І.В. Юрченко, В.С. Сікора

ПРОГРАМУВАННЯ
мовою Python

Міністерство освіти і науки України

Чернівецький національний університет
імені Юрія Федьковича

Юрченко І.В., Сікора В.С.

ПРОГРАМУВАННЯ МОВОЮ PYTHON

навчальний посібник

Чернівці
Чернівецький національний університет
2022

ББК 32.973
П–741
УДК 004.432

Рецензенти:

доктор фіз.-мат. наук, професор
Ясинський Володимир Кирилович

кандидат фіз.-мат. наук, доцент
Семчук Аркадій Романович

Юрченко І.В., Сікора В.С. Програмування мовою Python: навчальний посібник.– Чернівці: Чернівецький національний університет, 2022.– 104 с.

У навчальному посібнику наведено теоретичний матеріал, приклади розв'язування задач та завдання для самостійної роботи, які допоможуть студентам засвоїти основні положення та методи програмування мовою Python (типи даних, лінійні та розгалужені, циклічні програми, списки, кортежі, рядки, словники, множини, підпрограми, декоратори, генератори, лямбда-функції, об'єктно-орієнтоване програмування, обробка виключних ситуацій, файли, модулі) і вміти застосовувати набуті теоретичні знання для побудови програм з використанням мови Python.

Для студентів спеціальностей 122 – "Комп'ютерні науки", 124 – "Системний аналіз".

ББК 32.973
УДК 004.432

ПОНЯТТЯ АЛГОРИТМУ ТА АЛГОРИТМІЧНОЇ МОВИ

Алгоритмом називається чітке описання послідовності дій, які необхідно виконати для розв'язання задач.

Практичне розв'язування будь-якої задачі вимагає отримання результату із заданих початкових даних, тому справедливе ще й таке означення алгоритму.

Алгоритм – це описання послідовного процесу перетворення початкових даних на результат .

Розробити алгоритм розв'язування задачі означає розбити задачу на послідовно виконувані кроки або етапи (підзадачі). При цьому результати виконання попередніх етапів можуть використовуватись при виконанні наступних. Разом з цим потрібно чітко вказати не тільки зміст кожного етапу, але й порядок виконання етапу. Окремий крок алгоритму або складається з іншої, простішої задачі, алгоритм якої вже розроблений, або повинен бути досить простим і зрозумілим без пояснень. Чітко сформульована послідовність правил, яка описує цей процес, і є алгоритмом. Якщо алгоритм розроблений, то його можна передати для виконання. Виконавець, точно виконуючи правила (інструкції) алгоритма, отримає розв'язок задачі.

Приклад.

Початкові дані: продукти, які необхідні для приготування українського борщу з пампушками.

Задача: приготувати.

Алгоритм: рецепт приготування українського борщу з кулінарної книги.

Основні властивості алгоритму:

1) *Дискретність*.

Для виконання кожного етапу алгоритму потрібен деякий скінченний час, тобто перетворення початкових даних у результат здійснюється у часі дискретно.

2) *Конкретність або детермінованість*.

Кожне правило алгоритму має бути чітким, однозначним і не залишати місця для вільного трактування. Завдяки цій властивості виконання алгоритму відбувається механічно і не потребує жодних додаткових вказівок чи повідомлень про задачу, яка розв'язується.

3) *Результативність або скінченність*.

Алгоритм повинен приводити до розв'язку задачі за скінченну кількість кроків.

4) *Масовість або універсальність*.

Алгоритм розв'язку задачі розробляється в загальному вигляді, тобто він повинен застосовуватися не лише для розв'язання якоїсь конкретної задачі, а для деякого класу задач, які відрізняються лише початковими даними. При цьому початкові дані вибираються з деякої області, яка називається областю застосування алгоритму.

Етап розв'язування задачі, результатом якого є розробка алгоритму її розв'язання, називається *алгоритмізацією* (під цим розуміють зведення розв'язування задачі до ланцюжка простих кроків, які виконуються послідовно один за одним).

У широкому розумінні алгоритмізація складається з вибору методу розв'язування задачі та форми представлення вхідної інформації, які враховують специфіку комп'ютера.

Розроблений алгоритм можна зафіксувати трьома способами:

1) людською мовою;
2) у вигляді схем;
3) у вигляді програми, тобто за допомогою спеціфічної мови для запису алгоритмів, яка є зрозумілою для комп'ютера (на алгоритмічній мові).

Один і той самий алгоритм можна записати різними алгоритмічними мовами.

Алгоритмічна мова – це сукупність певних символів та правил, які вказують на те, яким чином з їхньою допомогою описувати алгоритм. Комп'ютер працює в двійковій системі числення (в машинних кодах). Таке представлення не є зручним для людини. Для полегшення спілкування людини з комп'ютером і розробляються алгоритмічні мови, які є ближчими до мови людини, ніж коди комп'ютера. У світі розроблено досить багато алгоритмічних мов високого рівня. Серед цих мов є проблемноорієнтовані та універсальні.

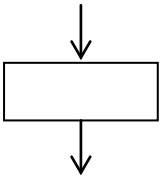
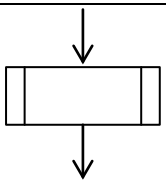
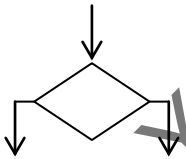
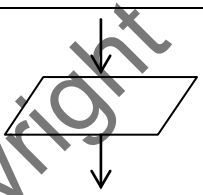
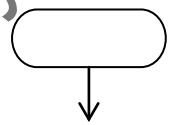
Способи складання алгоритму, структури алгоритму

Є два способи складання алгоритму: знизу вгору, згори вниз (метод покрокової деталізації).

Програмування *знизу вгору* спирається на дуже чітке і глибоке представлення всього ходу розв'язування задач. Для цього необхідно розв'язати задачу самостійно на папері. Для деяких наборів даних, що не вимагають громіздких обчислень, запам'ятовуються при цьому виконувані дії таким чином, щоб надалі їх формалізувати, тобто записати у вигляді послідовності чітких правил.

Програмуючи *згори вниз*, спочатку аналізується і фіксується загальна структура алгоритму без детальної розробки окремих її частин. Блоки, які вимагають подальшої деталізації, позначаються пунктирною

лінією, а які не вимагають – суцільною лінією. Потім деталізуються ті блоки, які не були деталізовані на попередньому кроці. Таким чином, на кожному подальшому кроці уточнюється реалізація частини алгоритму, що дозвоює мати справу з простішою задачею. Після закінчення деталізації всіх блоків ми отримуємо розв'язування всієї задачі. Цей метод називається методом покрокової деталізації. Зауважимо, що для розв'язування однієї й тієї самої задачі можна скласти різні алгоритми. Розроблений алгоритм можна зафіксувати за допомогою схеми. При цьому використовують певні геометричні фігури. Таким чином, *блок-схемою* називається наочне графічне зображення алгоритму за допомогою різноманітних геометричних фігур.

	Прямокутник – обчислення. Один вхід, один вихід.
	Окрема програма або підпрограма, яка вже створена. Записується назва підпрограми та її параметри. Один вхід, один вихід.
	Ромб – перевірка умови (розгалуження). Один вхід, два виходи.
	Паралелограм – введення або виведення даних. Один вхід, один вихід.
	Початок, кінець програми або вихід підпрограми. Або один вхід, або один вихід.

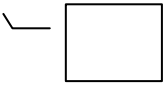
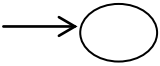
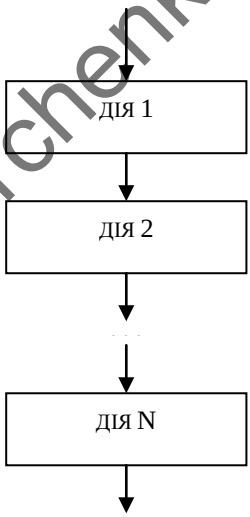
	Виведення на дисплей.
	Друк на папері.
	Коментар.
	Мітка.

Схема дозволяє наочно представити структуру алгоритму. На ній добре видно лінійні, розгалужені та циклічні структури алгоритму.

Лінійна структура

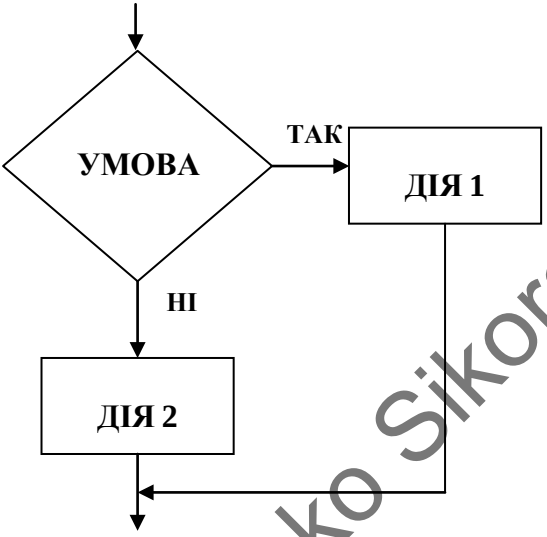
При виконанні лінійної структури алгоритму всі дії виконуються послідовно, одна за одною. Блок-схема лінійної структури алгоритму має такий вигляд.



Розгалужена структура

При виконанні розгалуженої структури дії відбуваються або не відбуваються в залежності від виконання певної умови. Зокрема, розглянемо наведену на блок-схемі розгалужену структуру. Якщо умова

справджується, тоді виконується дія 1, а дія 2 ігнорується. Якщо ж умова не справджується, тоді виконується дія 2, дія 1 – ігнорується.



Циклічні структури

Цикл – це виконання послідовності кроків, після останнього з яких переходять до нового виконання цієї послідовності, починаючи з першого. На схемі вони утворюють замкнуті ділянки.

Циклічні структури поділяються на цикли з післяумовою та з передумовою.

Цикли з післяумовою складаються з:

- підготовчої частини;
- робочої частини;
- перерахунку параметрів циклу;
- перевірки умови продовження циклу;
- продовження програми.

Параметром циклу називається така змінна, за значенням якої робиться висновок – продовжувати виконання циклу чи припинити.

У підготовчій частині циклу проводиться така підготовча робота, яка забезпечить вірне виконання циклу для розв'язування задачі. Наприклад, якщо в циклі обчислюється добуток чисел, то в комірку пам'яті, де зберігається цей добуток, потрібно записати одиницю, а якщо

в циклі обчислюється сума, то в комірку пам'яті потрібно записати нуль.

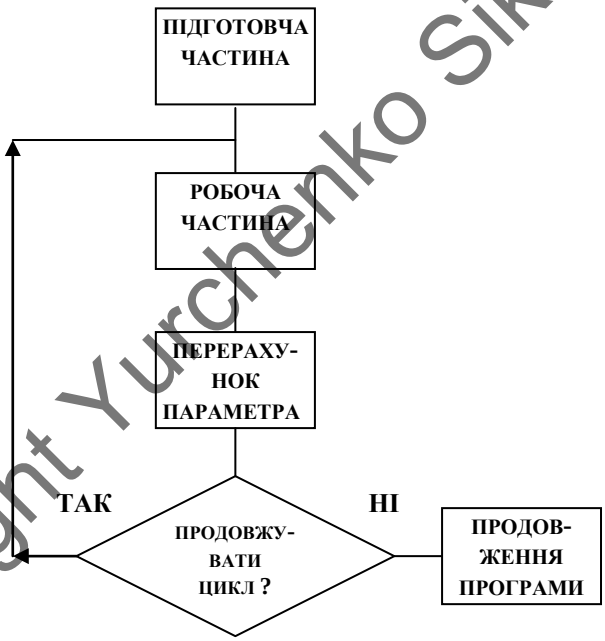
У підготовчій частині циклу обов'язково задається початкове значення параметру циклу.

Робоча частина циклу – це сукупність тієї послідовності кроків, яку необхідно виконати декілька разів для розв'язання задачі. Саме задля цієї послідовності кроків і організований цикл.

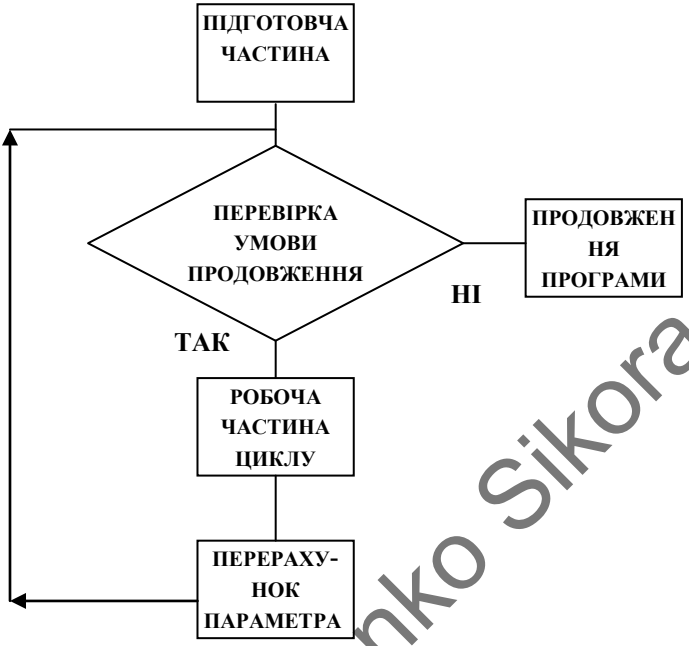
Перерахунок параметру циклу здійснюється таким чином, щоб нове виконання робочої частини проходило так, як цього вимагає розв'язування задачі.

Умова продовження циклу повинна бути записана так, щоб при кожному виконанні цього циклу було однозначно зрозуміло – чи потрібно продовжувати цикл.

Нижче наведено блок-схему циклу з післяумовою.



У *циклі з післяумовою* робоча частина виконується хоча б один раз, а в циклі з передумовою робоча частина може не виконуватися жодного разу (якщо передумова відразу не виконується). Якщо вищевказаний момент для розв'язання задачі не є принциповим, то можна використовувати будь-який із цих циклів.



КОРОТКИЙ ДОВІДНИК
З МОВИ ПРОГРАМУВАННЯ PYTHON

Ідентифікатори в Python

Ідентифікатори в Python – це імена, які використовуються для ідентифікації змінних, функцій, класів, модулів та інших об'єктів. Ідентифікатор починається з літер A-Z, або a-z, або підкреслення і складається з літер, підкреслень або цифр від 0 до 9.

Ідентифікатори Python не використовують символи @, $, %. Python чутливий до регістру, тобто Abc і abc є двома різними іменами (ідентифікаторами).

Ось основні правила іменування ідентифікаторів у Python:

- імена класів починаються з великої літери, а всі інші ідентифікатори починаються з нижнього регістру;
- якщо ідентифікатор починається з підкреслення (_), то він є приватним;
- якщо ідентифікатор починається двома підкресленнями і закінчується двома кінцевими підкресленнями, він є спеціальним іменем, визначеним мовою.

Зарезервовані імена

У наведеному нижче списку перелічено імена, зарезервовані в Python. Їх використання не допускається у використанні визначень констант, змінних або будь-яких інших користувацьких імен. Усі зарезервовані слова містять лише малі літери:

and	exec	not
assert	finally	or
break	for	pass

class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Лінії та відступи

Одне з найважливіших правил для тих, хто почав вивчати Python, полягає в тому, що ця мова не використовує звичайні фігурні дужки при маркуванні меж кодових блоків для класів і функцій, а також для управління потоками. Замість цього Python використовує відступ рядків. Кількість відступів на початку рядка не має значення, але всі оператори в межах такого блоку повинні мати однакову їхню кількість.

Призначення значень змінним

Змінні в Python не потрібно оголошували, щоб виділити їм пам'ять. Оголошення (присвоєння, призначення і т.д.) змінної і виділення пам'яті для зберігання її значення здійснюється в той момент, коли ви вказуєте деякі дані (призначаєте значення) цій змінній. Для цього використовуйте символ оператора призначення (=).

Операнд у лівій частині знака рівності вказує ім'я змінної, а операнд з правого боку вказує значення, яке буде зберігатися в цій змінній.

Завдання 10.

Увести з клавіатури координати (*x*,*y*) трьох точок. Визначити, чи можна утворити з цих точок трикутник. Обчислити площу та периметр трикутника, якщо точки належать до фігури, утвореної лініями: x**2+y**2<=9, y<=x/2, y<=0. При розв'язанні задачі використати підпрограму.

```
import math

def f(x,y):
   return ((x**2 + y**2 <= 9) and (y <= x/2) and (y <= 0))

print("Точка A:")
x_1 = float(input("Введіть координату x_1: "))
y_1 = float(input("Введіть координату y_1: "))

print("\nТочка B:")
x_2 = float(input("Введіть координату x_2: "))
y_2 = float(input("Введіть координату y_2: "))

print("\nТочка C:")
x_3 = float(input("Введіть координату x_3: "))
y_3 = float(input("Введіть координату y_3: "))

t = (x_2 - x_1)*(y_3 - y_1) - (x_3 - x_1)*(y_2 - y_1)

l_1 = t == 0

l_2 = f(x_1,y_1) and f(x_2,y_2) and f(x_3,y_3)

if not(l_1) and l_2:
  S = 0.5*abs(x_1*y_2 + x_2*y_3 + y_1*x_3 - x_3*y_2 -
x_1*y_3 - x_2*y_1)
```

```
    ab = math.sqrt((x_2 - x_1)**2 + (y_2 - y_1)**2)
    bc = math.sqrt((x_3 - x_2)**2 + (y_3 - y_2)**2)
    ac = math.sqrt((x_3 - x_1)**2 + (y_3 - y_1)**2)
    p = ab+bc+ac
    print("Площа трикутника ABC = %.5f." % S)
    print("Периметр трикутника ABC = %.5f." % p)
elif l_1:
    print("Три точки лежать на одній прямій.")
elif not(l_2):
    print("точки не належать до фігури.")
```

Чернівецький національний університет
імені Юрія Федьковича

Факультет математики та інформатики

Кафедра математичного моделювання

СИЛАБУС
навчальної дисципліни

Програмування мовою Python

вибіркова

Освітньо-професійна програма: “Інформаційні технології та управління проектами”, “Системний аналіз”

Спеціальність 6.122 – Комп’ютерні науки, 6.124 – Системний аналіз

Галузь знань 12 – Інформаційні технології

Рівень вищої освіти перший (бакалаврський)

Розробник: Юрченко Ігор Валерійович, доцент кафедри математичного моделювання, кандидат фіз.-мат. наук, доцент

Профайл викладача
http://matmod.fmi.org.ua/pro-kafedru/spivrobitnyky/yurchenko-igor-valeriyovich/

E-mail: i.yurchenko@chnu.edu.ua

1. **Анотація дисципліни** (призначення навчальної дисципліни).

Навчальна дисципліна призначена для вивчення основ програмування мовою Python.

2. **Мета навчальної дисципліни**: ознайомити студентів із основами програмування мовою Python, навчити їх застосовувати цю мову при розв'язуванні прикладних задач.

3. **Пререквізити**. Навчальна дисципліна: "Програмування".

4. **Результати навчання**. У результаті вивчення навчальної дисципліни студенти повинні

знати: основи мови Python (типи даних, лінійні та розгалужені, циклічні програми, списки, кортежі, рядки, словники, множини, підпрограми, декоратори, генератори, лямбда-функції, об'єктно-орієнтоване програмування, обробка виключних ситуацій, файли, робота з операційною системою, модулі);

вміти: застосовувати набуті теоретичні знання для побудови навчальних програм та програмних продуктів з використанням мови Python.

Компетенції освітньої програми:

ФК3. Здатність до логічного мислення, побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проектування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв'язності та нерозв'язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем.

ФК8. Здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно-орієнтованого, функціо-

нального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління.

5. Опис навчальної дисципліни
5.1. Загальна інформація

Назва навчальної дисципліни: “Програмування мовою Python”											
Форма навчання	Рік підготовки	Семестр	Кіль–кість		Кількість годин						Вид підсумкового контролю
			кредитів	годин	лекції	практичні	семінарські	лабораторні	самостійна робота	індивідуальні завдання	
Денна	2	3	3	90	15	–		30	45	–	залік

5.2. Дидактична карта навчальної дисципліни

Назви змістових модулів і тем	Кількість годин					
	денна форма					
	усього	у тому числі				
		л	п	лаб	інд	с.р.
1	2	3	4	5	6	7
Теми лекційних занять	Змістовий модуль 1					
НЕ 1.1. (Лекція) Початки роботи з Python.	4	2				2
НЕ 1.2. (Лекція) Циклічні програми в Python.	3	1				2
НЕ 1.3. (Лекція) Списки та кортежі в Python.	3	1				2
НЕ 1.4. (Лекція) Символи та рядки в Python.	4	2				2
НЕ 1.5. (Лекція)	4	2				2

Словники та множини в Python.						
НЕ 1.6. (Лабораторне заняття) Лінійні та розгалужені програми в Python.	4			2		2
НЕ 1.7 (Лабораторне заняття) Циклічні програми в Python.	6			4		2
НЕ 1.8. (Лабораторне заняття) Списки та кортежі в Python.	6			4		2
НЕ 1.9. (Лабораторне заняття) Символи та рядки в Python.	6			4		2
НЕ 1.10. (Лабораторне заняття) Словники та множини в Python.	4			2		2
Разом за змістовим модулем 2	44	8		16		20
	Змістовий модуль 2					
НЕ 2.1. (Лекція) Обробка виключень у Python.	5	2				3
НЕ 2.2. (Лекція) Підпрограми в Python.	5	2				3
НЕ 2.3. (Лекція) Файли в Python.	4	1				3
НЕ 2.4. (Лекція) Модулі в Python.	4	1				3
НЕ 2.5. (Лекція) Об'єктно-зорієнтоване програмування в Python.	4	1				3
НЕ 2.6. (Лабораторна робота) Обробка виключень у Python.	6			4		2
НЕ 2.7. (Лабораторна	6			4		2

робота) Функції у Python.						
НЕ 2.8. (Лабораторна робота) Файли у Python.	7			4		3
НЕ 2.9. (Лабораторна робота) Модулі у Python.	5			2		3
Разом за змістовим модулем 2	46	7		14		25
ВСЬОГО	90	15		30		45

5.3. Зміст завдань для самостійної роботи

Самостійна робота складається з повторення матеріалу, засвоєного на лекціях, самостійного опанування частини теоретичного матеріалу, роботи з контрольними запитаннями та завданнями.

Студент може індивідуально виконувати додаткові завдання навчально-дослідницької спрямованості за завданнями, наданими викладачем.

Студенти можуть отримати до 10 балів в рахунок ІНДЗ, якщо самостійно зареєструються на безкоштовному курсі платформи Prometheus "Основи програмування (мова Python)" або на курсах з Python платформи Coursera, пройдуть навчання, отримають відповідний сертифікат і надішлють його на сайт дистанційного навчання викладачу разом зі скріншотом успішності на курсі. Кількість балів буде виставлена пропорційно до навчальних результатів студента (згідно зі статистикою сайта Prometheus або Coursera).

6. Система контролю та оцінювання

Види та форми контролю

Формами поточного контролю є усна чи письмова (тестування, реферат, лабораторна робота, ІНДЗ) відповідь студента.

Формою підсумкового контролю є залік.

Засоби оцінювання

Усний контроль у вигляді індивідуального та фронтального опитування на лекціях та лабораторних заняттях, захист лабораторних робіт та індивідуального навчально-дослідницького завдання; письмовий контроль у вигляді контрольних робіт, тестів, підсумкове тестове опитування.

Критерії оцінювання результатів навчання з навчальної дисципліни

Критерієм успішного проходження здобувачем освіти підсумкового оцінювання є досягнення ним мінімальних порогових рівнів оцінок за кожним запланованим результатом навчання навчальної дисципліни.

Мінімальний пороговий рівень оцінки визначається за допомогою якісних критеріїв і трансформується в мінімальну позитивну оцінку використовуваної числової (рейтингової) шкали.

Розподіл балів, які отримують студенти

Поточне тестування та самостійна робота									Модуль-контроль	Сума
Змістовий модуль №1					Змістовий модуль № 2					
НЕ 1.6	НЕ 1.7	НЕ 1.8	НЕ 1.9	НЕ 1.10	НЕ 2.6	НЕ 2.7	НЕ 2.8	НЕ 2.9		
5	5	7	7	6	7	8	7	8	40	100

Шкала оцінювання: національна та ЄКТС

Оцінка за національною шкалою	Оцінка за шкалою ECTS	
	Оцінка (бали)	Пояснення за розширеною шкалою
Відмінно	A (90-100)	відмінно
Добре	B (80-89)	дуже добре
	C (70-79)	добре
Задовільно	D (60-69)	задовільно
	E (50-59)	достатньо
Незадовільно	Fx (35-49)	(незадовільно) з можливістю повторного складання
	F (1-34)	(незадовільно) з обов'язковим повторним курсом

7. Рекомендована література

7.1. Основна

1. Креневич А.П. Python у прикладах і задачах. Частина 1. Структурне програмування. Навчальний посібник із дисципліни "Інформатика та програмування".– К.: ВПЦ "Київський Університет", 2017. – 206 с.
2. Мэтиз Эрик. Изучаем Python. Программирование игр, визуализация данных, веб-приложения.– СПб.: Питер, 2017.–496 с.
3. Зед Шоу. Легкий способ выучить Python.– М.: Эксмо, 2017.– 352 с.
4. The Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/tutorial/index.html.
5. Навчальні матеріали: Python [Електронний ресурс] – Режим доступу до ресурсу: http://www.matfiz.univ.kiev.ua/pages/13.

7.2. Допоміжна

6. Орлов С. А. Технологии разработки программного обеспечения. Разработка сложных программных систем [Текст]: учеб. пособие для вузов по направлению "Информатика и вычисл. техника" / Сергей Александрович Орлов. – СПб.: Питер, 2002. – 463 с.
7. Прохоренок Н. А. Python 3 и PyQt. Разработка приложений / Николай Анатольевич Прохоренок. – СПб: БХВ-Петербург, 2012. – 704 с.
8. Васильев А. Н. Python на примерах. Практический курс по программированию / А. Н. Васильев. – СПб.: Наука и техника, 2016. – 432 с.
9. Інформатика (профільний рівень): підручник для 10 класів закладів загальної середньої освіти України / В.Д. Руденко, Н.В. Речич, В.О. Потієнко.– Харків: Вид-во “Ранок”, 2018.– 255 с.

10. Інформатика (профільний рівень): підручник для 11 класів закладів загальної середньої освіти України / В.Д. Руденко, Н.В. Речич, В.О. Потієнко.– Харків: Вид-во “Ранок”, 2019.– 256 с.

8. Інформаційні ресурси

http://moodle.chnu.edu.ua
http://www.python.org
http://www.matfiz.univ.kiev.ua/pages/13

Список використаної літератури

1. Креневич А.П. Python у прикладах і задачах. Частина 1. Структурне програмування. Навчальний посібник із дисципліни "Інформатика та програмування".– К.: ВПЦ "Київський Університет", 2017. – 206 с.
2. The Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/tutorial/index.html.
3. Навчальні матеріали: Python [Електронний ресурс] – Режим доступу до ресурсу: http://www.matfiz.univ.kiev.ua/pages/13.
4. Пориньте в Python 3. Вікіпідручник [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikibooks.org/wiki/%d0%9f%d0%be%d1%80%d0%b8%d0%bd%d1%8c%d1%82%d0%b5_%d1%83_python_3/%d0%92%d0%b5%d1%80%d1%81%d1%96%d1%8f_%d0%b4%d0%bb%d1%8f_%d0%b4%d1%80%d1%83%d0%ba%d1%83
5. Мізюк Олександр. Путівник мовою програмування Python [Електронний ресурс] – Режим доступу до ресурсу: http://pythonguide.rozh2sch.org.ua/
6. Маттес Ерік. Пришвидшений курс Python.– Львів: Видавництво Старого Лева, 2021.– 600 с.
7. Інформатика (профільний рівень): підручник для 10 класів закладів загальної середньої освіти України / В.Д. Руденко, Н.В. Речич, В.О. Потієнко.- Харків: Вид-во “Ранок”, 2018.- 255 с.
8. Інформатика (профільний рівень): підручник для 11 класів закладів загальної середньої освіти України / В.Д. Руденко, Н.В. Речич, В.О. Потієнко.- Харків: Вид-во “Ранок”, 2019.- 256 с.

9. Семчук А.Р., Юрченко І.В. Економічна інформатика: 7-е видання, доповнене.– Чернівці: МВІЦ "Місто", 2008.– 426 с. (Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів. Лист-погодження № 14/18.2–2089 від 21.09.2004).
10. Юрченко І.В. Інформатика та програмування. Частина 1. Навчальний посібник.– Чернівці: Книги–XXI, 2011.– 203 с.
11. Юрченко І.В., Сікора В.С. Інформатика та програмування. Частина 2. Навчальний посібник.– Чернівці: Видавець Яворський С.Н., 2015.– 210 с.
12. Програмування. Практикум / Укл.: Семенюк А.Д., Сопронюк Ф.О. – Чернівці: Рута, 2001.– 143 с.
13. Караванова Т.П., Любарщук Є.А., Скутар І.Д. Лабораторні роботи з програмування. Частина 1.– Чернівці: ЧНУ, 2007.– 42 с.

ЗМІСТ

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

Програмування мовою Python
Навчальний посібник

Юрченко Ігор Валерійович
Сікора Віра Степанівна

Публікується в авторській редакції

Підписано до друку 20.01.2022 р. Формат 60×84/16.
Ум. друк. арк. 6,04 Зам. № 94. Тираж 100 прим.

Виготовлено з готового оригінал-макета замовника.
Виготівник: Яворський С.Н.
Свідоцтво суб'єкта видавничої справи ЧЦ №18 від 17.03.2009 р.
58000, м.Чернівці, вул, Івана Франка, 20, офіс 12, тел. (0372) 55-05-85