil bi lahko ponujal kakšno zmožnost več kot jo, vendar zadosti osnovnim potrebam pisanja programske kode.

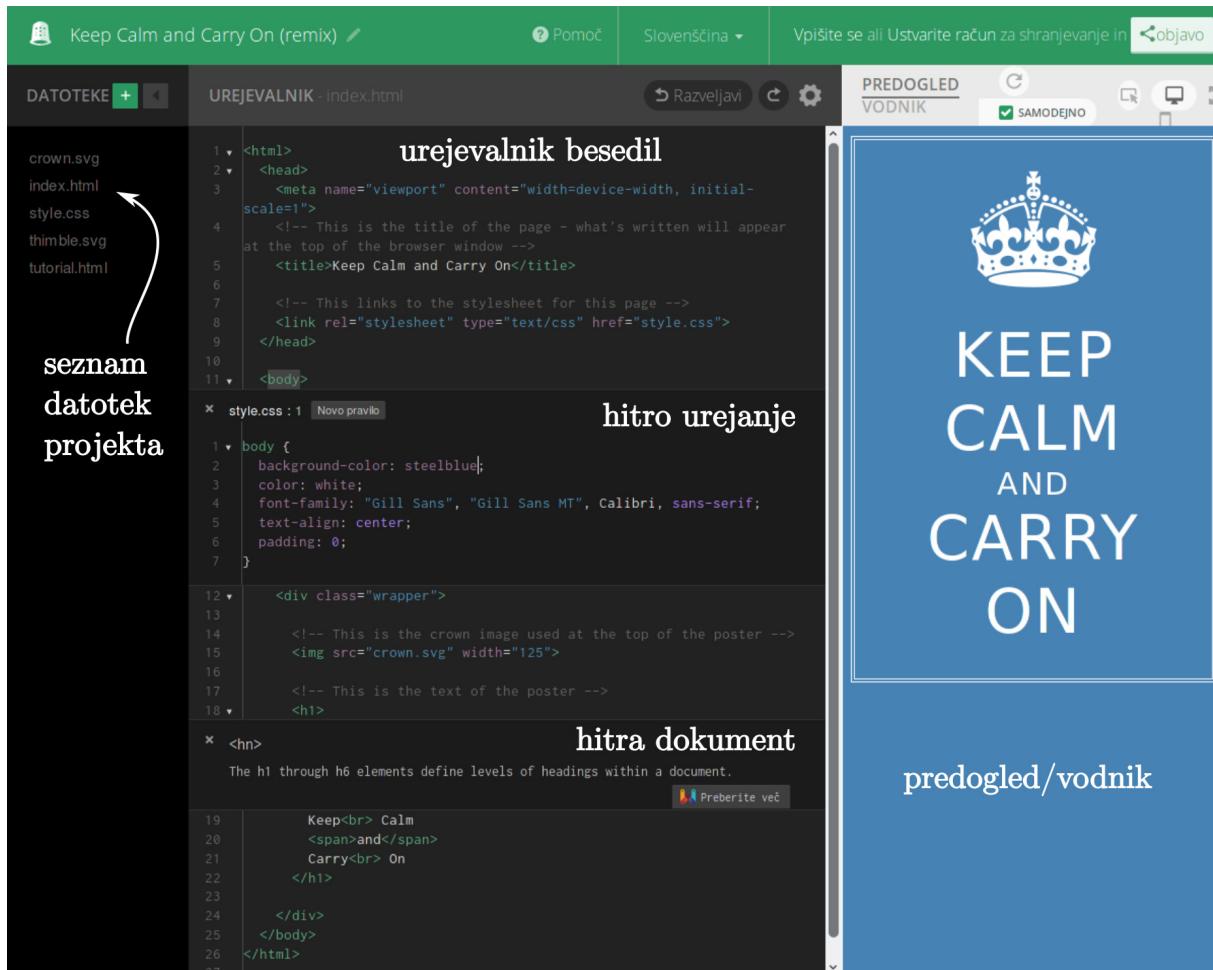
Tutorialspoint | <http://www.tutorialspoint.com/>

<b>Vrsta vsebine</b>	Osnova kombinirana vsebina: vodič in preizkus programske kode. Posebej spletna aplikacija za učenje programiranja: Codingground
<b>Jezik spletnne strani</b>	angleščina: da, slovenščina: ne, drugi: ne.
<b>Ponujena znanja</b>	Znanja prog. Jezikov + druge vsebine. Vodiči s številnih področij.
<b>Programski jeziki</b>	Velika knjižnica prog. Jezikov
<b>Težavnostna stopnja</b>	Srednja šola.
<b>Upoštevanje načel</b>	Problemski pristop: ne, sistematičnost: ne, postopnost: da (Vodič).
<b>Dosežki/Gamification</b>	Ne.
<b>Dodajanje lastnih vsebin</b>	Da. Ustvarjanje podporne programske kode v spletni aplikaciji za prog., vendar brez navodil in deljenje vsebine.
<b>Upravljanje razreda</b>	Na.
<b>Dostop vsebin</b>	Brezplačen.

## 7.6 Thimble

*Thimble* [49] je spletni portal, ki ga gosti podjetje **mozilla**, katero izdaja spletni brskalnik **Firefox**. Spletni portal je namenjen učenju spletnih tehnologij **HTML**, **CSS** in **Java Script**. Pripravljenih je šest spletnih vsebin, ki jih lahko odpremo v projektu in jih preurejamo. Največja

prednost spletnega portala je spletna aplikacija oz. orodje v katerem urejamo projekte (slika 31).



The screenshot shows the Thimble web editor interface. On the left, there's a sidebar titled "DATOTEKE" with files listed: "crown.svg", "index.html", "style.css", "thimble.svg", and "tutorial.html". A callout arrow points from this sidebar to the text "seznam datotek projekta". The main area is titled "UREJEVALNIK - index.html". It contains two tabs: "urejevalnik besedil" and "hitro urejanje". The "urejevalnik besedil" tab shows the following HTML code:

```

1 <html>
2   <head>
3     <meta name="viewport" content="width=device-width, initial-
4       scale=1">
5     <!-- This is the title of the page - what's written will appear
6       at the top of the browser window -->
7     <title>Keep Calm and Carry On</title>
8
9     <!-- This links to the stylesheet for this page -->
10    <link rel="stylesheet" type="text/css" href="style.css">
11  </head>
12
13  <body>
14    background-color: steelblue;
15    color: white;
16    font-family: "Gill Sans", "Gill Sans MT", Calibri, sans-serif;
17    text-align: center;
18    padding: 0;
19
20    <div class="wrapper">
21      <!-- This is the crown image used at the top of the poster -->
22      
23
24      <!-- This is the text of the poster -->
25      <h1>
26        Keep<br> Calm
27        <span>and</span>
28        Carry<br> On
29      </h1>
30
31    </div>
32
33  </body>
34
35 </html>

```

The "hitro urejanje" tab shows the CSS code for the body:

```

body {
  background-color: steelblue;
  color: white;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, sans-serif;
  text-align: center;
  padding: 0;
}

```

To the right of the editor is a preview window titled "PREDOGLED VODNIK" which displays the final "Keep Calm and Carry On" poster. Below the preview is the text "predogled/vodnik".

Slika 31: Urejanje projekta na strani *Thimble* [49].

Na desni strani spletnne aplikacije imamo **seznam dokumentov**, ki sestavljajo projekt. V projekt lahko dodajamo lastne dokumente. **Urejevalnik besedil** barva značke html in css jezika, prav tako upošteva barvanje besedila v primeru programskega jezika JavaScript. Urejevalnik ima dve priročni zmožnosti, kot je *hitri dokument* in *hitro urejanje*. Hitri dokument je takojšnja pomoč, ki se sproži ob pritisku kombinacije Alt + K na mestu kjer je značka za katero želimo dodatno razlago. Hitro ureja na mestu, kjer smo postavljeni v besedilu z pritiskom kombinacije tipk Alt+E odpre pod urejevalnik z css razredom, ki oblikuje značko dela html dokumenta. Funkcija hitro urejanje v css dokumentu, ko smo postavljeni na barvo, pomeni barve s barvne palete.

Vse spremembe, ki jih naredimo v urejevalniku se v živo in samodejno posodobijo v **oknu predogleda**, ki se nahaja na levem delu. Na tem mestu najdemo tudi vodnika. Uvodna spletna stran in uporabniški vmesnik sta preveden v **slovenščino**. Žal pa vodniki, ki so del pripravljeni na spletni strani niso prevedeni v slovenščino.

Če na spletni strani opravimo registracijo in se prijavimo, se nam spremembe na projektu shranjujejo samodejno. Projekt lahko preimenujemo in ga delimo z drugimi preko spletne povezave. Tisti, ki naš projekt odpre ga spreminja kot lastnega in se sprememb v našem ne pozna. Ustvarjamо lahko tudi nove projekte, katerim dodajamo `html`, `css`, `js` datoteke. Dodamo lahko tudi, datoteko vodiča, ki jo lahko poljubno spreminjamо.

#### 7.6.1 Povzetek

Spletni portal lahko uporabljamо na vseh stopnjah. Primeren je še posebej za uporabo v osnovni šoli pri izbirnem predmetu **Računalniška omrežja**, saj omogoča enostaven in učinkovit urejevalnik besedil. Učitelj se registrira in pripravi oz prilagodi obstoječi projekt za pouk. Učencem deli povezavo. Učencem se ni potrebno registrirati in kljub temu lahko urejajo dokument kot lastne. Učenci, svoje dokončane projekte delijo kot končen izdelek s povezavo nazaj učitelju. Na podoben način se lahko spletni portal uporablja tudi v srednji šoli.

Thimble   <a href="https://thimble.mozilla.org">https://thimble.mozilla.org</a>	
<b>Vrsta vsebine</b>	Napredna kombinirana vsebina: Vadnica (Vodnik + spletna aplikacija za programiranje).
<b>Jezik spletne strani</b>	angleščina: da, slovenščina: aplikacij, da; vodnik, ne. drugi: da.
<b>Ponujena znanja</b>	Znanje programskih jezikov, uporaba spletna in spletnih vsebin.
<b>Programski jeziki</b>	HTML, CSS, JavaScript.
<b>Težavnostna stopnja</b>	Osnovno šolo (3. triado) in srednjo šolo.
<b>Upoštevanje načel</b>	Problemski pristop: da, sistematičnost: ne, postopnost: ne
<b>Dosežki/Gamification</b>	Ne.
<b>Dodajanje lastnih vsebin</b>	Da. Možno je ustvarjanje lastnih vadnic, ki jih lahko delimo naprej.
<b>Upravljanje razreda</b>	Na.
<b>Dostop vsebin</b>	Brezplačen.

#### 7.7 Code combat

Spletni portal *Code combat* [47] je mešanica med igranjem igre in pisanjem programske kode. V predstavitvi spletnе strani pravijo naslednje, “*Če se želiš naučiti programirati, moraš napisati veliko programske kode*” in poudarjajo, da oni poskrbijo da pri tem početju ostane

zabava v ospredju [48]. Spletna stran ponuja tri načine registracije, ustvarite lahko **navaden**, **učiteljski** ali **učencev**, račun. V pregledu strani smo uporabili prijavo z navadnim računom, v nadaljevanju smo prav tako povzeli posebnosti ostalih dveh računov.

Po registracij in prijavi v račun si izberemo **lik in programske jezike** s katerim bomo igrali (slika 32). Spletna igra ponuja štiri programske jezike **Python**, **JavaScript**, **CoffeeScript**, **LUA**.



Slika 32: Izbira junaka in programskega jezika [47].

Po izbiri junaka preidemo na izbor **zemljevidov** (slika 33). Izberemo lahko samo zemljevid, ki je odklenjen. Druge zemljevide odklenemo tako, da rešimo vse naloge v njem. Posamezen zemljevid predstavlja cilje posameznih programskih konceptov, ki se jih uporabnik nauči, ko predela vse naloge.



Slika 33: Izbor zemljevida na katerem bomo reševali naloge [47].

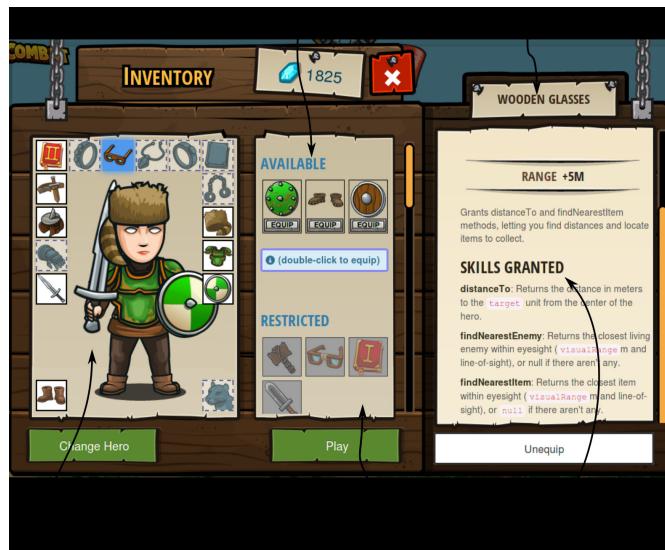
Igra se zgleduje po tipu iger igranja vloge (*ang. Role play game - RPG*). Z junakom napredujemo po zemljevidu z vsako opravljeno nalogo, ob koncu vsake naloge prejmemo **točke - izkušnje** in napredujemo v lastnih **stopnjah**. Junak nabira številne predmete, ki mu omogočajo nadgradnjo veščin in tako lažje napredovanje skozi misije. Za začetek naloge pritisnemo

na rdeče obarvan krog na zemljevidu (slika 34), prikaže se povzetek naloge in katere koncepte bomo uporabili pri reševanju naloge.



Slika 34: Podroben zemljevid za izbiro nalog [47].

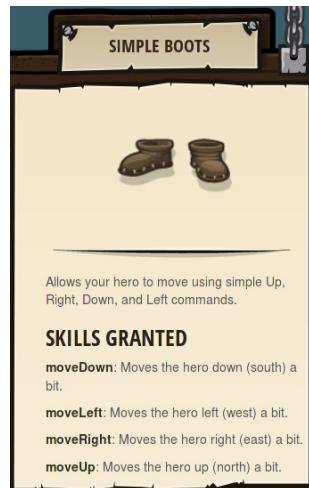
Sledi opremljanje junaka (slika 35). Igra tu pomaga v tolikšni meri, da omeji nekatere predmete, ki niso uporabni za trenutno naložbo. Seveda sledimo logiki igre in izbiramo veno najboljšo opremo, ki je na voljo.



Slika 35: Oprema junaka in njen opis [47].

Ko kažemo ta primer igre smo z napredkom rešenih nalog skoraj na polovici drugega zemljevida in so oprema in večine, ki jih uporablja naš junak že napredne, zato naredimo primerjavo med prejšnjo in nadgrajeno opremo. Z primerjave škornjev (slika 36a in 36b) lahko lahko povzamemo katere metode je junak pridobil. Če je pri *enostavnih škornjih* imel možnost gibanja le v smeri **levo, desno, gor in dol** se lahko pri *usnjeneh škornjih* giblje po najkrajši poti

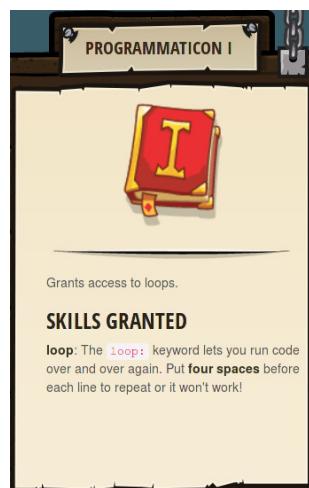
na koordinate, ki jih podamo kot argument metode `hero.moveXY(x, z)`. S primerjavo *knjige za programiranje* med verzijo *I in II* (slika 36c in 36d), ki smo jo pridobili kasneje je razlika med veščinami očitna. Če smo pri *knjigi za programiranje I* lahko uporabljali samo zanke loop: oz. `while True:`, pri *knjigi za programirane II* lahko zraven še uporabljam `if/else` stavek. Primerjali smo samo dva predmeta, junaku so na voljo, številni predmeti z različnimi metodami za različne dele telesa, vse od **mečev, ščitov, kap, ur, pasa, obleke, očal** in tako dalje. Z napredovanjem veščin in naborom predmetov junak pridobiva na zmožnostih, prav tako se s tem postopno izboljšujejo veščine in se širi znanje uporabniku spletne igre.



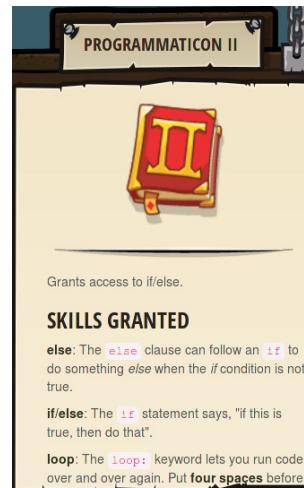
(a) Enostavni škornji (*ang. Simple boots*)



(b) Usnjeni škornji (*ang. Lether boots*)



(c) Knjiga programiranja I



(d) Knjiga programiranja II

Slika 36: Primerjava med prejšnjo različico opreme in njeno nadgradnjo, ki jo lahko zamenjamo junaku [47].

Po vsakem zagonu igre sledi najprej prikaz cilja, ki ga moramo uresničiti. Postavitev igre (slika 37) je tako, da nalogo rešujemo v **urejevalniku besedil**, ki je na desni strani zaslona. Programska koda, ki jo izvajamo se odvija v oknu na levi strani. Urejevalnik besedil omo-

goča nekatere napredne funkcije, kot je **barvanje kode**, **samodejno zamikanje vrstic**, **prikaz zamika vrstic**, **sprotno opozarjanje na napačno sintakso** ter **predlogi za samodejno dokončevanje** programske kode. Pri samem pisanju programske kode lahko zapišemo na primer samo del metode, kot je `find` in se nam ob potrditvi samodejnega predloga izpiše celotna programska koda `enemy = hero.findNearestEnemy()`.



Slika 37: Postavitev igre [47].

V trenutni nalogi je cilj tak, da moramo napadati sovražnike, ko je ta bližje skrinji kot *10m* in ga moramo napasti z metodo `cleve`, če ga sovražnika ni v bližini napadamo skrinjo. Ko smo zadovoljni s svojo rešitvijo poženemo program in čakamo na končan izid (slika 38). Če je iztek programa uspešen in smo rešili nalogo, jo *posredujemo*.



Slika 38: Uspešno končan izid igre s napisano programsko kodo [47].

Ob koncu igre poberemo še **dosežke**, to so **točke izkušenj**, **diamante** in **značke** (slika 39) .



Slika 39: Končni rezultat in pregled nad dobljenimi dostžki ob koncu igre[47].

Dostop do spletnega portala pa ni povsem brezplačen. Z **brezplačnim dostopom** lahko raziščemo 145 nalog v petih zemljevidih. Spletni portal ponuja **naročnino** 10\$mesec, s katero lahko pridobimo dodate **naloge, junake, diamante** in tako dalje.

### 7.7.1 Upravljanje razreda

Spletni portal omogoča **upravljanje razredov**. Razrede upravlja **učitelj**. Portal ima prilagojeno učno snov za tri stopnje po ameriškem **K-12** sistemu. Kot smo že primerjali šolske sisteme lahko povemo, da so stopnje po starosti v slovenski šoli prilagojene na naslednje stopnje **osnovno šolo (2. triado in 3. triado) in srednjo šolo**. Upravljanje razredov (slika 40) je podobno kot smo to videli pri **Code academy** (poglavlje 7.2.1). Učitelj ima nadzor nad dodajanjem učencev in ima pregled o napredku učencev. Z njimi preko portala ne more komunicirati.

The screenshot shows the 'Računalništvo 01' class overview page. It includes:

- Class Overview:**
  - Language: Python
  - Students: 1
  - Average level playtime: a minute
  - Total play time: 3 minutes
  - Average levels completed: 2.0
  - Total levels completed: 2
  - Created: 6/20/2016
- Adding students:**
  - DarkSpoonFoot (button)
  - Copy Class Code (button)
  - New students can enter this class code on their dashboard or visit [codecombat.com/courses](https://codecombat.com/courses) to join the class.
- Course Selection:**
  - Earliest incomplete level: Course 1, Level 3: Shadow Guard (jureneme)
  - Latest completed level: Course 1, Level 2: Gems in the Deep (jureneme)
  - Select course to view: Introduction to Computer Science (dropdown menu with options: Introduction to Computer Science, Computer Science 2, Computer Science 3, Computer Science 4, Computer Science 5)
- Student Progress:**
  - Export Student Progress (CSV) button
  - Add Students Manually button
  - Students tab (selected)
  - Course Progress tab
  - Enrollment Status tab
  - Sort by: Name Progress
  - Student list: jureneme (jureneme@gmail.com)
  - Progress bar: 1 through 20, with the first three segments filled (green, yellow, grey).

Slika 40: Učiteljev pogled na upravljanje razreda [47].

Dostop za **šole ni brezplačen**. Učitelj lahko zahteva demonstracijsko različico in v njo povabi neomejeno število učencev. V tej demo različici je na voljo samo prvi tečaj **Uvod v računalniško znanost**, vsi ostali so zaklenjeni. Za uporabo nadaljevalnih tečajev mora učitelj zaprositi za poizvedbo cene za nakup licence za posameznega učenca.

Učenec rešuje naloge podobno kot smo to lahko videli pri navadnem računu, vendar ne more stopenj izbirati iz mape. Naloge oz. stopnje, ki jih rešuje so prilagojene tečaju v katerega ga

je vpisal učitelj. Po opravljeni nalogi, učenec takoj nadaljuje na naslednjo stopnjo kot je ta predvidena v tečaju. Učenec ima vpogled na naloge, ki ga čakajo v tečaju. V seznamu lahko izbere tiste naloge, ki jih je že opravil oz. tisto zadnjo v kateri je ostal. Za nadaljevanje mora učenec rešiti prejšnjo nalogu. Če v navadnem računu lahko menjujemo različne dele oblačil in opreme, je to v načinu učenčevega načina onemogočeno. Učenčev lik ima navojo stvari, ki jih potrebuje pri rešitvi naloge. S to omejitvijo je olajšano delo učitelja saj se tako lahko razred osredotoči le na reševanje naloge.

Spletni portal ima prevode v več večjih svetovnih jezikov, žal trenutno slovenščina ni med njimi. V padajočem meniju, kjer učenci lahko spreminjače jezike se najde tudi slovenščino, vendar se ob njenem izboru prikaže pojavno okno z pozivom za prevod v izbran jezik. Uporabniku se ponudi spremenjen račun, ki omogoča prevajanje v druge jezike. Torej obstaja neko upanje, da se bo mogoče nekoč nekdo lotil tega projekta in začel prevajati spletni portal in naloge v slovenski jezik.

#### 7.7.2 Povzetek

Spletni portal oz. spletna igra ima vsekakor veliki motivacijski faktor. Naloge so narejene sistematično in postopno. Če prav se v naših osnovnih šolah ne priporoča učenje programskih jezikov, kjer pišemo programsko kodo razen morda **Logo**. Naj bi se uporabljali vizualni programski jeziki kot je **Scratch**. Za ta spletni portal verjamem, da bi ga lahko uporabili pri osnovno šolcih, kater bi lahko poučevali **Python**. Težava je edino, kot vedno **slovenščina**, saj na strani še ni prevoda nalog in **plačljivost** za uporabo razredov. To učitelj sicer lahko zaobide z uporabo **navadnih računov** in uporabo brezplačnih vsebin, vendar mu to uteži delo ali se zaprosi za demonstracijsko različico in v ta name izkoristi vsebine, ki so na voljo brezplačno.

## Codecombat | <https://codecombat.com/>

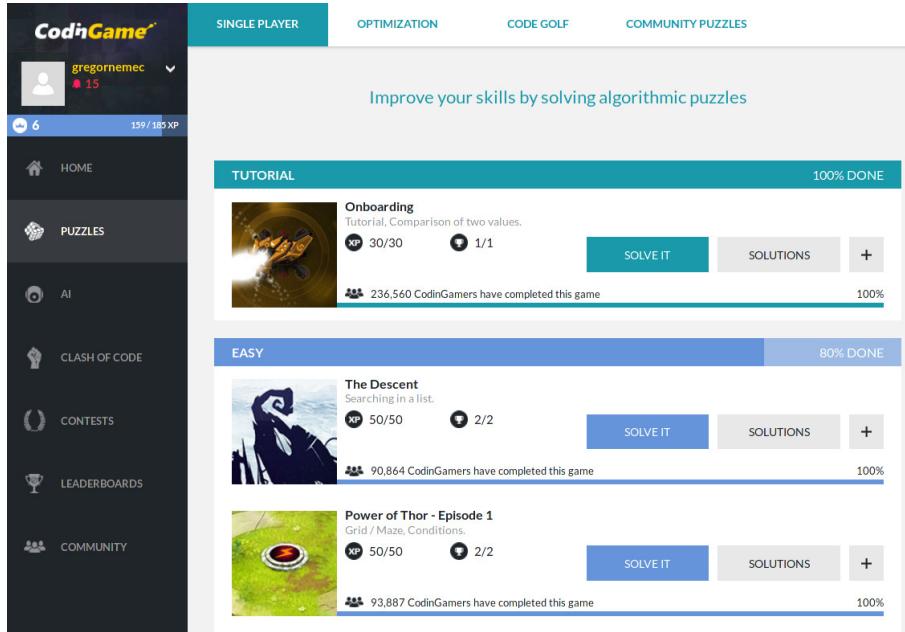
Vrsta vsebine	Spletna igra za programiranje. Kombinacija igre RPG + pisanje programske kode.
Jezik spletne strani	angleščina: da, slovenščina: ne, drugi: da.
Ponujena znanja	Znanja programskih jezikov in algoritmov.
Programski jeziki	Python, JavaScript, CoffeScript, LUA
Težavnostna stopnja	Osnovno solo (2/3 in 3/3) in srednjo šolo.
Upoštevanje načel	Problemski pristop: da, sistematičnost: da, postopnost: da.
Dosežki/Gamification	Da (Značke, izkušnje, diamanti)
Dodajanje lastnih vsebin	Ne.
Upravljanje razreda	Da (Za osnovni tečaj brezplačno za ostale tečaje plačljiva licenca)
Dostop vsebin	Pol plačljiv: brezplačnih je 145 nalog, plačljive so dodatne naloge 95, dodatni diamanti, ... za (9,99\$/mesec).

## 7.8 Codingame

Spletni portal *Codingame* [50] je spletna igra, katere bistvo je da uporabnik rešuje probleme, ki so zahtevni in so predstavljeni kot izzivi v obliki igre. Uporabnik z podajanjem rešitev izboljšuje veščine programiranja in znanje algoritmov.

Za vsakega uporabnika se igranje oz. reševanje problemov začne pri **sestavljkah** (*ang. puzzles*) (slika 41). V eno igralskem načinu so igre razdeljene na več težavnosti vse od **lahkih** do **zelo težkih**. Na tej strani lahko izbiramo med sestavljkami, kjer imamo nalogu, da programsko kodo čim bolj **optimiramo** (*ang. optimisation*), torej poiščemo rešitev, ki potrebuje najmanjšo časovno in prostorsko zahtevnost. Druga vrsta iger je tako da z čim manj kode, torej znanjem trikov programskega jezika rešimo problem, na strani *ang. Code golf*. Imamo še možnost, da rešujemo naloge, ki jih je pripravila skupnost, na strani *ang. Community puzzles*. Na tej strani se lahko posredujejo lastne naloge, ki pa morajo biti odobrene s strani razvijalcev spleten strani preden se lahko objavijo.

Poglejmo kako rešujemo problem. V ta name smo izbrali sestavljanko **Most - Episoda I** (*ang. The bridge - Episode I*) (slika 42). Za reševanje naloge lahko izbiramo med številnimi programskimi jeziki, kot so **C#, C++, Java, JavaScript, Python3, Bash, C, Clojure, Dart, F#**. V našem primeru bomo uporabili programski jezik **Python3**.



Slika 41: Podstran Codingame [50] - sestavljanke.

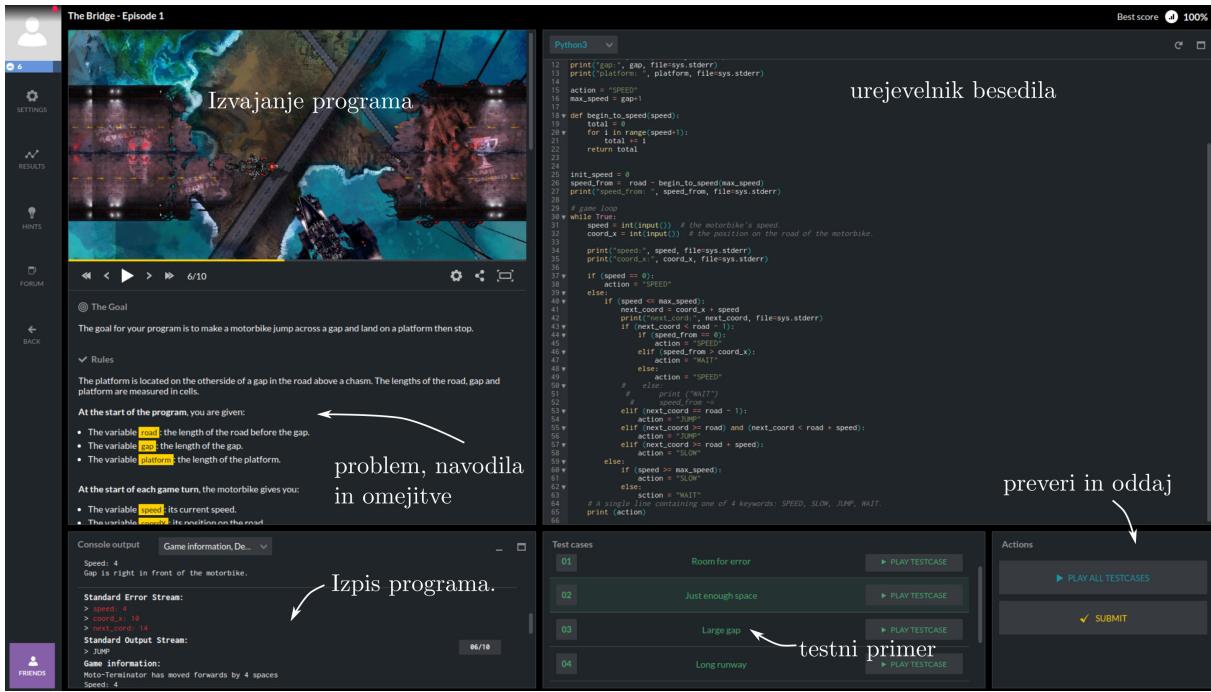
Primer izbrane naloge pravi, da moramo napisati program, ki upravlja hitrost motorja, kateri se vozi po mostu in mora preskočiti prepad ter se na koncu pravočasno ustaviti. Celoten program teče v neskončni while True zanki. Na vsakem koraku moramo izpisati kaj je naslednja akcija motorja ali ta naj POSPEŠI, ČAKA, ZAVIRA ali SKOČI. Na vsakem koraku se hitrost poveča oz zmanjša za 1. Hitrost se na vsakem koraku odraža za razdaljo, ki je enaka hitrosti. Vhodni podatki programa so naslednji, cesta, ki je razdalja pred prepadom, naslednji podatek je velikost prevara in razdalja zaviralne poti.

Na levi strani zgoraj **opazujemo potek rešitve**. Levo v sredini so napisana **navodila**, ki natanko opisajo problem, *vhodne podatke, omejitve* in pričakovani *izpis programa* na vsakem koraku. Levo spodaj je *ukazna vrstica*, na kateri spremljamo izpis programa in ki nam je v pomoč pri razhroščevanju.

Na desni strani je **urejevalnik besedil**, v katerega pišemo rešitev oz program. Urejevalnik zna barvati programsko kodo, samodejno upošteva zamik programske kode in podaja samodejne predloge za metode, ki so vgrajene ko napišemo “.”. V nastavitevah urejevalnika lahko izbiramo način, ki prilagodi vedenje in ga lahko nastavimo na *klasičen, emacs* in *vim* način. Izbiramo lahko med temno in svetlo barvno shemo. Nastavimo lahko samodejno zaključevanje oklepajev.

Spodaj desno poganjamo program s **testnimi primeri**, ki rešitev testirajo v skrajnih mejah. Testi, ki so podani kot vhodni podatki, so narejeni tako, da onemogočajo, da bi napisali program, ki bi imel statično vgrajene rešitve. Ko rešitev prestane vse teste jo lahko posredujemo.

Pri reševanju problemov se uči tudi življenjski krog razvijanja programske opreme in strategije



Slika 42: Reševanje sestavljanke *The Bridge* [50]

reševanja problemov, kot smo jih opisali v poglavju 4.2, saj **problem najprej analiziramo, izberemo pristop, problem razgradimo, razvijemo algoritem in testiramo njegovo pravilnost**. Ta postopek lahko ponavljamo v krajših korakih dokler ne pridemo do prave rešitve in ponovimo faze razvoja in testiranja algoritma več krat. Ko algoritem reši problem, se lahko ukvarjam z njegovo **učinkovitostjo**, ki jo pridobimo z upoštevanjem lastnosti implementacije algoritmov torej časovne in prostorske zahtevnosti ter z značilnosti programskega jezika in njegovih zmožnosti. Na tak posreden način nas spletni portal uči programski jezik, saj rešitev od nas zahteva optimalno rešitev, ki jo lahko dosežemo s dobrim poznavanjem programskega jezika in njegovih knjižnic.

Med reševanjem si lahko pomagamo z *namigi* in *pseudo kodo*. Ko je rešitev oddamo, sledi izračun točk in nagrad v obliki **točk izkušenj in značkami**. Uporabnik lahko brska med rešitvami drugih uporabnikov in tako nabira dodatno znanje.

Na spletni strani obstajajo še drugi načini igranja. Lahko se včlanimo v različne igre, kjer programiramo **bot-a**, ki tekmuje z **umetno inteligenco** na pod strani *ang. AI. Spopad kode* ali *ang. Clash of code* je način v katerem do 8 igralcev tekmujejo eden proti drugemu v bojih po 5 ali 10 minut. Vsak zase rešuje problem, na koncu zmaga najbolj točkovana rešitev. Spletni portal prireja tudi uradna **tekmovanja** na katerih se brezplačno registriramo. Izbiramo lahko med dvema sistemoma nagrajevanja, ali so nagrade v fizični obliki, kot je npr. 3D tiskalini, majice in podobno ali se odločimo za način, kjer se potegujemo za razgovor na določenem delovnem mestu, ki je takrat razpisano. Dober koncept prepoznavanja talentov in iskanja službe.

### 7.8.1 Povzetek

Spletni portal je namenjen predvsem k treniranju veščin programiranja in učenju algoritmov. Uporabnik zato mora poznati več kot so le osnove programiranja, zato je spletni portal namenjen predvsem za srednje šole, višje in univerzitetne programe študija. Spletna igra ni primerna za pouk, je pa lahko zelo dobro dopolnilo za dodatno vajo ter izpopolnjevanje lastnih izkušenj programiranja. Priporočamo ga lahko vsakemu, ki ga zanima programiranje. Sistem nalog in reševanja teh je dobro zasnovan. Problemško je zasnovano tudi ozadje vsake naloge. Naloge si po težavnosti sledijo postopoma, sistematično so naloge razmetane po vseh težavnostih stopnjah, zato ne moremo potrditi načela sistematičnosti.

**Codingame** | <https://www.codingame.com>

<b>Vrsta vsebine</b>	Vadnica: spletna aplikacija za programiranje + reševanje problemov + testiranje)
<b>Jezik spletne strani</b>	angleščina: da, slovenščina: ne, drugi: da.
<b>Ponujena znanja</b>	Znanje algoritmov in veščine programiranja.
<b>Programski jeziki</b>	C#, C++; Java, JavaScript, Python3, Bash, C, Clojure, Dart, F#
<b>Težavnostna stopnja</b>	Srednja šola, visoki, višji in univerzitetni.
<b>Upoštevanje načel</b>	Problemški pristop: da, sistematičnost: na, postopnost: da.
<b>Dosežki/Gamification</b>	Da, značke, izkušnje, rangiranje
<b>Dodajanje lastnih vsebin</b>	Da. Dodajanje lastnih nalog/problemov, ki jih rešuje skupnost.
<b>Upravljanje razreda</b>	Ne.
<b>Dostop vsebin</b>	Brezplačno

## 8 Možni načini uporabe spletnih portalov pri pouku

V spodnjem sestavku izpostavimo nekatere skupne prednosti, ki jih lahko prinese uporaba spletnih portalov za učenje programiranja pri pouku. S pregledom spletnih portalov, ki so nastali na univerzah (poglavlje 5) in tistimi ki smo jih podrobno pregledali (poglavlje 7) lahko ugotovimo naslednje prednosti.

**Nepotrebna namestitev programske opreme** je zagotovo velika prednost. Mentorju praktično ni potrebno nameščati nobenega urejevalnika besedil, IDE, niti prevajalnika ali tolmača. Prav tako ni potrebe po nastavljanju sistemskih poti, ki jih mnogi prevajalniki zahtevajo. Uporaba SPUP je neodvisna od uporabe operacijskega sistema. Vsaka naprava, na kateri lahko poganjamo novodobni spletni brskalnik omogoča uporabo SPUP. Učencem na domačih računalnikih ni potrebno instalirati nobene programske opreme. Na tem mestu bi poudarili, da učitelj mora biti previden pri dajanju domače naloge z uporabo računalnika. Zares se mora prepričati, da to zmožnost imajo vsi učenci. Najbolje je da vso dodatno delo, ki ga učitelj predvidi lahko učenci opravijo v šolski računalniški učilnici.

**Seznanjanje s programsko opremo** pri pouku ni potrebno. Znebimo se potrebe, da bi z novinci morali spoznavati urejevalnik besedil ali kompleksno IRO. Spoznavanje programskega jezika in reševanje problemov se lahko lotijo nemudoma.

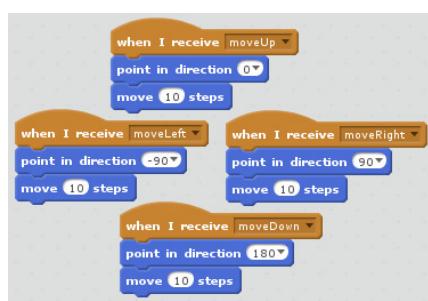
**Pisanje programa od začetka do konca** ni potrebno. Večina spletnih portalov, ki ponujajo vsebine v obliki vadnice, imajo programske naloge pripravljene tako, da uporabnik mora vnesti le del programske kode. Za novinca, to pomeni, da se osredotoči le na nalogu in del sintakse, ki jo v danem trenutku potrebuje, da reši zadano nologo. To prednost smo že spoznali na SPUP avstralske univerze, kjer so uporabili, tip naloge, zapolni prazna mesta [7].

Pripravili smo tudi konkretna primera, za OŠ in SŠ, kako bi s spletnim portalom lahko uresničili učne cilje, ki jih najdemo v učnem načrtu.

### 8.1 Primer uresničevanja ciljev učnega načrta v osnovni šoli

V učnem načrtu Računalništvo - neobvezni izbirni predmet [9] vsebuje vsebinski sklop **Algoritmi**, ki smo ga povzeli v poglavju 2.3.2. Spletni portal *Code combat* [47] lahko učitelj uporabi pri **ponovitvi sklopa**, ko je snov že predelal na primer s programskim jezikom in spletnim orodjem *Scratch* [41]. K obstoječim ciljem, ki so podani v učnem načrtu lahko dodamo še naslednje: *učenci spoznajo osnove programskega jezika Python*. Ker se učenci s pisanim programskim jezikom srečujejo prvič bi bilo dobro, da učitelj pripravi učne liste s primerjavo med

*Scratchom* in sintakso **Pythona** oz. metodami, ki se uporablja na portalu (slika 43b). V tem primeru niti ni pomembno podrobna obravnava programskega jezika Python. Pomembno je, da učenci razumejo kaj posamezna vrstica programske kode izvrši, kateri del kode predstavlja junaka in kateri akcijo junaka. To lahko pokažemo nazorno z delom programske kode v **Scrach-u** (slika 43a), ki ponazarja metode, ki jih uporabljamo na za vodenje junaka na spletni strani *Comdecombat* (slika 43b).



(a) Programska koda napisana v *Scratch-u* [41] za premikanje *gor*, *levo*, *desno*, *dol*.

```

1 #Premakni se navzgor
2 hero.moveTo(0)
3 #Premakni se desno
4 hero.moveRight()
5 #Premakni se levo
6 hero.moveLeft()
7 #Premakni se navzdol
8 hero.moveDown()
9

```

(b) Programska koda v **Pythonu** klica predpripripravljene metode, ki premikajo junaka *gor*, *levo*, *desno*, *dol*.  
[47]

Slika 43: Primerjava programskih blokov napisanih v **Scratchu** in enačic klica metod napisanih v **Pythonu** na spletni strani *Codecombat*

Kot smo že omenili v predstavitvi spletnne ga portala *Codecombat* ima učitelj dve možnosti:

- učenci uporabljajo svoje ustvarjene račune,
- učitelj ustvari **demo** razred in v njega vključi učence.

Glavna slabost v prvem primeru je ta, da tako lastno ustvarjeni računi učencev omogočajo izbiro junakov, predmetov, ki jih uporabljajo pri reševanju nalog in se zato lahko pri učni urzi zaplete pri nadzoru napredka pri posameznem učencu in je s tem delo učitelja oteženo. Iz teh razlogov se raje nagibamo k drugi možnosti.

Učitelj lahko s prošnjo zaprosi **demonstracijski** račun v katerem ima možnost, ustvariti razred s prvim vsebinskim sklopom **Uvoda v računalniško znanost** (*ang. Introduction to Computer Science*). To je edini sklop v tem načinu, ki je brezplačen in ga lahko uporabimo v šoli pri nas brez dodatnih stroškov. Učitelj učence povabi v razred z unikatno ustvarjenim **URL** naslovom. Registracija po kliku na povezavo je potrebna. Učenci se od naloge do naloge premikajo brez povzetka prejema dosežkov in izbire predmetov in izbire opreme za svojega junaka. Za tako delo je primeren tip učen ure **praktično vodeno delo**, kjer učitelj delo vodi z vnaprej pripravljenimi učnimi listi primerjave programskih blokov (slika 43) in povzetki navodil v slovenščini. Ko učenci v začetnih nalogah osvojijo osnovne metode junaka lahko učitelj stopnjuje samostojnost dela učencev. Delo na spletnem portalu je primerno za dvojno učno uro oz. **90min**. Tiste naloge, ki jih učenci ne uspejo dokončati, jih lahko učitelj da kot domače

delo.

Sklop vsebinsko predstavlja naslednje koncepte programiranja **osnove sintakse, argumenti, spremenljivke, tip spremenljivke:string, while zanka**. Pri preverjanju sklopa nekatere operativne cilje ne moremo preveriti, zapisali smo tiste, ki jih ni moč preveriti in zakaj.

- **algoritem predstavijo simbolno (z diagramom poteka) ali s pomočjo navodil v preprostem jeziku.** Na spletni strani algoritmi niso predstavljeni z diagrami, lahko jih sicer pripravi učitelj, vendar v tej fazi to ni nujno potrebno.
- **znajo v algoritem vključiti vejitev.** Uporaba vejitev v tem prvem sklopu **CS1** na spletnem portalu ni omogočena.
- *znajo uporabiti nekatere ključne algoritme za sortiranje in iskanje,*
- *poznajo osnovne algoritme za iskanje podatkov.*

Menimo, da ostale cilje lahko uspešno povzamemo in preverimo v učni uri ponovitve vsebinskega sklopa. Čeprav uporaba spletnne igre predstavlja kar nekaj priprave učitelja ima izjemno motivacijsko vrednost, ki učence lahko navduši za nadaljnje delo.

## 8.2 Primer uresničevanja ciljev učnega načrta v srednji šoli

Učni načrt **informatike** na programu Splošna gimnazija predvideva učenje algoritmov in programiranja pa ravni **posebnih znanj** v tematskem sklopu obdelave podatkov. Predpostavimo, da mentor oz. profesor predmeta Informatika, uresničuje znanja algoritmov in programiranja, ta lahko uporabi spletni portal *Codeacademy* [39] v namene domačega dela. Profesor mora upoštevati med predmetno povezovanj s predmetom angleščina in za dani sklop pripraviti prevode navodil, ki so lahko dijakom v pomoč. Prednost portala je še ta, da vsebinski sklopi in posamezne teme, ki so na voljo, lahko poljubno izbiramo ne glede na vrstni red oz. na to kaj smo že predelali. Kako lahko profesor izkoristi uporabo sledenju napredka dijakov smo že podrobno opisali v poglavju 7.2.1.

Spletni portal lahko uporabi pred obravnavo neke snovi in se tako dijaki že seznanijo z določeno funkcionalnostjo in sintakso programskega jezika. Ker dijaki že del znanja in sintakse usvojijo so naloge pri pouku lahko zahtevnejše in je lahko večji poudarek na strategiji reševanja (težjih) problemov. Konkretno, za domače delo lahko dijaki rešujejo nalogo “*Strings & Console Output*”, kjer osvojijo osnovno znanje ravnjanja z **pisanjem izhodnih podatkov** in **podatkovnega tipa string** oz. niza. Naučijo se uporabljati metode, ki jih ponuja podatkovni tip **string**, lepijo dve besedi skupaj in tako dalje. Kot že reženo, z osvojenim znanjem pozneje pri pouku lahko rešujejo zahtevnejše naloge.

V drugem primeru lahko ta isti portal uporabi po obravnavi določenega vsebinskega sklopa in

vadnico na portalu *Codeacademy* [39] izkoristi, dijaki z domačim delom utrdijo svoje znanje. Poglejmo konkreten primer. Dijaki na primer po **obravnavi sklopa, vhodni podatki, podatkovni tip, string in if else vejitev**, imajo zadostno znanje da rešijo nalogo “*PygLatin*”. Napisati morajo program, ki sprejme besedo in jo pretvorji po določenem pravilu slovarja ter jo izpiše na zaslon. Naloga je razdeljena na več vadnic, dijaki tako postopno rešujejo korak po koraku naloge. Napredek reševanja naloge lahko spremi profesor preko portala.

## 9 Zaključek

V diplomskem delu smo najprej spoznali, da učenje računalniške znanosti in programiranja spada v **primarno področje uporabe računalnika v izobraževanju**. Pri nas se je na splošnem izobraževalnem področju uveljavila usmeritev, ki zagovarja **splošno usposobljenost** za delo z računalnikom, kljub temu se z računalniško znanostjo in programiranjem večina učencev prvič srečaja pri izbirnih predmetih *Urejanje besedil, Multimedija in Računalniška omrežja* in pri neobveznem izbirnem predmetu *Računalništva v OŠ*. Dijaki se srečajo s programiranjem v 1. letniku pri predmetu *Informatika*. Posebnost so strokovni programi, katerim je osnova računalništvo in je poudarek na programiranju dosti večji, kar smo spoznali pri pregledu učnega načrta predmeta *Računalništva* Tehniške gimnazije. Ugotovimo lahko da je programiranje pri splošnih predmetih zastopano do te mere, da se učenci oz. dijaki dotaknejo teh znanj, pri čemer je v bolj strokovnih predmetih kot je pri *Računalništvo* v OŠ in na programu Tehnični gimnazije zastopano v vse podrobnosti.

Pri pregledu osnovnih pojmov smo spoznali kaj predstavljajo programske paradigme in ugotovili, da danes prevladuje paradigma **objektno orientiranega** programiranja. Podrobneje smo predstavili značilnosti programskih jezikov, ki so trenutno najbolj popularni in ti so **Java, C++, JavaScript in Python**. Čeprav so v slovenskem izobraževalnem prostoru na srednje šolskem nivoju prisotni vsi omenjeni, se za začetnike priporoča uporaba **Python-a**. V osnovni šoli bi se naj uporabljal programski jezik **Scratch**, ki uporablja grafičen način sestavljanja gradnikov. Pri spoznavanju osnovnih konceptov programiranja smo pripravili nekaj primerov programov, ki jih skušajo čim bolj nazorno predstaviti.

Pri proučevanju **aktivnega pristopa** smo spoznali, da pouk računalništva mora biti pozitivno naravna in je naj znanje grajeno z **aktivnim učenje**, torej učenci z lastnim delom odkrivajo in gradijo miselni model. Pri tem so naj uporabljene **konstruktivne metode** poučevanja. Za aktivno učenje je ravno tako značilen problemsko naravnani pouk. Reševanje nalog oz. raznovrstnih problemov je naravno prisoten pri poučevanju računalniške znanosti, zato smo izpostavili **strategije reševanja problemov** in smo osnove korake podrobno razdelili. Strategije reševanja problemov niso pomembne le za področje računalništva temveč za mnoga področja tehničnih in znanstvenih disciplin ter na sploh pri vsakdanjem življenju. Zanimalo nas ali so vse te značilnosti pouka računalništva zajete tudi z uporabo spletnih portalov.

Prvi spletni portali za učenje programiranja so nastali v akademskem okolju na različnih univerzah po svetu. Spoznali smo, da se študenti novinci srečujejo vsepovod s podobnimi težavami, kot je instalacija in nastavitev programske opreme, uporaba IRO, uporaba sintakse programskega jezika, razumevanje prevajalnika in tako naprej. Pri vseh treh spletnih portalih, ki smo jih pregledali, smo lahko strnili naslednje značilnosti, glavni del vsakega spletnega portala je **spletна aplikacija za učenje programiranja (SAZP)**. Njene značilnosti so, da vsebuje

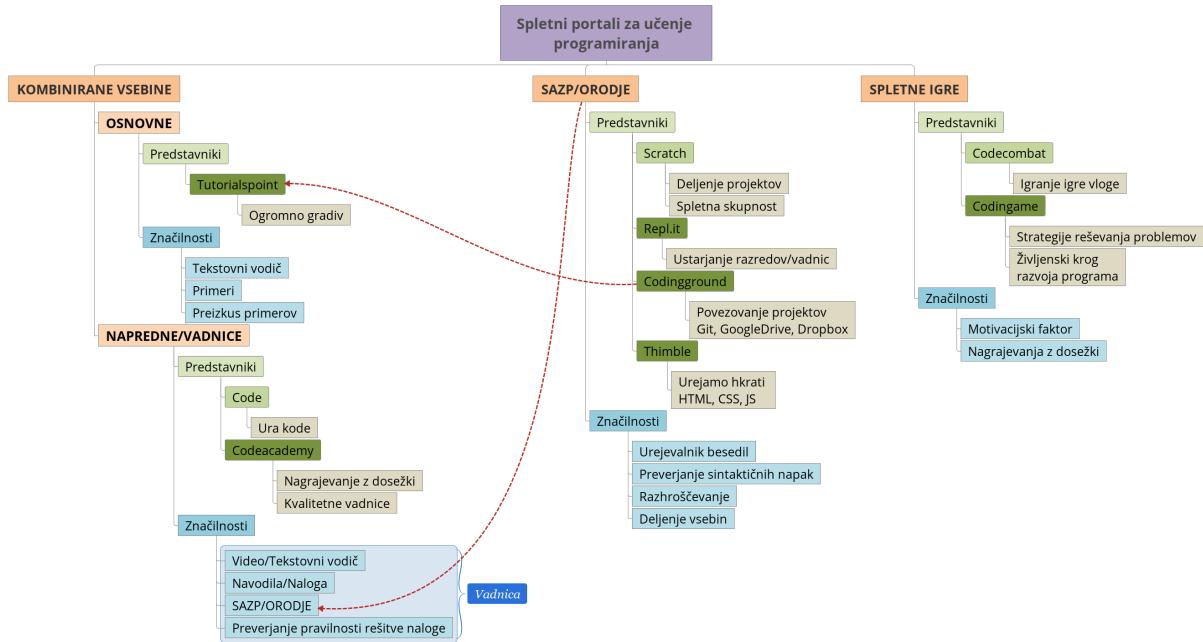
**urejevalnik besedil, omogoča zagon napisanega programa, vrača povratne informacije o napakah ter izpis vhodno-izhodnih podatkov.** Drugi elementi spletnega portala poleg SPZP so še razdelane vsebine z nalogami, omogočena je komunikacija med mentorjem in novincem, omogočen je pregled nad napredkom novinca.

Najzahtevnejši del tega diplomskega dela, je bilo iskanje kriterijev s katerimi smo lahko spletnne portale klasificirali in jih ovrednotili. Najprej smo jih razdelili po vrsti vsebine in ugotovili, da so najzanimivejše tiste vsebine, ki so **kombinirane**. Kombinirane vrste vsebin smo razdelili na **osnovne in napredne**, v slednjih se prepleta **SPZP**, tekstovni ali/in video **vodiči** ter **navodila**. Vsi ti elementi napredne kombinirane vsebine tvorijo **vadnico**, ki je navadno osnovna enota neke učne teme. Za posamezni spletni portal želimo določiti v katerem **jeziku** je predstavljen, katera **znanja ponuja**, ali so to veščine programiranja, znanja algoritmov, ali morda druga projektna znanja, kot je izdelava spletnih strani. Za vsak spletni portal smo zapisali, učenje katerih **programskih jezikov** omogoča, kakšni **težavnostni stopnji** je namenjen. Podrobnejše so nas zanimala ali spletni portali, tisti z vsebino znajo upoštevati nekatera **učna načela**, kot je **problemski pristop, načelo sistematičnosti in načelo postopnosti**. Pri posameznem spletnem portalu smo izpostavili ali ta uporablja **ocenjevanje, dosežkov**, ki je značilno za video igre. Zanimalo nas je še ali omogočajo spletni portali, da **dodajamo lastne vsebine** ter ali je omogočeno **upravljanje razreda**. Nazadnje nas je zanimalo ali so gradiva na spletnih portalih **brezplačna, pol plačljiva**, torej so nekatera gradiva ali storitve brezplačne druga plačljiva in **popolnoma plačljive**.

Spletih portalov, ki želijo poučevati računalniške znanosti in programiranja je na spletu zarez veliko, zato smo morali določiti omejitve s katerimi smo izbrali tiste, za katere smatramo, da nam bodo pri pouku najbolj v pomoč. Te omejitve so naslednje, spletni portal mora vsebovati **SAZP**, ki pa lahko nastopa tudi samostojno brez vsebine, vrsta vsebin naj bo **kombinirana** ter naj bodo vsebine na spletnih portalih dosegljive **brezplačno** ali **pol plačljivo**. Veliko spletnih portalov, ki na prvi pogled vabijo s kvalitetno pripravljeno vsebino in vadnico, zaradi popolne plačljivosti nismo mogli uvrstiti na seznam podrobne obravnave, saj zaradi tega razloga niso uporabne za pouk.

V diagramu (slika 44) smo zbrali povzetek pregleda spletnih portalov. Portale smo razvrstili po vrsti vsebine. Vsaki smo dodali predstavnike, ki smo jih pregledali, povzeli smo skupne značilnosti vsake vrste in za posamezni portal pripisali izstopajoče lastnosti.

Najprej povzemimo spletnne portale **Kombiniranih vsebin**, ti se ponašajo s kvalitetno pripravljenimi **vadnicami**. Posamezni vsebinski sklopi so razdeljeni na manjše enote, ki so med seboj sistematično povezane. Taki spletni portali nudijo tudi ideje in pripravljena gradiva za mentorje. Pregledali smo portal *Code*, ki je uporaben predvsem za uvodne učne ure računalništva v OŠ tudi v razredih prve triade. Spletni portal *Codeacademy* je primeren predvsem v SŠ kot dopolnilno gradivo učenja programskih jezikov kot je **Python** in osnovnih konceptov



Slika 44: Miselni vzorec z povzetkom pregledanih spletnih portalov s predstavniki in značilnostmi za posamezno vrsto vsebine.

programiranja. Glavna težava teh portalov je ta, da so gradiva napisana v **angleškem jeziku**. Mentor mora v ta namen imeti pripravljene prevode navodil nalog. Učne ure z uporabo teh spletnih portalov lahko uporablja z med predmetno povezavo s predmetom Angleščina.

V posebno kategorijo smo uvrstili spletne portale, ki ponujajo samostojno **spletno aplikacijo za programiranje**, ki jo uporabimo kot orodje. Ti portali navadno ne vsebujejo lastnih vsebin. Kot prvo smo pregledali spletni aplikacijo **Scrach**, ki je v OŠ že dobro poznana. Scratch ima številne zmožnosti in bi ga lahko uporabili takoj po uvodu, ki ga naredimo s spletnim portalom **Code**. Naslednja SAZP, **Repl.it** omogoča ustvarjanje razredov in lastnih vadnic. V vadnicah lahko vključimo teoretični uvod, navodila, ter ogrodje programa. **Codinground**, ki je del spletnega portala *Tutorials point* omogoča dober urejevalnik besedil z ukazno vrstico ter dobro povezljivost z oblačnimi shrambami kot je **Dropbox**, **GoogleDrive**, **OneDrive** in kontrolo verzije **Git**. **Thimble** omogoča ustvarjanje projektov, z večimi ločenimi datotekami, projekt delimo naprej, tako da jih uporabniki ali učenci lahko spreminja. Izkaže se za odlično orodje učenja spletnih tehnologij. Bistvena prednost uporabe SAZP je ta, da ima mentor, svobodno izbiro, katero vsebino bo podajal, vendar mora v priprava vsebine vložiti več truda in časa. Vsebino prilagaja in jo priteva po potrebi učnega načrta ter lastni presoji zmožnosti učencev.

Posebna kategorija spletnih portalov za učenje programiranja so **spletne igre**. Izpostavili smo dve, prva **Codecomba**, je namenjena višjim razredom OŠ ter SŠ. Čeprav smo govorili o tem, da pisani programske jeziki niso primerni za uporabo v OŠ ta spletni portal predstavlja izjemo, kar smo povzeli s primerom možnih načinov uporabe spletnih portalov. Spletna igra

temelji na **igranju igre vloge**, igralec prevzame vlogo junaka, ki ga vodi in se z njim bojuje s pisanjem programske kode. Junak ima omogočene veščine torej metode programskega jezika, ki jih uporablja in so vgrajene v različne predmete ali opremo. Drug primer spletnne igre je *Codingame*, ki sodi v višjo zahtevnost in je namenjen predvsem tistim, ki želijo svoje znanje in veščino programiranja, poznavanje programskega jezika ter algoritmov izpopolniti z dokaj zahtevnimi problemsko zastavljenimi nalogami. Z reševanjem nalog se uporabnik ne le v strategijah reševanja problemov in pisanja algoritmov temveč tudi optimizaciji napisanega algoritma. Spletna aplikacija prav tako uči življenjski krog pisanja programa. Skupna značilnost spletnim igram je ta, da imajo močan motivacijski faktor, ki je včasih zelo potreben, da novinci ostanejo pri pisanju programske kode in reševanju problemov.

V splošnem lahko trdimo, da je uporaba spletnih portalov za učenje programiranja problemsko naravnana in v sami osnovi vzpodbuja aktivnost učencev. Uporaba prinaša prednosti za mentorja in novince ko je ta, da ni potrebe po instalaciji programske opreme, kar omogoča kompatibilnost na številnih platformah, saj lahko spletno aplikacijo naložimo v vsakem novodobnem brskalniku. Novince ni potrebno priučiti uporabe zahtevnih uporabniških vmesnikov IRO. Navadno je vsebina spletnih portalov nastavljena tako, da jim ni potrebno pisati celotne programske kode, saj je ogrodje že podano in morajo napisati samo zahtevno rešitev. Tako se lahko bolje osredotočimo na reševanje problemov. Kot smo v diplomskem delu spoznali so spletni portali za učenje programiranja uporabni, vsak od njih ima svojo prednost. Na mentorju je naloga, da čim bolj skuša izkoristiti te prednosti in s tem poskušti navdušiti čim več novincev, da bodo postali dobri programerji.

## Literatura in viri

- [1] Gerlič, Ivan, *Sodobna informacijska tehnologija v izobraževanju*, DZS, Ljubljana, 2000.
- [2] Anthony Robins, Janet Rountree, and Nathan Rountree, "Learning and Teaching Programming: A Review and Discussion" v *Computer Science education*, vol 13, No. 2, 2003, pp. 137 - 172.
- [3] S.C. Ng, S.O Choy, R. Kwan, S.F. Chan, "A Web-Based Environment to Improve Teaching and Learning of Computer Programming in Distance Education", *ICWL'05 Proceedings of the 4th international conference on Advances in Web-Based Learning*, 2005
- [4] L. Ma, J. D. Ferguson, M. Roper, I. Ross, M. Wood, "A web-based learning model for improving programming students' mental models", v *Proceedings of the 9th annual conference of the subject centre for information and computer sciences*, HE Academy, 2008 pp. 88-94.
- [5] O. Hazzan, T. Lapidot, N. Ragonis, *Guide to Teaching Computer Science*, Springer, 2011.
- [6] Wikipedia contributors, *Problem solving*, Wikipedia, The Free Encyclopedia. Pridobljeno 25.4.2016 iz, [https://en.wikipedia.org/wiki/Problem\\_solving#Problem-solving\\_strategies](https://en.wikipedia.org/wiki/Problem_solving#Problem-solving_strategies).
- [7] Nghi Truong, *A web-based programming environment for novice programmers*, Queensland University of Technology, Australia, 2007.
- [8] Vladimir Batagelj et al., *UČNI načrt, Izbirni predmet: Program osnovnošolskega izobraževanja, Računalništvo*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2002. Pridobljeno 2.4.2016 iz, [http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/predmeti\\_izbirni/Racunalnistvo\\_izbirni.pdf](http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/predmeti_izbirni/Racunalnistvo_izbirni.pdf)
- [9] Radovan Kranjc et al., *UČNI načrt, Program osnovnošolskega izobraževanja, Računalništvo: neobvezni izbirni predmet*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2002. Pridobljeno 2.4.2016 iz, [http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/program\\_raszirjeni/Racunalnistvo\\_izbirni\\_neobvezni.pdf](http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/program_raszirjeni/Racunalnistvo_izbirni_neobvezni.pdf).
- [10] Wechtersbach Rado, *UČNI načrt, Informatika [Elektronski vir]: gimnazija : splošna, klasična, strokovna gimnazija : obvezni predmet (70 ur), izbirni predmet (210 ur), matura (70 + 210 ur)*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2008. Pridobljeno 2.4.2016 iz, [http://www.mss.gov.si/fileadmin/mss.gov.si/pageuploads/podrocje/ss/programi/2008/Gimnazije/UN\\_\\_INFORMATIKA\\_gimn.pdf](http://www.mss.gov.si/fileadmin/mss.gov.si/pageuploads/podrocje/ss/programi/2008/Gimnazije/UN__INFORMATIKA_gimn.pdf).

- [11] Predmetna komisija Tea Lončarič et al., *Računalništvo [Elektronski vir] : gimnazija, tehniška gimnazija : izbirni strokovni maturitetni predmet (280 ur)*, Ministrstvo za šolstvo, znanost in šport: Zavod RS za šolstvo, Ljubljana, 2010. Pridobljeno 2.4.2016 iz, [http://eportal.mss.edus.si/msswww/programi2010/programi/media/pdf/un\\_gimnazija/tehniska-gimnazija/UN\\_Racunalnistvo.pdf](http://eportal.mss.edus.si/msswww/programi2010/programi/media/pdf/un_gimnazija/tehniska-gimnazija/UN_Racunalnistvo.pdf).
- [12] Wikipedia contributors, *Computer program*, Wikipedia, The Free Encyclopedia. Pridobljeno 25.4.2016 iz, [https://en.wikipedia.org/wiki/Computer\\_programming](https://en.wikipedia.org/wiki/Computer_programming).
- [13] Wikipedia contributors, *Algorithem*, Wikipedia, The Free Encyclopedia. Pridobljeno 25.4.2016 iz, <https://en.wikipedia.org/wiki/Algorithm>.
- [14] Jonah Bitautas, *The Differences Between Programmers and Coders*, Workfunc. Pridobljeno 26.4.2016 iz, <http://workfunc.com/differences-between-programmers-and-coders/>.
- [15] Carl Reynolds, Paul Tymann, *Principles of Computer science*, McGraw-Hill, London, 2008.
- [16] Stoyan Stefanov, Kumar Chetan Sharman, *Object-Oriented Java Script, Second edition*, Packt Publishing, Ltd, Birmingham, 2013.
- [17] Wikipedia contributors, *Programming paradigm*, Wikipedia, The Free Encyclopedia. Pridobljeno 26.4.2016 iz, [https://en.wikipedia.org/wiki/Programming\\_paradigm](https://en.wikipedia.org/wiki/Programming_paradigm).
- [18] Wikipedia contributors, *Object-oriented programming*, Wikipedia, The Free Encyclopedia. Pridobljeno 26.4.2016 iz, [https://en.wikipedia.org/wiki/Object-oriented\\_programming](https://en.wikipedia.org/wiki/Object-oriented_programming).
- [19] Wikipedia contributors, *Java(programming language)*, Wikipedia, The Free Encyclopedia. Pridobljeno 27.4.2016 iz, [https://simple.wikipedia.org/wiki/Java\\_%28programming\\_language%29](https://simple.wikipedia.org/wiki/Java_%28programming_language%29).
- [20] Wikipedia contributors, *C++*, Wikipedia, The Free Encyclopedia. Pridobljeno 27.4.2016 iz, <https://en.wikipedia.org/wiki/C%2B%2B>.
- [21] Wikipedia contributors, *Python(programming language)*, Wikipedia, The Free Encyclopedia. Pridobljeno 30.4.2016 iz, [https://en.wikipedia.org/wiki/Python\\_%28programming\\_language%29](https://en.wikipedia.org/wiki/Python_%28programming_language%29).
- [22] Zed A. Shaw, *Learn Python the Hard Way*. Pridobljeno 2.5.2016 iz, <http://learnpythonthehardway.org/book/>.
- [23] Wikipedia contributors, *Tutorial*, Wikipedia, The Free Encyclopedia. Pridobljeno 5.6.2016 iz, <https://en.wikipedia.org/wiki/Tutorial>.

- [24] The Python Software Foundation, *The Python tutorial*. Pridobljeno 6.6.2016 iz, <https://docs.python.org/3/tutorial/index.html>.
- [25] Richard E. Mayer, *Principles for multimedia learning*. Pridobljeno 6.6.2016 iz, <http://hilt.harvard.edu/blog/principles-multimedia-learning-richard-e-mayer>.
- [26] *Udemy*. Pridobljeno 6.6.2016 iz, <https://www.udemy.com>.
- [27] Python fiddle, *Python Cloud IDE*. Pridobljeno 6.6.2016 iz, <http://pythonfiddle.com/>.
- [28] *Cloud9*. Pridobljeno 6.6.2016 iz, <https://c9.io/>.
- [29] *Codenvy*. Pridobljeno 6.6.2016 iz, <https://codenvy.com/>.
- [30] *w3school*. Pridobljeno 6.6.2016 iz, <http://www.w3schools.com/default.asp>.
- [31] *Fightcode*. Pridobljeno 6.6.2016 iz, <http://fightcodegame.com/>.
- [32] *Codeschool*. Pridobljeno 6.6.2016 iz, <https://www.codeschool.com/>.
- [33] Wikipedia contributors, *Education in the United States*, Wikipedia, The Free Encyclopedia. Pridobljeno 5.6.2016 iz, [https://en.wikipedia.org/wiki/Education\\_in\\_the\\_United\\_States#K.E2.80.9312\\_education](https://en.wikipedia.org/wiki/Education_in_the_United_States#K.E2.80.9312_education).
- [34] *Moodle*. Pridobljeno 6.6.2016 iz, <https://moodle.org/>.
- [35] *Code.org Promote*, Pridobljeno 22.6.2016 iz, <https://code.org/promote>
- [36] *Code.org About us*, Pridobljeno 22.6.2016 iz, <https://code.org/about>
- [37] *Code.org*, Pridobljeno 22.6.2016 iz, <https://code.org>
- [38] *Code studio*, Pridobljeno 27.6.2016 iz, <https://studio.code.org>
- [39] *Codeacademy*, Pridobljeno 6.9.2016 iz, <https://www.codecademy.com>
- [40] Elliott Bristow, *Gaming in education: Gamification*. Pridobljeno 6.6.2016 iz, <https://www.theedublogger.com/2015/01/20/gaming-in-education-gamification/>.
- [41] *Scratch*, Pridobljeno 15.9.2016 iz, <https://scratch.mit.edu/>
- [42] *Scratch About*, Pridobljeno 15.9.2016 iz, <https://scratch.mit.edu/about/>
- [43] *Repl.it*, Pridobljeno 18.9.2016 iz, <https://repl.it/>
- [44] *Freecodecamp*, Pridobljeno 18.9.2016 iz, <https://www.freecodecamp.com>
- [45] *Tutorialspoint*, Pridobljeno 18.9.2016 iz, <http://www.tutorialspoint.com/>

- [46] *Tutorialspoint, Codingground*, Pridobljeno 18.9.2016 iz, <http://www.tutorialspoint.com/codingground.htm>
- [47] *Code combat*, Pridobljeno 20.6.2016 iz, <https://codecombat.com>
- [48] *Code combat, About*, Pridobljeno 20.6.2016 iz, <https://codecombat.com/about>
- [49] *Thimble by mozilla*, Pridobljeno 30.6.2016 iz, <https://thimble.mozilla.org>
- [50] *Codingame*, Pridobljeno 28.6.2016 iz, <https://www.codingame.com/home>
- [51] Codingground, *Execute python online*. Pridobljeno 6.6.2016 iz, [http://www.tutorialspoint.com/execute\\_python\\_online.php](http://www.tutorialspoint.com/execute_python_online.php).